Evolutionary Variational Optimization for Probabilistic Unsupervised Learning

Thesis by

Jakob Drefs

Accepted as Fulfilling the Requirements

for the Degree of

Doctor of Natural Sciences

in Faculty V - Mathematics and Science of

Carl von Ossietzky University of Oldenburg

2022

Evolutionary Variational Optimization for Probabilistic Unsupervised Learning
Doctoral Thesis by Jakob Drefs
Born in Oldenburg (Oldb), Germany
on August 1st, 1990

Promotion Committee

Chair:	Prof. Dr. Jörg Lücke, Carl von Ossietzky University of Oldenburg
2nd Reviewer:	Prof. Dr. Marius Lindauer, Leibniz University Hannover
3rd Reviewer:	Prof. Dr. Alexander K. Hartmann, Carl von Ossietzky University of Oldenburg

Date of Defense

November 4, 2022

Abstract

Probabilistic generative modeling is a powerful machine learning paradigm suitable for a variety of tasks, such as missing value estimation, denoising, clustering, outlier detection, or compression. A key factor determining the efficiency and effectiveness of algorithms based on generative models is the internally used approximate inference scheme. This work focuses on inference and learning in generative models with binary latents and studies a novel approach that combines variational optimization of truncated posteriors with evolutionary algorithms. As a distinction to related Gaussian and factored (a.k.a., mean field) variational approximations, the use of truncated variational distributions allows for flexible adaptation to complex posterior landscapes with multiple modes and correlations. Through the application of evolutionary algorithms as integral part of the variational optimization scheme, the approach becomes generically applicable without the need for model-specific analytical derivations ('black box' inference). Distinctions to related 'black box' inference schemes include that the approach studied here does neither involve sampling approximations nor gradient-based optimization of variational parameters, thereby circumventing auxiliary mechanisms such as continuous relaxations of discrete distributions and variance reduction techniques required by stochastic gradient estimators. Compared to previous truncated variational approximation schemes, the approach studied has the distinguishing feature of guaranteeing a monotonic increase of a log-likelihood lower bound. The studies presented in this thesis consider applications of the novel evolutionary variational optimization to a variety of data models with a wide range of characteristics, including: binary and binary-continuous priors, binary and continuous noise models, shallow and deep model architectures, linear and non-linear latent interaction models including learnable and non-learnable non-linearities, and global and latent-specific variance encodings. When studying deep generative models, the evolutionary variational optimization is combined with automatic differentiation tools for model parameter optimization, thereby demonstrating the suitability of the investigated approach for 'black box' learning and inference. The numerical experiments presented focus on image patch modeling and provide extensive performance evaluations using established denoising and inpainting benchmarks that allow for comparison to a broad range of related methods. These investigations reveal that evolutionary variational optimization of expressive data models such as spike-and-slab sparse coding or variational autoencoders results in competitive performances in benchmark settings with only a single noisy image available for training. Analyses of the learned data representations show that the models use comparably sparse encodings in theses cases. Experiments with SARS-CoV-2 microscopy imaging data illustrate potential contributions that the developed methods may deliver in applications, for instance, possibly improving visualization of image details to potentially ease image interpretation. In general, this work highlights the importance of effective procedures for learning and inference in generative models, and opens the door to a variety of possible future research directions.

Zusammenfassung

Probabilistische generative Modellierung ist ein leistungsfähiges Paradigma des maschinellen Lernens, das sich für eine Vielzahl von Aufgaben eignet, z. B. Schätzung fehlender Werte in unvollständigen Daten, Entrauschung, Clustering, Ausreißererkennung oder Komprimierung. Ein Schlüsselfaktor für die Effizienz und Effektivität von Algorithmen, die auf generativen Modellen basieren ist das intern verwendete Inferenz Approximationsschema. Diese Arbeit konzentriert sich auf Inferenz und Lernen in generativen Modellen mit binären latenten Variablen und untersucht einen neuartigen Ansatz, der Variationsoptimierung von trunkierten Posterior-Verteilungen mit evolutionären Algorithmen kombiniert. Im Unterschied zu verwandten Gauß'schen und faktorisierten (auch bekannt als Mean Field) Variationsapproximationsverfahren erlaubt die Verwendung von trunkierten Variationsverteilungen eine flexible Anpassung an komplexe Posterior-Landschaften mit mehreren Modi und Korrelationen. Durch die Anwendung von evolutionären Algorithmen als integraler Bestandteil des Variationsoptimierungsschemas wird der Ansatz generisch anwendbar, ohne dass modellspezifische analytische Herleitungen erforderlich sind ('Black-Box'-Inferenz). Zu den Unterschieden zu verwandten 'Black-Box'-Inferenzschemata gehört, dass der hier untersuchte Ansatz weder Sampling-Approximationen noch gradientenbasierte Optimierung von Variationsparametern beinhaltet, wodurch Hilfsmechanismen wie kontinuierliche Relaxierungen diskreter Verteilungen und Varianzreduktionstechniken, die von stochastischen Gradientenschätzern benötigt werden, umgangen werden. Im Vergleich zu früheren trunkierten Variationsapproximationsverfahren zeichnet sich der untersuchte Ansatz dadurch aus, dass er einen monotonen Anstieg einer unteren Schranke der Log-Likelihood garantiert. Die in dieser Arbeit vorgestellten Studien betrachten Anwendungen der neuartigen evolutionären Variationsoptimierung auf eine Vielzahl von Datenmodellen mit einem breiten Spektrum von Eigenschaften, darunter: binäre und binär-kontinuierliche Priormodelle, binäre und kontinuierliche Rauschmodelle, flache und tiefe Modellarchitekturen, lineare und nicht-lineare Interaktionsmodelle der latenten Variablen einschließlich lernbarer und nicht-lernbarer Nichtlinearitäten sowie globale und latent-spezifische Varianzkodierungen. Bei der Untersuchung tiefer generativer Modelle wird die evolutionäre Variationsoptimierung mit automatischen Differenzierungswerkzeugen zur Modellparameteroptimierung kombiniert, wodurch die Eignung des untersuchten Ansatzes für 'Black-Box'-Lernen und -Inferenz demonstriert wird. Die vorgestellten numerischen Experimente konzentrieren sich auf die Modellierung von Bildfeldern und bieten umfangreiche Leistungsevaluationen unter Verwendung etablierter Denoising- und Inpainting-Benchmarks, die einen Vergleich mit einer breiten Palette verwandter Methoden ermöglichen. Diese Untersuchungen zeigen, dass die evolutionäre Variationsoptimierung von ausdrucksstarken Datenmodellen wie Spike-and-Slab Sparse Coding oder Variations-Autoencodern zu konkurrenzfähigen Leistungen in Benchmark-Settings führt, bei denen nur ein einziges verrauschtes Bild zum Training zur Verfügung steht. Analysen der gelernten Datenrepräsentationen zeigen, dass die Modelle in diesen Fällen vergleichbar spärliche Kodierungen verwenden. Experimente mit SARS-CoV-2-Mikroskopie-Bilddaten veranschaulichen potenzielle Beiträge, die die entwickelten Methoden bei Anwendungen liefern könnten, z. B. eine mögliche Verbesserung der Visualisierung von Bilddetails zur potentiellen Erleichterung der Bildinterpretation. Im Allgemeinen unterstreicht diese Arbeit die Bedeutung effektiver Verfahren für Lernen und Inferenz in generativen Modellen und öffnet die Tür zu einer Vielzahl möglicher zukünftiger Forschungsrichtungen.

Declaration of Authorship

I, Jakob Drefs, declare that this thesis and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this university.
- I am aware of the guidelines of good scientific practice of the Carl von Ossietzky University of Oldenburg and observed them.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- I have not availed myself of any commercial placement or consulting services in connection with my promotion procedure.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Jakob Drefs

Acknowledgments

I am deeply grateful to all those who have supported me in completing this work.

To Jörg Lücke for the opportunity, trust and freedom to let me develop and work as a researcher in his group. I greatly appreciate his constant, enlightening and encouraging advice.

To Stefan Harfst for extremely helpful technical support with high performance computing clusters.

To Enrico Guiraud, Hamid Mousavi, and Sebastian Salwig for fruitful collaborations, and also to all other members of the Machine Learning Lab at the University of Oldenburg for interesting discussions from which I learned a lot, and many meals together that I enjoyed very much.

To my sister and parents for their unconditional support and encouragement.

Contents

Title Page i

Abstract iii

Declaration of Authorship vii

Acknowledgments ix

Contents xi

1 Introduction 1

1.1 Outline 4

2 Evolutionary Variational Optimization 7

Abstract 7

Author Contributions 7

Acknowledgments 8

2.1 Introduction 9

2.2 Evolutionary Variational Optimization 10

2.2.1 Evolutionary Optimization of Truncated Posteriors 11

2.2.2 Related Work on Applications of Evolutionary Approaches to Learning 14

2.2.3 Data Estimator 15

2.3 Application to Generative Models, Verification and Scalability 16

2.3.1 Used Generative Models 16

2.3.2 Verification: Recovery of Generating Parameters 18

2.3.3 Scalability to Large Generative Models 19

2.4 Performance Comparison on Denoising and Inpainting Benchmarks 21

2.4.1 Algorithm Types and Comparison Categories 22

2.4.2 Image Denoising . . . 22

2.4.2.1 Relation Between PSNR Measure and Variational Lower Bound . . . 22

2.4.2.2 Comparison of Generative Models and Approximate Inference 25

2.4.2.3 General Comparison of Denoising Approaches . . . 27

2.4.3 Image Inpainting . . . 31

2.5 Discussion . . . 33

2.S1 Log-Pseudo Joint Formulation . . . 38

2.S2 Sparsity-Driven Bitflips . . . 39

2.S3 Data Estimator . . . 39

2.S4 M-step Update Equations . . . 41

2.S5 Evaluation Criteria for Image Restoration . . . 42

2.S6 Details on Numerical Experiments . . . 44

2.S6.1 Soft- and Hardware . . . 44

2.S6.2 Hyperparameters . . . 47

2.S6.3 Comparison to Other Generative Models – Details . . . 48

3 Zero-Shot Microscopy Image Enhancement 52

Abstract . . . 52

Author Contributions . . . 53

Acknowledgments . . . 53

3.1 Introduction . . . 54

3.2 Results . . . 56

3.2.1 Denoising TEM Images of SARS-CoV-2 Infected Cell Cultures . . . 56

3.2.2 Variances of Non-Noisy Pixel Estimations . . . 59

3.2.3 Zero-Shot Denoising Fluorescence Microscopy Images . . . 60

3.3 Discussion . . . 60

3.4 Methods . . . 64

3.4.1 Generative Model Algorithms . . . 64

3.4.2 Probabilistic Data Estimation . . . 65

3.4.3 Estimation of Non-Noisy Image Pixels . . . 66

3.4.4 Evaluation Metrics . . . 66

3.S1 Derivations . . . 68

3.S1.1 EM Update Rules Spike-and-Slab Sparse Coding 68
3.S1.2 EM Update Rules Gamma-Poisson Mixtures 69
3.S1.3 Data Estimator Gamma-Poisson Mixtures 73
3.S2 Details on Numerical Experiments 74
3.S2.1 Datasets 74
3.S2.2 Implementations and Hyperparameters 75
3.S2.3 Further Results 76

4 Truncated Variational Autoencoders 80

Abstract 80
Author Contributions 81
Acknowledgments 81
4.1 Introduction and Related Work 82
4.1.1 Related Work 83
4.2 Direct Variational Optimization 84
4.2.1 Optimization of the Encoding Model 85
4.2.2 Evolutionary Search 86
4.2.3 Optimization of the Decoding Model 87
4.3 Numerical Experiments 88
4.3.1 Verification and Scalability 88
4.3.2 Denoising - Controlled Conditions 88
4.3.3 Denoising - Uncontrolled Conditions 90
4.3.4 Inpainting 91
4.4 Discussion 92
4.S1 Details of Encoder and Decoder Optimization 94
4.S1.1 Computational Complexity 96
4.S1.2 Evolutionary Variational Optimization 97
4.S1.3 Numerical Evaluation in Related Work 98
4.S2 Details on Numerical Experiments 98
4.S2.1 Hyperparameters 98
4.S2.2 Verification Experiments 99
4.S2.3 Scalability Experiments 101
4.S2.4 Image Denoising 101

4.S2.5 Image Inpainting 107
4.S2.6 Single-Object Image Data 108
4.S2.7 Zero-Shot Audio Denoising 109

5 Double-Dictionary Maximal Causes Analysis 112
Abstract 112
Author Contributions 112
Acknowledgments 113
5.1 Introduction 114
5.2 Methods 114
5.2.1 A Non-Linear Sparse Coding Model with Mean and Variance Dictionaries 114
5.2.2 Parameter Optimization 115
5.3 Numerical Experiments 117
5.3.1 Bars Test 117
5.3.2 Encoding of Natural Image Patches 117
5.4 Conclusion 118

6 Discussion and Conclusion 122
6.1 Outlook 125

Publication List 128

Bibliography 130

Chapter 1

Introduction

Fueled by the increasing availability of data and computational resources in recent decades, machine learning (ML) algorithms have been very successfully applied to a wide range of information processing tasks, for which they have received considerable attention. In contrast to rule-based algorithms, whose behavior is determined by hand-crafted protocols of processing steps, ML methods build internal models of a given problem from data, and they are consequently also referred to as data-driven algorithms. Prominent examples of the many different types of problems to which ML approaches have successfully been applied include, e.g., computer vision applications such as image generation (e.g., Goodfellow et al., 2014; Kos et al., 2018; van den Oord et al., 2016), image denoising (e.g., Krull et al., 2019a; Mairal et al., 2009; Zhang et al., 2017), image inpainting (e.g., Pathak et al., 2016; Roth and Black, 2009; Zoran and Weiss, 2011), or image super-resolution (e.g., Chen and Pock, 2017; Lim et al., 2017; Shocher et al., 2018). Likewise, ML methods have been applied to audio signal processing problems such as speech separation (e.g., Luo and Mesgarani, 2019; Wang and Chen, 2018), or speech enhancement (e.g., Kumar and Florencio, 2016; Nustede and Anemüller, 2021; Tammen and Doclo, 2021; Westhausen and Meyer, 2020; Xia et al., 2020). ML approaches have furthermore become integral parts of recommender systems used, e.g., on online platforms for fashion retail (e.g., Freno, 2017), lodging (e.g., Haldar et al., 2019), music (e.g., van den Oord et al., 2013) or video streaming (e.g., Covington et al., 2016). Medicine represents yet another application area in which ML methods have gained increasing attention, e.g., for assisting with tasks such as disease detection (e.g., Sarki et al., 2020) or autonomous surgery (e.g., Kassahun et al., 2016).

Widespread are discriminative methodologies which learn direct task-specific mappings from input to target data. In a standard problem such as classification, discriminative models might, for instance, be optimized to map pixel values of given images to names of depicted objects. Or discriminative models might be optimized to map, e.g., waveform amplitudes of sound recordings to the waveform amplitudes of involved sound sources. A strength of discriminative approaches is that, once trained, they can be very fast at test time. Training, on the other hand, might be significantly more time consuming and typically involves supervision: A supervised classification algorithm, for instance, would be fed with paired examples of images and associated object names to be identified during training. Supervised learning consequently relies on the availability of, typically large amounts of, training data including data for supervision such as labels (e.g., object identities in classification problems) or other forms of targets (e.g., clean data in data enhancement problems). Being tailored to a specific task, discriminative approaches are likely to need retraining, in the worst case from scratch, if the task to be solved is slightly modified or if the algorithm is to be applied to data that is considerably different from the data used for training.

Complementing discriminative methodologies are generative approaches, which, in a probabilistic setting, aim at capturing a data density model of the data generating process such that data generated by the model becomes increasingly similar to the training data. A probabilistic generative data representation has the advantage of being suitable for a variety of tasks in addition to data generation, including, e.g., missing value estimation, denoising, clustering, outlier detection, or compression. Generative representations can usually be learned without supervision (while supervised learning is also possible), which is a key feature in scenarios in which labeled data sets cannot be acquired. Their increased flexibility is accompanied by an increased computational cost for training, making large scale applications of algorithms based on generative models challenging in general. Moreover, learning a density model might, for certain tasks, turn out to be disproportionately complex and not necessarily guarantee performance improvements over task-optimized discriminative methods (e.g., Jebara, 2004; Ulusoy and Bishop, 2005). As yet another aspect, discriminative approaches may be at risk of being perceived as 'black box' systems when the internally learned models turn out to be incomprehensible to humans. Variables capturing factors of the data generating process with an explicit treatment of uncertainty, on the other hand, may appear more illustrative and ease interpretability, in particular when they are aligned with constraints resulting from domain knowledge (e.g., characteristics of data generating factors such as non-negativity, discreteness, or circularity). Especially in the context of high-stakes ML applications, interpretability is considered increasingly important (compare, e.g., Murdoch et al., 2019; Rudin, 2019).

Generative approaches are commonly defined in terms of a latent variable model (LVM) which distinguishes between observable data and underlying latent causes of the data generating process. Simplistically stated, observables model measurable quantities (e.g., the pixel values of an image), while latents describe compositional elements of the measured data (e.g., the presence/absence or positions/sizes of depicted objects, or elementary features such as edges or textures in the image). Concepts captured by latent variables are, however, not necessarily that illustrative per se, and in fact disentanglement of latent codes into interpretable features is subject of current research (e.g., Higgins et al., 2017; Tonolini et al., 2020). LVM architectures exist in a wide variety of variants, many of which can be described by a directed acyclic graph describing interactions between latents and observables. Well-known examples include, e.g., sparse coding approaches (SC; Olshausen and Field, 1996) or, as a more recent example, deep generative models such as variational autoencoders (VAEs; Kingma and Welling, 2014; Rezende et al., 2014). SC was originally discussed in a neurobiological context as a model of encoding strategies in mammalian visual cortex. The term sparse thereby refers to the concept that only a small fraction of all neurons are assumed to be involved in the processing of individual sensory inputs. Standard SC approaches model sparsely active latents by continuous variables which are defined to follow heavy-tailed, e.g. Laplace, distributions. Standard VAE approaches, on the other hand, likewise use continuously-valued latents, however assuming Gaussianity for their activation. As a further similarity between SC and VAE approaches, standard forms of both models assume Gaussian observation noise. While standard SC approaches employ linear superposition rules for modeling latent interactions, a key characteristic of VAEs is their use of deep neural networks (DNNs) to parameterize non-linear mappings from latent to observed space. For both SC and VAEs, extensions of the standard models have been investigated which replaced continuous by discrete latents (e.g., Exarchakis and Lücke, 2017; Haft et al., 2004; Henniges et al., 2010; Jang et al., 2017; Maddison et al., 2017; Shelton et al., 2011). Formal introductions of both models will be given in Chs. 2 and 4.

Not less important than the design of an appropriate generative model architecture is the choice of efficient and effective procedures for optimizing its parameters given data. A natural choice in a probabilistic setting are maximum likelihood based approaches which seek parameters that maximize the evidence of the data under the density model. Direct likelihood maximization is often intractable and tackled with indirect approaches, for example based on expectation maximization (EM; Dempster et al., 1977). EM optimizes a lower bound of the likelihood in an iterative two-step procedure, with an integral part of the algorithm being the evaluation of expectation values w.r.t. posteriors over the latent variables. For most non-trivial models, this step turns out to be intractable and to require approximations, leading to the field of approximate inference. Approximate inference schemes are broadly distinguishable based on whether deterministic or stochastic strategies are employed. Prominent examples of the former category include, e.g., maximum a-posteriori (MAP) approaches, which approximate posteriors via point estimates (e.g., Bradley and Bagnell, 2008; Lee et al., 2007; Lewicki and Sejnowski, 2000; Mairal et al., 2009; Olshausen and Field, 1997), and approaches based on variational approximations (e.g., Jordan et al., 1999; Neal and Hinton, 1998; Opper and Winther, 2005; Wainwright and Jordan, 2008). Stochastic strategies, on the other hand, are typically based on sampling approximations of expectation values (e.g., Mohamed et al., 2012; Zhou et al., 2009, 2012), and they have also been combined with variational strategies (e.g., Hoffman et al., 2013; Kingma and Welling, 2014; Rezende et al., 2014).

While MAP approaches may appear appealing in terms of computational efficiency, they are, by definition, severely limited in modeling the typically complex structure of the posterior density. Sampling approaches, in contrast, allow, in principle, for approximating posterior expectations with high accuracy, however at the cost of increasingly large sample sizes, which tendentiously slow down convergence speed, particularly in high dimensional latent spaces (compare, e.g., Bishop, 2006; Shelton et al., 2011; Zhou et al., 2012). Variational procedures, on the other hand, introduce functionals which are optimized to approximate the log-likelihood as closely as possible. As a distinction to sampling-based approaches, the design of variational distributions for these functionals typically involves that assumptions about the posterior structure are made, for example Gaussianity (e.g., Kingma and Welling, 2014; Opper and Winther, 2005; Rezende et al., 2014), factored forms (e.g., Jordan et al., 1999; Titsias and Lázaro-Gredilla, 2011), or even both. Compared to MAP approaches, variational approximations are able to provide significantly more accurate posterior approximations. Variational approaches that include assumptions that are inconsistent with the properties of the true posterior (e.g., neglect of posterior correlations as a consequence of imposing a factored form for the variational distribution) may, however, still result in a poor approximation.

Truncated posteriors (Lücke and Eggert, 2010; Lücke and Sahani, 2008) have been proposed as a class of discrete variational distributions that neither contain the assumption of a-posteriori independence of the latents nor assume a mono-modal form of the posterior distribution. In turn, truncated posteriors are based on the assumption that significant portions of the posterior mass are concentrated on small subsets of the latent space. Truncation refers to the principle of neglecting (potentially large) parts of the latent space (ideally those with smallest posterior mass) through exact zeros in the posterior approximation (a formal definition will be given later, see, e.g., Sec. 2.2.1). Previous work has shown that truncated posteriors enable effective optimization of standard as well as intricate generative models (e.g., Dai and Lücke, 2014; Henniges et al., 2014; Sheikh et al., 2014; Shelton et al., 2015) while providing high functional performance (e.g., Sheikh et al., 2014) and scalability to very high-dimensional latent spaces (Hirschberger et al., 2022; Sheikh and Lücke, 2016). The key element of truncated variational techniques are efficient search strategies for determining latent subspaces with concentrated posterior mass. Strategies that have been explored so far can be broadly divided into deterministic approaches using feed-forward selection functions (e.g., Exarchakis and Lücke, 2017; Henniges et al., 2010, 2014; Lücke and Eggert, 2010; Sheikh et al., 2014, 2019), combinations of deterministic selection with sampling (Sheikh and Lücke, 2016; Shelton et al., 2011, 2015), and fully variational optimization of truncated posteriors (Exarchakis et al., 2022; Hirschberger et al., 2022; Lücke and Forster, 2019; Lücke et al., 2018). The former strategy is an early version of amortized inference (Kingma and Welling, 2014; Mnih and Gregor, 2014; Rezende et al., 2014), which enables efficiency. The latter strategy offers several functional advantages, however: It allows to reformulate the optimization objective, i.e. the variational lower bound of the log-likelihood, into a very simple and efficiently computable form, which allows to define procedures that guarantee a monotonic increase of the objective. Monotonicity thereby represents a distinguishing feature compared to the other aforementioned strategies for which decreases of the optimization objective during training can not be excluded.

Motivated by the various successful previous applications of truncated posterior approximations, this thesis investigates a novel fully variational training scheme based on truncated posteriors with two distinguishing features compared to previous related methods: First, the novel approach studied here links variational optimization to a very different yet powerful optimization technique, namely evolutionary algorithms (Bäck, 1996). Second, it dispenses entirely with model-specific derivations; applicability, in turn, depends solely on the model's joint probability distribution being analytically tractable. As such, the novel method, entitled *Evolutionary Variational Optimization* (EVO), paves the way for 'black box' inference in generative models with binary latents.

In a series of studies presented in this thesis, EVO's versatility will be tested by applying the method to a variety of data models covering a wide range of characteristics, including: (i) binary and binary-continuous priors, (ii) binary and continuous noise models, (iii) shallow and deep model architectures, (iv) linear and non-linear latent interaction models including learnable and non-learnable non-linearities, and (v) global and latent-specific variance encodings. The numerical experiments presented will focus on image patch modeling and investigate EVO's functional efficacy based on standard benchmarks that allow for comparisons to other approximate inference schemes including (i) factored variational approaches, (ii) sampling-based procedures, (iii) stochastic variational inference schemes, and (iv) previous truncated variational methods. In addition, the considered benchmarks will allow for extensive comparison to recent and standard image processing algorithms (which are not necessarily based on probabilistic generative models).

1.1 Outline

Chapter 2 develops Evolutionary Variational Optimization (EVO), thereby setting the foundation for all studies presented in this thesis. The chapter investigates the application of EVO to three data models, all of which use a shallow architecture and analytical solutions for model parameter update rules. The concretely investigated models are: Binary Sparse Coding (BSC; Haft et al., 2004; Henniges et al., 2010), Noisy-OR Bayes Nets (NOR; e.g., Jernite et al., 2013; Rotmensch et al., 2017; Šingliar and Hauskrecht, 2006), and Spike-and-Slab Sparse Coding (SSSC; e.g., Goodfellow et al., 2012; Sheikh et al., 2014; Titsias and Lázaro-Gredilla, 2011). After demonstrating EVO's ability to accurately recover ground-truth generating parameters using artificial data, the study presented extensively evaluates performance on standard image denoising and inpainting benchmarks. Systematic comparisons to other approximate inference schemes reveal that EVO offers performance improvements over sampling-based methodologies (Zhou et al., 2012), and variational approaches using factored posterior assumptions (Titsias and Lázaro-Gredilla, 2011), preselection procedures (Sheikh et al., 2014) and stochastic gradient estimators for discrete variables (Jang et al., 2017). Evolutionary variational optimization of SSSC models (termed ES3C), in particular, is shown to establish novel state-of-the-art performances in benchmark settings with only a single noisy image available for training (referred to as 'zero-shot' setting) and, moreover, to be competitive in comparison to models that exploit more a-priori information.

Chapter 3 presents a case-study that arose in the context of the COVID-19 pandemic and exemplifies how the ES3C algorithm developed in Ch. 2 may be leveraged in the context of a concrete application. The study focuses on the specific problem of improving the visualization of details in electron micrographs of SARS-CoV-2 infection scenes and asks how 'zero-shot' image enhancement algorithms can be exploited for this task. 'Zero-shot' approaches are generally appealing in this context given that microscopy imaging data is available exclusively in the form of noisy images, implying that enhancement algorithms necessarily require the ability of being directly applicable to noisy data. The study considers a variety of enhancement algorithms including, besides ES3C, a novel approach based on Gamma-Poisson mixtures (Monk et al., 2018), and, as competing methods, a convolutional variational autoencoder (Prakash et al., 2021b), two non-generative denoising neural networks (Krull et al., 2019a; Quan et al., 2020a), and a standard, non-data-driven baseline (Dabov et al., 2007). All of these algorithms seem to provide improved visualizations of structural details that are difficult to identify in the raw noisy electron micrographs. The study, moreover, suggests that the visualization of fine details such as the SARS-CoV-2 spike protein may be further enhanced through non-standard higher-order statistics of image reconstruction. The chapter concludes by discussing how data-driven algorithms can facilitate the interpretation of microscopy imaging data and thereby potentially contribute to improving the understanding of infectious diseases such as SARS-CoV-2 (which represented a challenge of very significant societal relevance at the time of writing).

Chapter 4 focuses on deep generative modeling and investigates evolutionary variational optimization of variational autoencoders (VAEs; Kingma and Welling, 2014; Rezende et al., 2014) with binary latents. Specifically, the study presented in the chapter explores how EVO can be applied to replace the DNN-based encoding model of standard VAEs. As a consequence of dispensing with gradient-based optimization of variational parameters, EVO allows to bypass a range of auxiliary mechanisms commonly applied by standard VAE training methods such as sampling approximation, reparameterization, variance reduction, and softening of discrete latents. A further distinguishing feature is EVO's non-amortized inference scheme which optimizes variational parameters individually per data point; in contrast, standard VAEs typically train encoding models that are shared across data points. EVO-based training of binary latent VAEs (termed TVAE) is consequently associated with a higher computational cost in comparison to standard, amortized VAE approaches. The decoding model of TVAE is defined analogously to standard VAEs, i.e., latents map to observables through a DNN non-linearity whose parameters are optimized via gradient descent. Verification experiments with artificial data demonstrate that TVAE accurately recovers ground-truth generating parameters including correlations between data-generating causes. Applications to image patch data reveal that, in contrast to conventionally trained VAEs, TVAE uses a particularly sparse data encoding, which proves highly effective for 'zero-shot' denoising and inpainting tasks.

Chapter 5 marries evolutionary variational optimization with maximal causes analysis (MCA; Bornschein et al., 2013; Lücke and Sahani, 2008; Puertas et al., 2010). The study presented in the chapter explores a novel MCA variant that replaces the global variance parameter used by former MCA and standard SC approaches (as well as by the BSC, SSSC and VAE models considered in Chs. 2 to 4) with a latent-specific variance encoding. Accordingly, the parameterization of the investigated model comprises two dictionaries, one for component means and another one for component variances. The dictionary elements are combined according to a maximum non-linearity. The maximum non-linearity of the double-dictionary model represents a distinction to the NOR model considered in Ch. 2, which uses a noisy-OR non-linearity, and to the VAEs discussed in Ch. 4, which use a non-linearity parameterized by a DNN. After deriving a learning rule for the variance dictionary and numerically verifying it on artificial data, the study deploys EVO to learn a double-dictionary encoding with 1,000 components from natural image patch data.

Chapter 6 provides a concluding discussion of the results presented in this thesis and points to possible future research directions.

Chapters 2 to 5 correspond to paper manuscripts that resulted from collaborative research work and have either been published or accepted for publication in peer-reviewed venues in the form of journal articles or conference proceedings or are in revision for publication at the time of writing. The full bibliographic reference and author contributions will be provided in the beginning of each of these chapters. A complete list of publications, including work that is not part of this thesis, is provided in the Publication List at the end of this thesis.

Chapter 2

Evolutionary Variational Optimization

This chapter has appeared as: Jakob Drefs, Enrico Guiraud, Jörg Lücke. Evolutionary Variational Optimization of Generative Models. *The Journal of Machine Learning Research*, 23(21):1-51, 2022.

Abstract. We combine two popular optimization approaches to derive learning algorithms for generative models: variational optimization and evolutionary algorithms. The combination is realized for generative models with discrete latents by using truncated posteriors as the family of variational distributions. The variational parameters of truncated posteriors are sets of latent states. By interpreting these states as genomes of individuals and by using the variational lower bound to define a fitness, we can apply evolutionary algorithms to realize the variational loop. The used variational distributions are very flexible and we show that evolutionary algorithms can effectively and efficiently optimize the variational bound. Furthermore, the variational loop is generally applicable ('black box') with no analytical derivations required. To show general applicability, we apply the approach to three generative models (we use Noisy-OR Bayes Nets, Binary Sparse Coding, and Spike-and-Slab Sparse Coding). To demonstrate effectiveness and efficiency of the novel variational approach, we use the standard competitive benchmarks of image denoising and inpainting. The benchmarks allow quantitative comparisons to a wide range of methods including probabilistic approaches, deep deterministic and generative networks, and non-local image processing methods. In the category of 'zero-shot' learning (when only the corrupted image is used for training), we observed the evolutionary variational algorithm to significantly improve the state-of-the-art in many benchmark settings. For one well-known inpainting benchmark, we also observed state-of-the-art performance across all categories of algorithms although we only train on the corrupted image. In general, our investigations highlight the importance of research on optimization methods for generative models to achieve performance improvements.

Keywords: Expectation Maximization, Variational Methods, Evolutionary Algorithms, Sparse Coding, Denoising, Inpainting

Author Contributions. Based on previous results of the Machine Learning lab at Oldenburg, the idea of exploring evolutionary algorithms for fully variational optimization of truncated posteriors was developed and translated into a concrete study design by all paper authors. The literature research related to image denoising and inpainting benchmarks was primarily carried out by Jakob Drefs (JD) with major contributions by Jörg Lücke (JL) and contributions by Enrico Guiraud (EG). The literature research for methodologies for learning in generative models and related evolutionary approaches was carried out by all authors with major contributions by JL and JD. The derivations related to probabilistic data estimation (Sec. 2.S3) were worked out by JD with input from JL. JD ran all numerical experiments with BSC and SSSC models as well as all Gumbel-softmax-related control experiments. EG performed all NOR-related experiments. The manuscript was written by JD and JL with contributions from EG; parts of the manuscript are based on Guiraud et al. (2018) which was written by EG and JL with contributions from JD. Figures and tables were created by JD with input from JL and EG. All authors discussed the results and approved the final version of the manuscript.

The software was developed collaboratively: The implementation of the core EVO training algorithm was designed and written jointly by EG and JD and initially deployed within separate libraries for BSC and SSSC models (written by JD, inspired by work by ProSper developers, 2019) and NOR (written by EG). The former library which implements EVO-based training of BSC and SSSC models has been open-sourced (EVO developers, 2022). Subsequent collaborative efforts of EG and JD paved the way to TVO, a software library for general Truncated Variational Optimization of binary latent generative models. Important contributions of JD to this library include implementations for multi-core CPU parallelization, implementations of BSC and SSSC models including data estimation routines, and tutorials showcasing example usages of the framework. JD furthermore contributed implementations for patch-based image reconstruction. The TVO library has been open-sourced (TVO developers, 2022).

Acknowledgments. We acknowledge support by the German Research Foundation (DFG) under Grant 352015383 (SFB 1330, B2). Furthermore, we acknowledge the support of Enrico Guiraud through a Wolfgang Gentner scholarship of the German Federal Ministry of Education and Research (BMBF, Grant 13E18CHA), and support through computing infrastructure by the HPC Cluster CARL of Oldenburg University (DFG, Grant INST 184/157-1 FUGG) and the North-German Supercomputing Alliance (HLRN, Project nim00006).

2.1 Introduction

Variational approximations (Jordan et al., 1999; Neal and Hinton, 1998; Saul and Jordan, 1995; Saul et al., 1996, and many more) are popular and successful approaches to efficiently train probabilistic generative models. In the common case when inference based on a generative data model is not tractable or not scalable to desired problem sizes, variational approaches provide approximations for an efficient optimization of model parameters. Here we focus on variational expectation maximization (variational EM) to derive learning algorithms (Jordan et al., 1999; Neal and Hinton, 1998), while acknowledging that variational approaches are also well suited and successful in the context of fully Bayesian (non-parametric) approaches (e.g. Ghahramani and Jordan, 1995; Zhou et al., 2009). Variational EM typically seeks to approximate intractable full posterior distributions by members of a specific family of variational distributions. Prominent such families are the family of Gaussian distributions (e.g., Kingma and Welling, 2014; Opper and Winther, 2005; Rezende et al., 2014) and the family of factored distributions (e.g., Jordan et al., 1999), with the latter often being referred to as *mean field* approaches. Variational approaches can, in general, be regarded as one of two large classes of approximations with the other being made up of sampling approximations. Variational training can be very efficient and has been shown to realize training of very large-scale models (Blei et al., 2003; Kingma and Welling, 2014; Rezende et al., 2014) with thousands or even millions (Sheikh and Lücke, 2016) of parameters. In terms of efficiency and scalability, they are often preferred to sampling approaches (see, e.g., Angelino et al., 2016, Section 6, for a discussion), although many successful algorithms often combine both variational and sampling techniques (e.g., Dayan et al., 1995; Kingma and Welling, 2014; Rezende et al., 2014; Shelton et al., 2011).

One drawback of variational approaches lies in their limits in modeling the posterior structure, which can result in biases introduced into the learned parameters. Gaussian variational distributions, by definition, only capture one posterior mode, for instance, and fully factored approaches (i.e., mean field) do not capture explaining-away effects including posterior correlations. Gaussian variational EM is consequently particularly popular for generative models known to result in posteriors with essentially one mode (Opper and Archambeau, 2009; Opper and Winther, 2005; Seeger, 2008). Mean field approaches are more broadly applied but the deteriorating effects of assuming a-posteriori independence have repeatedly been pointed out and discussed in different contexts (e.g., Ilin and Valpola, 2005; Mackay, 2001; Sheikh et al., 2014; Turner and Sahani, 2011; Vértes and Sahani, 2018). A further drawback of variational approaches, which they share with most sampling approaches, is the challenge of analytically deriving an appropriate approximation for any new generative model. While solutions specific to a generative model are often the best choice w.r.t. efficiency, it has been argued (see, e.g. Ranganath et al., 2014; Steinruecken et al., 2019) that the necessary expertise to derive such solutions is significantly taxing the application of variational approaches in practice.

The drawbacks of variational approaches have motivated novel research directions. Early on, mean field approaches were generalized, for instance, to contain mutual dependencies (e.g., structured mean field; Bouchard-Côté and Jordan, 2009; MacKay, 2003; Murphy, 2012; Saul and Jordan, 1995) and, more recently, methods to construct arbitrarily complex approximations to posteriors, e.g., using normalizing flows, were suggested (Kingma et al., 2016; Rezende and Mohamed, 2015). Ideally, a variational approximation should be both very efficiently scalable as well as generally applicable. Also, with variational inference becoming increasingly popular for training deep unsupervised models (e.g., Kingma et al., 2016; Rezende et al., 2014), the significance of fast and flexible variational methods has further increased. Not surprisingly, however, increasing both scalability and generality represents a major challenge because, in general, a trade-off can be observed between the flexibility of the used variational method on the one hand, and its scalability and task performance on the other.

In order to contribute to novel methods that are both flexible and scalable, we consider here generative models with discrete latents and explore the combination of variational and evolutionary optimization. For our purposes, we use truncated posteriors as a family of variational distributions. In terms of scalability, truncated posteriors suggest themselves as their application has enabled training of very large generative models (Forster and Lücke, 2018; Hirschberger et al., 2022; Sheikh and Lücke, 2016). Furthermore, the family of truncated posteriors is directly defined by the joint distribution of a given generative model, which allows for algorithms that are generically applicable to generative models with discrete latents. While truncated posteriors have been used and evaluated in a series of previous studies (e.g. Dai and Lücke, 2014; Lücke and Eggert, 2010; Shelton et al., 2017), they have previously not been optimized variationally, i.e., rather than seeking the optimal members of the variational family, truncations were estimated by one-step feed-forward functions. Instead, we use here a fully variational optimization loop which improves the variational bound by using evolutionary algorithms. For mixture models, fully variational approaches based on truncated posteriors have recently been suggested (Forster et al., 2018; Hirschberger et al., 2022), but they exploit the non-combinatorial nature of mixtures to derive speed-ups for large scales (also compare Hughes and Sudderth, 2016). Here we, for the first time, apply a fully variational approach based on truncated posteriors in order to optimize more complex generative models (see Lücke et al., 2018 and Guiraud et al., 2018 for preliminary results).

2.2 Evolutionary Variational Optimization

A probabilistic generative model stochastically generates data points (here referred to as $\vec{y}$) using a set of hidden (or latent) variables (referred to as $\vec{s}$). The generative process can be formally defined by a joint probability $p(\vec{s}, \vec{y} \mid \Theta)$, where Θ is the set of all model parameters. We will introduce concrete models in Sec. 2.3. To start, let us consider general models with binary latents. In such case $p(\vec{y} \mid \Theta) = \sum_{\vec{s}} p(\vec{s}, \vec{y} \mid \Theta)$, where the sum is taken over all possible configurations of the latent variable $\vec{s} \in \{0,1\}^H$ with H denoting the length of the latent vector. Given a set of N data points $\mathcal{Y} = \{\vec{y}^{\,(n)}\}_{n=1,\ldots,N}$, we seek parameters Θ that maximize the data likelihood $\mathcal{L}(\Theta) = \prod_{n=1}^{N} p(\vec{y}^{\,(n)} \mid \Theta)$. Here we use an approach based on Expectation Maximization (EM; e.g. Gupta and Chen, 2011, for a review). Instead of maximizing the (log-)likelihood directly, we follow, e.g., Saul and Jordan (1995) and Neal and Hinton (1998), and iteratively increase a variational lower bound (referred to as *free energy* or *ELBO*), which is given by:

$$\mathcal{F}(q, \Theta) = \sum_{n=1}^{N} \sum_{\vec{s}} q^{(n)}(\vec{s}) \, \log\big(p(\vec{y}^{\,(n)}, \vec{s} \mid \Theta)\big) \, + \sum_{n=1}^{N} H[q^{(n)}]. \tag{2.1}$$

$q^{(n)}(\vec{s})$ are variational distributions and $H[q^{(n)}] = -\sum_{\vec{s}} q^{(n)}(\vec{s}) \, \log\big(q^{(n)}(\vec{s})\big)$ denotes the entropy of these distributions. We seek distributions $q^{(n)}(\vec{s})$ that approximate the intractable posterior distributions $p(\vec{s} \mid \vec{y}^{(n)}, \Theta)$ as well as possible and that, at the same time, result in tractable parameter updates. If we denote the parameters of the variational distributions by Λ, then a variational EM algorithm consists of iteratively maximizing $\mathcal{F}(\Lambda, \Theta)$ w.r.t. Λ in the variational E-step and w.r.t. Θ in the M-step. In this respect, the M-step can maintain the same functional form as for exact EM but expectation values now have to be computed with respect to the variational distributions.

2.2.1 Evolutionary Optimization of Truncated Posteriors

Instead of using specific functional forms such as Gaussians or factored (mean field) distributions for $q^{(n)}(\vec{s})$, we choose, for our purposes, truncated posterior distributions (see, e.g., Hirschberger et al., 2022; Lücke and Eggert, 2010; Lücke and Forster, 2019; Sheikh et al., 2014; Shelton et al., 2017):

$$q^{(n)}(\vec{s} \mid \mathcal{K}^{(n)}, \hat{\Theta}) := \frac{p\left(\vec{s} \mid \vec{y}^{(n)}, \hat{\Theta}\right)}{\sum\limits_{\vec{s}\,' \in \mathcal{K}^{(n)}} p\left(\vec{s}\,' \mid \vec{y}^{(n)}, \hat{\Theta}\right)} \delta(\vec{s} \in \mathcal{K}^{(n)}) = \frac{p\left(\vec{s}, \vec{y}^{(n)} \mid \hat{\Theta}\right)}{\sum\limits_{\vec{s}\,' \in \mathcal{K}^{(n)}} p\left(\vec{s}\,', \vec{y}^{(n)} \mid \hat{\Theta}\right)} \delta(\vec{s} \in \mathcal{K}^{(n)}), \tag{2.2}$$

where $\delta(\vec{s} \in \mathcal{K}^{(n)})$ is equal to 1 if a state $\vec{s}$ is contained in the set $\mathcal{K}^{(n)}$ and 0 otherwise. The variational parameters are now given by $\Lambda = (\mathcal{K}, \hat{\Theta})$ where $\mathcal{K} = \{\mathcal{K}^{(n)}\}_{n=1,\ldots,N}$. With this choice of variational distributions, expectations w.r.t. the full posterior can be approximated by efficiently computable expectations w.r.t. truncated posteriors (2.2):

$$\mathbb{E}_{q^{(n)}}\big[g(\vec{s})\big] = \frac{\sum\limits_{\vec{s} \in \mathcal{K}^{(n)}} g(\vec{s})\; p(\vec{s}, \vec{y}^{(n)} \mid \hat{\Theta})}{\sum\limits_{\vec{s}\,' \in \mathcal{K}^{(n)}} p(\vec{s}\,', \vec{y}^{(n)} \mid \hat{\Theta})}. \tag{2.3}$$

Instead of estimating the relevant states of $\mathcal{K}^{(n)}$ using feed-forward (preselection) functions (e.g., Lücke and Eggert, 2010; Sheikh et al., 2014, 2019), we here apply a fully variational approach, i.e., we define a variational loop to optimize the variational parameters. Based on the specific functional form of truncated posteriors, optimal variational parameters $\Lambda = (\mathcal{K}, \hat{\Theta})$ are given by setting $\hat{\Theta} = \Theta$ and by seeking $\mathcal{K}$ which optimize (see Lücke, 2019, for details):

$$\mathcal{F}(\mathcal{K}^{(1\ldots N)}, \Theta) = \sum_{n=1}^{N} \log\Big(\sum_{\vec{s} \in \mathcal{K}^{(n)}} p\big(\vec{y}^{(n)}, \vec{s} \mid \Theta\big)\Big). \tag{2.4}$$

Equation (2.4) represents a reformulation of the variational lower bound (2.1) for $\hat{\Theta} = \Theta$. Because of the specific functional form of truncated posteriors, this reformulation does not contain an explicit entropy term, which allows for an efficient optimization using pairwise comparisons. More concretely, the variational bound is provably increased in the variational E-step if the sets $\mathcal{K}^{(n)}$ are updated by replacing a state $\vec{s}$ in $\mathcal{K}^{(n)}$ with a new state $\vec{s}^{\,\mathrm{new}}$ such that:

$$p(\vec{s}^{\,\mathrm{new}}, \vec{y}^{(n)} \mid \Theta) \;>\; p(\vec{s}, \vec{y}^{(n)} \mid \Theta)\,, \tag{2.5}$$

where we make sure that any new state $\vec{s}^{\,\mathrm{new}}$ is not already contained in $\mathcal{K}^{(n)}$. By successively replacing states (or bunches of states), we can keep the size of each set $\mathcal{K}^{(n)}$ constant. We use the same constant size S for all sets (i.e., $|\mathcal{K}^{(n)}| = S$ for all n), which makes S an important parameter for the approximation. The crucial remaining question is how new states can be found such that the lower bound (2.4) is increased efficiently.

A distinguishing feature when using the family of truncated posteriors as variational distributions is the type of variational parameters, which are given by sets of latent states (i.e. by the sets $\mathcal{K}^{(n)}$). As these states, for binary latents, are given by bit vectors ($\vec{s} \in \{0,1\}^H$), we can here interpret them as genomes of individuals. Evolutionary algorithms (EAs) then emerge as a very natural choice: we can use EAs to mutate and select the bit vectors $\vec{s} \in \{0,1\}^H$ of the sets $\mathcal{K}^{(n)}$ in order to maximize the lower bound (2.4). In the EA terminology, the variational parameters $\mathcal{K}^{(n)}$ then become *populations* of individuals, where each *individual* is defined by its latent state $\vec{s} \in \{0,1\}^H$ (in the following, we will use *individual* to also refer to its genome/latent state).

For each population $\mathcal{K}^{(n)}$, we will use EAs with standard genetic operators (parent selection, mutation and crossover) in order to produce offspring, and we will then use the offspring in order to improve each population. The central function for an EA is the fitness function it seeks to optimize. For our purposes, we will define the fitness $f(\vec{s}; \vec{y}^{(n)}, \Theta)$ of the individuals $\vec{s}$ to be a monotonic function of the joint $p(\vec{s}, \vec{y}^{(n)} \mid \Theta)$ of the given generative model, i.e., we define the fitness function to satisfy:

$$f(\vec{s}^{\,\mathrm{new}}; \vec{y}^{(n)}, \Theta) \;>\; f(\vec{s}; \vec{y}^{(n)}, \Theta) \quad \Leftrightarrow \quad p(\vec{s}^{\,\mathrm{new}}, \vec{y}^{(n)} \mid \Theta) \;>\; p(\vec{s}, \vec{y}^{(n)} \mid \Theta)\,. \tag{2.6}$$

A fitness satisfying Eq. (2.6) will enable the selection of parents that are likely to produce offspring with high joint probability $p(\vec{s}^{\,\mathrm{new}}, \vec{y}^{(n)} \mid \Theta)$. Before we detail how the offspring is used to improve a population $\mathcal{K}^{(n)}$, we first describe how the EA generates offspring. For each population $\mathcal{K}^{(n)}$, we set $\mathcal{K}_0^{(n)} = \mathcal{K}^{(n)}$ and then iteratively generate new generations $\mathcal{K}_g^{(n)}$ by successive application of Parent Selection, Crossover, and Mutation operators:

Parent Selection. To generate offspring, N_p parent states are first selected from the initial population $\mathcal{K}_0^{(n)}$ (see Fig. 2.1). Ideally, the parent selection procedure should be balanced between exploitation of parents with high fitness (which might be expected to be more likely to produce children with high fitness) and exploration of mutations of poor performing parents (which might be expected to eventually produce children with high fitness while increasing population diversity). Diversity is crucial, as the $\mathcal{K}^{(n)}$ are sets of unique individuals and therefore the improvement of the overall fitness of the population depends on generating as many as possible *different* children with high fitness. In our numerical experiments, we explore both random uniform selection of parents and fitness-proportional selection of parents (i.e., parents are selected with a probability proportional to their fitness). For the latter case, we define fitness here as follows:

$$f(\vec{s}; \vec{y}^{(n)}, \Theta) := \widetilde{\log} p(\vec{s}, \vec{y}^{(n)} \mid \Theta) + \mathrm{const.} \tag{2.7}$$

The term $\widetilde{\log} p(\vec{s}, \vec{y}^{(n)} \mid \Theta)$ denotes a monotonously increasing function of the joint $p(\vec{s}, \vec{y}^{(n)} \mid \Theta)$ which is more efficiently computable and which has better numerical stability than the joint itself. $\widetilde{\log} p(\vec{s}, \vec{y}^{(n)} \mid \Theta)$ is defined as the logarithm of the joint where summands that do not depend on $\vec{s}$ have been elided (see Sec. 2.S1 for examples). As the fitness (2.7) satisfies (2.6), the parents are likely to have high joint probabilities if they are selected proportionally to $f(\vec{s}; \vec{y}^{(n)}, \Theta)$. The constant term in Eq. (2.7) is introduced to ensure that the fitness function always takes positive values for fitness proportional selection. The

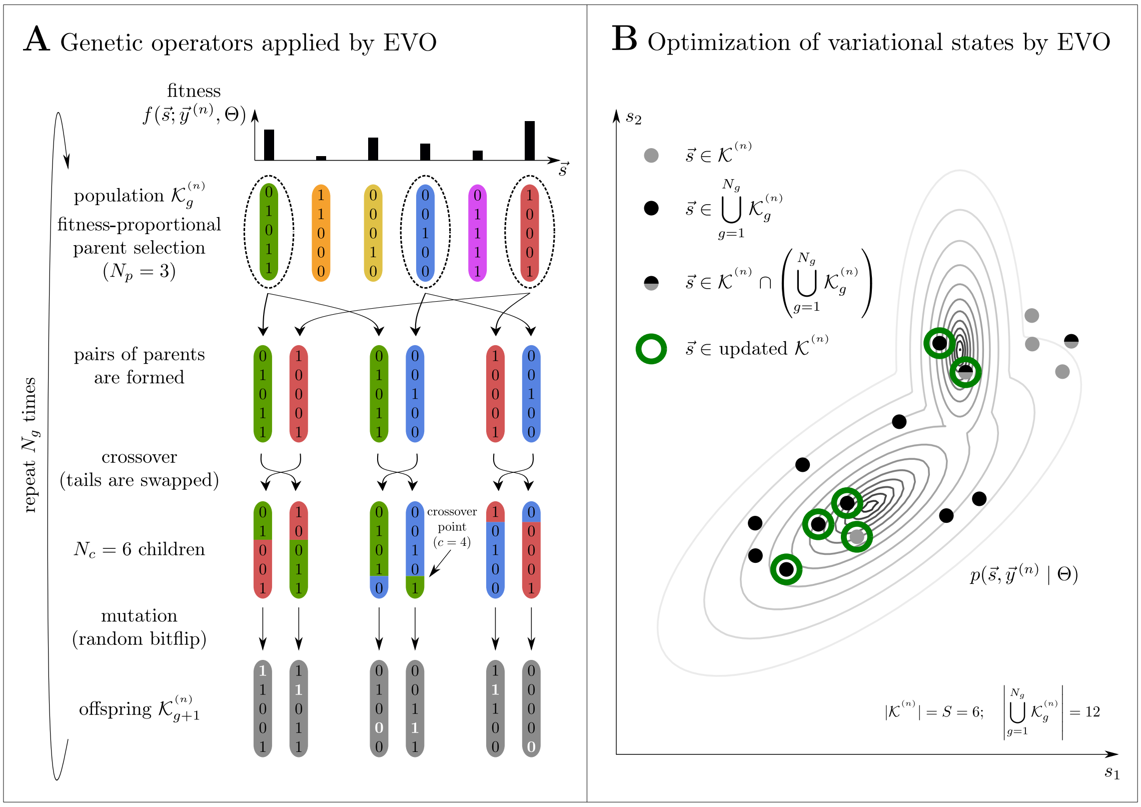

A Genetic operators applied by EVO
fitness
$f(\vec{s};\vec{y}^{(n)},\Theta)$
$\vec{s}$
population $\mathcal{K}_g^{(n)}$
fitness-proportional parent selection ($N_p = 3$)
pairs of parents are formed
crossover (tails are swapped)
crossover point ($c = 4$)
$N_c = 6$ children
mutation (random bitflip)
offspring $\mathcal{K}_{g+1}^{(n)}$
repeat N_g times
B Optimization of variational states by EVO
s_2
s_1
$\vec{s} \in \mathcal{K}^{(n)}$
$\vec{s} \in \bigcup_{g=1}^{N_g} \mathcal{K}_g^{(n)}$
$\vec{s} \in \mathcal{K}^{(n)} \cap \left(\bigcup_{g=1}^{N_g} \mathcal{K}_g^{(n)}\right)$
$\vec{s} \in$ updated $\mathcal{K}^{(n)}$
$p(\vec{s},\vec{y}^{(n)} \mid \Theta)$
$|\mathcal{K}^{(n)}| = S = 6;\quad \left|\bigcup_{g=1}^{N_g} \mathcal{K}_g^{(n)}\right| = 12$

Figure 2.1: Illustration of the variational optimization using genetic operators.

constant is defined to be const $= \left|2\min_{\vec{s}\in\mathcal{K}_g^{(n)}} \widetilde{\log p}(\vec{s},\vec{y}^{(n)} \mid \Theta)\right|$. For a given population of individuals $\mathcal{K}_g^{(n)}$, the value of const is thus constant throughout the generation of the next population $\mathcal{K}_{g+1}^{(n)}$. According to Eq. (2.7), the fitness function can be evaluated efficiently given that the joint $p(\vec{s},\vec{y}^{(n)} \mid \Theta)$ can efficiently be computed (which we will assume here).

Crossover. For the crossover step, all the parent states are paired in each possible way. Each pair is then assigned a number c from 1 to $H - 1$ with uniform probability (c defines the single crossover point). Finally, each pair swaps the last $H - c$ bits to generate offspring. The procedure generates $N_c = N_p(N_p - 1)$ children. The crossover step can be skipped, making the EA more lightweight but decreasing variety in the offspring.

Mutation. Finally, each child undergoes one random bitflip to further increase offspring diversity. In our experiments, we compare results of random uniform selection of the bits to flip with a more refined sparsity-driven bitflip algorithm (the latter scheme assigns to 0's and 1's different probabilities of being flipped in order to produce children with a sparsity compatible with the one learned by a given model; details in Sec. 2.S2). If the crossover step is skipped, the parent states are copied N_m times (N_m can be chosen between $1 \leq N_m \leq H$) and offspring is produced by mutating each copy by one random bitflip, resulting in $N_c = N_p N_m$ children.

The application of the genetic operators is iteratively repeated. The offspring $\mathcal{K}_{g+1}^{(n)}$ pro-

Algorithm 1: Evolutionary Variational Optimization (EVO).

```
define selection, crossover and mutation operators;
set hyperparameters $S$, $N_g$, etc.;
initialize model parameters $\Theta$;
for each $n$: populate set $\mathcal{K}^{(n)}$ with $S$ distinct latent states ($|\mathcal{K}^{(n)}| = S$);
repeat
    for $n = 1, \ldots, N$ do
        set $\mathcal{K}_0^{(n)} = \mathcal{K}^{(n)}$;
        for $g = 1, \ldots, N_g$ do
            $\mathcal{K}_g^{(n)} = \text{mutation}\Big(\text{crossover}\Big(\text{selection}\Big(\mathcal{K}_{g-1}^{(n)}\Big)\Big)\Big)$;
            $\mathcal{K}^{(n)} = \mathcal{K}^{(n)} \cup \mathcal{K}_g^{(n)}$;
        remove those ($|\mathcal{K}^{(n)}| - S$) elements $\vec{s}$ in $\mathcal{K}^{(n)}$ with lowest $p(\vec{s}, \vec{y}^{(n)} \mid \Theta)$;
    use M-steps with Eq. (2.3) to update $\Theta$;
until parameters $\Theta$ have sufficiently converged;
```

duced by the population $\mathcal{K}_g^{(n)}$ becomes the new population from which parents are selected (Fig. 2.1). After N_g iterations, we obtain a large number of new individuals (i.e., all the newly generated populations $\mathcal{K}_g^{(n)}$ with $g = 1, \ldots, N_g$). The new populations of individuals can now be used to improve the original population $\mathcal{K}^{(n)}$ such that a higher value for the lower bound (2.4) is obtained. To find the best new population for a given n, we collect all generated populations (all $\mathcal{K}_g^{(n)}$ with $g = 1, \ldots, N_g$) and unite them with $\mathcal{K}^{(n)}$. We then reduce this larger set back to the population size S, and we do so by only keeping those S individuals with the highest[1] joints $p(\vec{s}, \vec{y}^{(n)} \mid \Theta)$. By only keeping the states/individuals with highest joints, the update procedure for $\mathcal{K}^{(n)}$ is guaranteed to monotonically increase the variational lower bound (2.4) since the variational bound is increased if criterion (2.5) is satisfied.

Algorithm 1 summarizes the complete variational algorithm; an illustration is provided in Fig. 2.1. As the variational E-step is realized using an evolutionary algorithm, Alg. 1 will from now on be referred to as *Evolutionary Variational Optimization* (EVO). The EVO algorithm can be trivially parallelized over data-points. For later numerical experiments, we will use a parallelized implementation of EVO that can be executed on several hundreds of computing cores (for further technical details see Sec. 2.S6.1 and also compare Salimans et al., 2017)[2]. For the numerical experiments, we will indicate which parent selection procedure ("fitparents" for fitness-proportional selection, "randparents" for random uniform selection) and which bitflip algorithm ("sparseflips" or "randflips") are used in the variational loop. The term "cross" will be added to the name of the EA when crossover is employed.

---

[1] Instead of selecting the S elements with largest joints, we in practice remove the ($|\mathcal{K}^{(n)}| - S$) elements of $\mathcal{K}^{(n)}$ with the lowest joint probabilities. Both procedures are equivalent.

[2] The source code can be accessed via `https://github.com/tvlearn`.

2.2.2 Related Work on Applications of Evolutionary Approaches to Learning

Evolutionary algorithms (EAs) have repeatedly been applied to learning algorithms (e.g. Goldberg, 1989). In the context of reinforcement learning, for instance, evolution strategies have successfully been used as an optimization alternative to Q-learning and policy gradients (Salimans et al., 2017). EAs have also previously been applied to unsupervised learning based on generative models, also in conjunction with (or as substitute for) expectation maximization (EM). Myers et al. (1999) used EAs to learn the structue of Bayes Nets, for instance. Pernkopf and Bouchaffra (2005) have used EAs for clustering based on Gaussian mixture models (GMMs), where GMM parameters are updated relatively conventionally using EM; EAs are used to select the best GMM models for the clustering problem (using a minimum description length criterion). Work by Tohka et al. (2007) uses EAs for a more elaborate initialization of Finite Mixture Models, which is followed by EM. Work by Tian et al. (2011) goes further and combines EAs with a variational Bayes approach for GMMs. The algorithm they suggested is based on a fully Bayesian GMM approach with hyperpriors in the place of standard GMM parameters. The algorithm first uses an EA to find good hyperparameters; given the hyperparameters, the GMM is then trained using standard (but variational) EM iterations. The use of EAs by Pernkopf and Bouchaffra (2005) and Tian et al. (2011) is similar to their use for deep neural network (DNN) training. For DNNs, EAs are applied to optimize network hyperparameters in an outer loop (Loshchilov and Hutter, 2016; Oehmcke and Kramer, 2018; Real et al., 2017; Stanley and Miikkulainen, 2002; Suganuma et al., 2017, etc.) while DNN weight parameters are at the same time trained using standard back-propagation algorithms. Yet other approaches have applied EAs to directly optimize a clustering objective, using EAs to *replace* EM approaches for optimization (compare Hruschka et al., 2009). Similarly, in a non-probabilistic setting, EAs have been used to replace back-propagation for DNN training (see, e.g., Prellberg and Kramer, 2018, and references therein). Also in a non-probabilistic setting, for sparse coding with weights optimized based on a l_1-penalized reconstruction objective (e.g. Olshausen and Field, 1996), an EA tailored to this objective has been used to address maximum a-posteriori optimization (Ahmadi and Salari, 2016). In contrast to all such previous applications, the EVO approach (Alg. 1) applies EAs as an integral part of variational EM, i.e., EAs address the key optimization problem of variational EM in the inner variational loop.

2.2.3 Data Estimator

As part of the numerical experiments, we will apply EVO to fit generative models to corrupted data which we will aim to restore; for instance, we will apply EVO to denoise images corrupted by additive noise or to restore images with missing data (see Sec. 2.4). Here we illustrate how an estimator for such data reconstruction can be derived (details are given in Sec. 2.S3). Generally speaking, our aim is to compute estimates $\vec{y}^{\,\mathrm{est}}$ based on observations $\vec{y}^{\,\mathrm{obs}}$. If data points are corrupted by noise but not by missing values, we define the observations $\vec{y}^{\,\mathrm{obs}}$ to be equal to the data points $\vec{y} \in \mathbb{R}^D$ and compute estimates y_d^{est} for every $d = 1, \ldots, D$ with D denoting the dimensionality of the data. In the presence of missing values, we define data points to comprise an observed part $\vec{y}^{\,\mathrm{obs}}$ and a missing part $\vec{y}^{\,\mathrm{miss}} = \vec{y} \backslash \vec{y}^{\,\mathrm{obs}}$, and we aim to compute estimates $\vec{y}^{\,\mathrm{est}}$ for the missing part. A data estimator can be derived based on the posterior predictive distribution $p(\vec{y}^{\,\mathrm{est}} \mid \vec{y}^{\,\mathrm{obs}})$. Here we will sketch how to derive an estimator for general models with binary latents and continuous observables (in Sec. 2.S3 we provide a detailed derivation and show how to extend it to models with binary-continuous latents). The posterior predictive distribution for a model with binary latents and continuous observables is given by:

$$p(\vec{y}^{\,\mathrm{est}} \mid \vec{y}^{\,\mathrm{obs}}, \Theta) = \sum_{\vec{s}} p(\vec{y}^{\,\mathrm{est}} \,|\, \vec{s}, \Theta)\; p(\vec{s} \mid \vec{y}^{\,\mathrm{obs}}, \Theta). \tag{2.8}$$

We choose to take expectations w.r.t. the posterior predictive distribution, which leads to the following estimator:

$$\mathbb{E}_{p(\vec{y}^{\,\mathrm{est}} \,|\, \vec{y}^{\,\mathrm{obs}}, \Theta)}\left[y_d^{\mathrm{est}}\right] = \mathbb{E}_{p(\vec{s} \,|\, \vec{y}^{\,\mathrm{obs}}, \Theta)}\left[\mathbb{E}_{p(y_d^{\mathrm{est}} \,|\, \vec{s}, \Theta)}\left[y_d^{\mathrm{est}}\right]\right]. \tag{2.9}$$

The inner expectation in Eq. (2.9) can be identified with the mean of the distribution of the observables given the latents. This distribution is part of the definition of the generative model (compare Sec. 2.3.1). The outer expectation in Eq. (2.9) is taken w.r.t. the posterior distribution of the latents. Such expectation can efficiently be approximated using truncated expectations (2.3). Details and examples of data estimators for concrete models are given in Sec. 2.S3.

2.3 Application to Generative Models, Verification and Scalability

Considering Alg. 1, observe that probabilistic inference for the algorithm is fully defined by the joint $p(\vec{s}, \vec{y}^{\,(n)} \,|\, \Theta)$ of a given generative model and by a set of hyperparameters for the optimization procedure. No additional and model specific derivations are required for the variational E-step, which suggests Alg. 1 for 'black box' inference for models with binary latents. Using three different generative models, we here (A) verify this 'black box' applicability, (B) numerically evaluate the algorithm's ability to recover generating parameters, and (C) show scalability of the approach by applying it to large-scale generative models. To be able to study the properties of the novel variational optimization procedure, we consider generative models for which parameter update equations (M-steps) have been derived previously.

2.3.1 Used Generative Models

The generative models we use are Noisy-OR Bayes Nets (binary latents and binary observables), Binary Sparse Coding (binary latents and continuous observables), and Spike-and-Slab Sparse Coding (binary-continuous latents and continuous observables).

Noisy-OR Bayes Nets. A Noisy-OR Bayes Net (NOR; e.g., Jernite et al., 2013; Rotmensch et al., 2017; Šingliar and Hauskrecht, 2006) is a non-linear bipartite data model with all-to-all connectivity between the layer of hidden and observed variables (and no intra-layer connections). Latents and observables both take binary values, i.e., $\vec{s} \in \{0, 1\}^H$ and $\vec{y} \in \{0, 1\}^D$ with H and D denoting the dimensions of the latent and observed spaces.

The model assumes a Bernoulli prior for the latents, and non-zero latents are then combined via the noisy-OR rule:

$$p(\vec{s} \mid \Theta) = \prod_{h=1}^{H} \mathcal{B}(s_h; \pi_h), \qquad p(\vec{y} \mid \vec{s}, \Theta) = \prod_{d=1}^{D} N_d(\vec{s})^{y_d}(1 - N_d(\vec{s}))^{1-y_d}, \tag{2.10}$$

where $\mathcal{B}(s_h; \pi_h) = \pi_h^{s_h}(1 - \pi_h)^{1-s_h}$ and where $N_d(\vec{s}) := 1 - \prod_{h=1}^{H}(1 - W_{dh}s_h)$. In the context of the NOR model, $\Theta = \{\vec{\pi}, W\}$, where $\vec{\pi}$ is the set of values $\pi_h \in [0, 1]$ representing the prior activation probabilities for the hidden variables s_h, and W is a $D{\times}H$ matrix of values $W_{dh} \in [0, 1]$ representing the probability that an active latent variable s_h activates the observable y_d. M-step update rules for the NOR model can be derived by inserting Eq. (2.10) into the lower bound (2.1) and by then taking derivatives of the resulting expression w.r.t. all model parameters. The update rule for the W parameter does not allow for a closed-form solution, and a fixed-point equation is employed instead. The resulting M-step equations are shown in Sec. 2.S4.

Binary Sparse Coding. Binary Sparse Coding (BSC; Haft et al., 2004; Henniges et al., 2010) is an elementary sparse coding model for continuous data with binary latent and continuous observed variables. Similar to standard sparse coding (Lee et al., 2007; Olshausen and Field, 1997), BSC assumes that, given the latents, the observables follow a Gaussian distribution. BSC and standard sparse coding differ from each other in their latent variables. In standard sparse coding, latents take continuous values and are typically modeled, e.g., with Laplace or Cauchy distributions. BSC uses binary latents $\vec{s} \in \{0, 1\}^H$ which are assumed to follow a Bernoulli distribution $\mathcal{B}(s_h; \pi)$ with the same activation probability π for each hidden unit s_h. The combination of the latents is described by a linear superposition rule. Given the latents, the observables $\vec{y} \in \mathbb{R}^D$ are independently and identically drawn from a Gaussian distribution:

$$p(\vec{s} \mid \Theta) = \prod_{h=1}^{H} \mathcal{B}(s_h; \pi), \qquad p(\vec{y} \mid \vec{s}, \Theta) = \prod_{d=1}^{D} \mathcal{N}(y_d; \sum_{h=1}^{H} W_{dh}s_h, \sigma^2). \tag{2.11}$$

The parameters of the BSC model are $\Theta = (\pi, \sigma, W)$. W is a $D \times H$ matrix whose entries W_{dh} contain the weights associated with each hidden unit s_h and obervable y_d; σ determines the standard deviation of the Gaussian. The M-step equations for BSC can be derived analogously to the NOR model by inserting Eq. (2.11) into Eq. (2.1) and by then optimizing the resulting expression w.r.t. each model parameter (compare, e.g., Henniges et al., 2010). As opposed to NOR, each of the BSC update equations can be derived in closed-form. The explicit expressions are listed in Sec. 2.S4.

Spike-and-Slab Sparse Coding. As a final example, we consider a more expressive data model and use EVO to optimize the parameters of a spike-and-slab sparse coding (SSSC) model. SSSC extends the generative model of BSC in the sense that it uses a spike-and-slab instead of a Bernoulli prior distribution. The SSSC model has been used in a number of previous studies and in a number of variants (Goodfellow et al., 2012; Lücke and Sheikh, 2012; Sheikh et al., 2014; Titsias and Lázaro-Gredilla, 2011, and many more). It can be formulated using two sets of latents, $\vec{s} \in \{0, 1\}^H$ and $\vec{z} \in \mathbb{R}^H$, which are combined via pointwise multiplication s.t. $(\vec{s} \odot \vec{z})_h = s_h z_h$. Here we take the continuous latents to be distributed according to a multivariate Gaussian with mean $\vec{\mu}$ and a full covariance matrix Ψ. The binary latents follow a Bernoulli distribution with individual activation probabilities π_h, $h = 1, \dots, H$:

$$p\,(\vec{s} \mid \Theta) = \prod_{h=1}^{H} \mathcal{B}(s_h; \pi_h), \qquad p\,(\vec{z} \mid \Theta) = \mathcal{N}(\vec{z}; \vec{\mu}, \Psi). \tag{2.12}$$

As in standard sparse coding, SSSC assumes components to combine linearly. The linear combination then determines the mean of a univariate Gaussian for the observables:

$$p\,(\vec{y} \mid \vec{s}, \vec{z}, \Theta) = \mathcal{N}(\vec{y}; \sum_{h=1}^{H} \vec{W}_h s_h z_h, \sigma^2 \mathbb{1}) \tag{2.13}$$

where W is a $D \times H$ matrix with columns $\vec{W}_h$ and where $\mathbb{1}$ is the unit matrix. The M-step update rules of the SSSC model can be derived by taking derivatives of the free energy

$$\mathcal{F}(q, \Theta) = \sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\big[\log(p(\vec{y}^{\,(n)}, \vec{s}, \vec{z} \mid \Theta))\big] + \sum_{n=1}^{N} H[q^{(n)}] \tag{2.14}$$

w.r.t. each of the model parameters $\Theta = (\vec{\pi}, \sigma, W, \vec{\mu}, \Psi)$. The term $\mathbb{E}_{q^{(n)}}\big[f(\vec{s}, \vec{z})\big]$ in Eq. (2.14) denotes the expectation of $f(\vec{s}, \vec{z})$ w.r.t. a variational distribution $q^{(n)}(\vec{s}, \vec{z})$:

$$\mathbb{E}_{q^{(n)}}\big[f(\vec{s}, \vec{z})\big] = \sum_{\vec{s}} \int q^{(n)}(\vec{s}, \vec{z}) f(\vec{s}, \vec{z})\, \mathrm{d}\vec{z}. \tag{2.15}$$

The term $H[q^{(n)}]$ in Eq. (2.14) denotes the entropy $-\mathbb{E}_{q^{(n)}}\big[\log(q^{(n)}(\vec{s}, \vec{z}))\big]$ of the variational distributions. For a detailed derivation of the SSSC M-step equations see Sheikh et al. (2014) and compare Goodfellow et al. (2012). For SSSC, all M-step updates (A) have closed-form solutions and (B) contain as arguments expectation values (2.15). Importantly for applying EVO, all these expectation values can be reformulated as expectations w.r.t. the posterior over the binary latent space. For more details see Sec. 2.S4 and compare Sheikh et al. (2014). Based on the reformulations, the expectations w.r.t. the posterior over the binary latent space can be approximated using the truncated expectations (2.3). Since the joint $p(\vec{y}, \vec{s} \mid \Theta)$ of the SSSC model is computationally tractable and given by

$$p(\vec{y}, \vec{s} \mid \Theta) = \mathcal{B}(\vec{s}; \vec{\pi})\; \mathcal{N}(\vec{y};\, \tilde{W}_{\vec{s}}\, \vec{\mu}, C_{\vec{s}}), \tag{2.16}$$

with $C_{\vec{s}} = \sigma^2 \mathbb{1} + \tilde{W}_{\vec{s}}\, \Psi\, \tilde{W}_{\vec{s}}^{\mathrm{T}}$, $(\tilde{W}_{\vec{s}})_{dh} = W_{dh} s_h$ s.t. $W\,(\vec{s} \odot \vec{z}) = \tilde{W}_{\vec{s}}\, \vec{z}$, the variational lower bound (2.4) can be efficiently optimized using Alg. 1. The EVO algorithm consequently provides a novel, evolutionary approach to efficiently optimize the parameters of the SSSC data model.

2.3.2 Verification: Recovery of Generating Parameters

First, we verified EVO by training NOR, BSC and SSSC models on artificial data for which the ground-truth generative parameters were known. We used the bars test as a standard setup for such purposes (Földiák, 1990; Hoyer, 2003; Lücke and Sahani, 2008). We started with a standard bars test using a weight matrix W^{gen} whose $H^{\mathrm{gen}} = 2\sqrt{D}$ columns represented generative fields in the form of horizontal and vertical bars. We considered a dictionary of $H^{\mathrm{gen}} = 10$ (Lücke and Eggert, 2010; Lücke and Sahani, 2008; Lücke et al.,

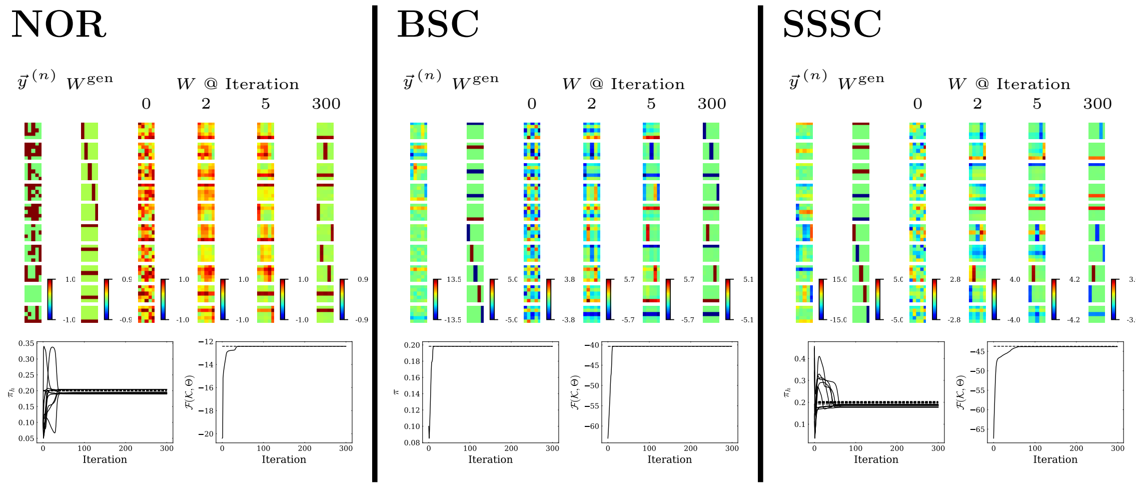


Figure 2.2: Recovering ground-truth generative parameters of NOR, BSC and SSSC models on artificial data using EVO (see text for details).

2018; Sheikh and Lücke, 2016) components. For NOR, we set the amplitudes of the bars and of the background to a value of 0.8 and 0.1, respectively; for BSC and SSSC, the bar amplitudes were uniformly randomly set to 5 and -5, and the background was set to 0. For BSC and SSSC, we chose $(\sigma^{\mathrm{gen}})^2 = 1$ as ground-truth variance parameter; for SSSC, we furthermore defined $\vec{\mu}^{\mathrm{gen}} = \vec{0}$ and $\Psi^{\mathrm{gen}} = \mathbb{1}$.

For each model, 30 data sets of $N = 5{,}000$ samples were generated. We used $\pi_h^{\mathrm{gen}} = \frac{2}{H^{\mathrm{gen}}}\,\forall\, h = 1, \ldots, H^{\mathrm{gen}}$ (see Fig. 2.2 for a few examples). We then applied EVO to train NOR, BSC and SSSC models with $H = H^{\mathrm{gen}} = 10$ components (EVO hyperparameters are listed in Tab. 2.S2 in Sec. 2.S6.2). For NOR, priors π_h were initialized to an arbitrary sparse value (typically $\frac{1}{H}$) while the initial weights W^{init} were sampled from the standard uniform distribution. The $\mathcal{K}^{(n)}$ sets were initialized by sampling bits from a Bernoulli distribution that encouraged sparsity. In our experiments, the mean of the distribution was set to $\frac{1}{H}$. For BSC, the initial value of the prior was defined $\pi^{\mathrm{init}} = \frac{1}{H}$; the value of $(\sigma^{\mathrm{init}})^2$ was set to the variance of the data points averaged over the observed dimensions. The columns of the W^{init} matrix were initialized with the mean of the data points to which some Gaussian noise with a standard deviation of $0.25\,\sigma^{\mathrm{init}}$ was added. The initial values of the $\mathcal{K}^{(n)}$ sets were drawn from a Bernoulli distribution with $p(s_h = 1) = \frac{1}{H}$. To initialize the SSSC model, we uniformly randomly drew $\vec{\pi}^{\mathrm{init}}$ and $\vec{\mu}^{\mathrm{init}}$ from the interval $[0.1, 0.5]$ and $[1, 5]$, respectively (compare Sheikh et al., 2014) and set Ψ^{init} to the unit matrix. For SSSC, we proceeded as we had done for the BSC model to initialize the dictionary W, the variance parameter σ^2 and the sets $\mathcal{K}^{(n)}$. The evolution of the model parameters during learning is illustrated in Fig. 2.2 for an exemplary run of the “fitparents-randflip” EA (illustrations of the parameters σ of the BSC model and Ψ, $\vec{\mu}$ and σ of the SSSC model are depicted in Fig. 2.S1 in Sec. 2.S6).

Fig. 2.2 illustrates that the ground-truth generative parameters can accurately be recovered for each model using EVO. For all models, the bar-like structure of the ground-truth generative fields (GFs) is correctly captured (in permuted order) in the learned GFs. For the SSSC model, the GFs, latent means and latent covariances are recovered in scaled versions compared to the respective ground-truth values. This effect can be considered as a consequence of the spike-and-slab prior which allows for multiple equivalent solutions for the amplitudes of W and $\vec{z}$ given a data point. We further used the setup with artificial data and known ground-truth generative parameters to explore EAs with different types of combinations of genetic operators. Different combinations of genetic operators resulted in different degrees of ability to recover the ground-truth parameters. For instance, for NOR trained with EVO, the combination of random uniform parent selection, single-point crossover and sparsity-driven bitflip ("randparents-cross-sparseflips") resulted in the best behavior for bars test data (also compare Guiraud et al., 2018). For BSC and SSSC trained with EVO, the best operator combinations were "fitparents-cross-sparseflips" and "fitparents-randflips", respectively (compare Fig. 2.S2 in Sec. 2.S6). In general, the best combination of operators will depend on the data model and the data set used.

2.3.3 Scalability to Large Generative Models

Having verified the ability of EVO to recover ground-truth generating parameters for the three generative models of Sec. 2.3.1, we continued with investigating how well the evolutionary approximation was able to optimize large-scale models. Scalability is a property of crucial importance for any approximation method, and large-scale applicability is essential to accomplish many important data processing tasks based on generative models. To investigate scalability, we used natural image patches which are known to be richly structured and which require large models to appropriately model their distribution. We used image patches extracted from a standard image database (van Hateren and van der Schaaf, 1998). We trained the NOR, BSC and SSSC data models using the "fitparents-cross-sparseflips" variant of EVO as we observed this operator combination to perform well across all models for this data.

For NOR, we considered raw image patches, i.e., images which were not mean-free or whitened and reflected light intensities relatively directly. We used image patches that were generated by extracting random 10×10 subsections of a single 255×255 image of overlapping grass wires (part of image 2338 of the database). We clamped the brightest $1\,\%$ pixels from the data set, and scaled each patch to have gray-scale values in the range $[0, 1]$. From these patches, we then created $N = 30{,}000$ data points $\vec{y}^{(n)}$ with binary entries by repeatedly choosing a random gray-scale image and sampling binary pixels from a Bernoulli distribution with parameter equal to the pixel values of the patches with $[0, 1]$ gray-scale. Because of mutually occluding grass wires in the original image, the generating components of the binary data could be expected to follow a non-linear superimposition, which motivated the application of a non-linear generative model such as NOR. For the data with $D = 10 \times 10 = 100$ observables, we applied a NOR model with $H = 100$ latents (EVO hyperparameters are listed in Tab. 2.S2 in Sec. 2.S6.2). During training, we clamped the priors π_h to a minimum value of $1/H$ to encourage the model to make use of all generative fields. Fig. 2.3 (top) shows a random selection of 50 generative fields learned by the NOR model (the full dictionary is displayed in Fig. 2.S3 in Sec. 2.S6). Model parameter initialization followed the same procedure as for the bars tests. As can be observed, EVO is efficient for large-scale NOR on real data. Many of the generative fields of Fig. 2.3 (top) resemble curved edges, which is in line with expectations and with results, e.g., as obtained by Lücke and Sahani (2008) with another non-linear generative model.

For the BSC generative model, we are not restricted to positive data. Therefore, we can use a whitening procedure as is customary for sparse coding approaches (Olshausen and

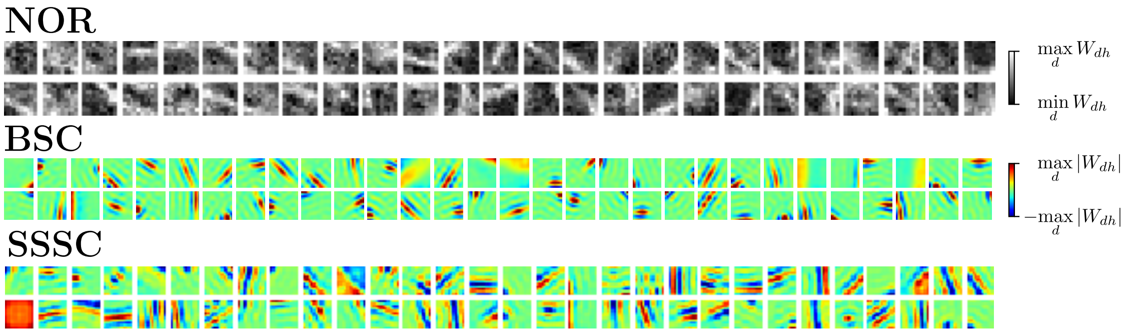


Figure 2.3: Dictionary elements learned by NOR (top), BSC (middle) and SSSC (bottom) models on natural image patches. The models were trained on separate data sets that had been obtained using different preprocessing schemes (see text). For NOR, a random selection of 50 out of 100 fields is displayed; for BSC the fields corresponding to the 60 out of 300 most active hidden units are shown; for SSSC a random selection of 60 out of 512 fields is displayed. The full dictionaries learned by the individual models are depicted in Fig. 2.S3 in Sec. 2.S6.

Field, 1997). We employed $N = 100{,}000$ image patches of size $D = 16 \times 16 = 256$ which were randomly picked from the van Hateren data set. The highest 2 % of the pixel values were clamped to compensate for light reflections, and patches without significant structure were removed to prevent amplifications of very small intensity differences. The whitening procedure we subsequently applied is based on ZCA whitening (Bell and Sejnowski, 1997) retaining 95 % of the variance (compare Exarchakis and Lücke, 2017). We then applied EVO to fit a BSC model with $H = 300$ components to the data (EVO hyperparameters are listed in Tab. 2.S2 in Sec. 2.S6.2). To initialize the model and the variational parameters, we proceeded as we had done for the bars test (Sec. 2.3.2). Fig. 2.3 (middle) depicts some of the generative fields learned by the BSC model. These fields correspond to the 60 hidden units with highest prior activation probability (see Fig. 2.S3 in Sec. 2.S6 for the full dictionary). The obtained generative fields primarily take the form of the well-known Gabor functions with different locations, orientations, phase, and spatial frequencies.

We collected another $N = 100{,}000$ image patches of size $D = 12 \times 12 = 144$ from the van Hateren data set to fit a SSSC model with $H = 512$ components (EVO hyperparameters listed in Tab. 2.S2 in Sec. 2.S6.2). We applied the same preprocessing as for the BSC model and initialized the model and the variational parameters similarly to the bars test. Fig. 2.3 (bottom) shows a random selection of 60 out of the 512 generative fields learned by the SSSC model (the full dictionary is displayed in Fig. 2.S3 in Sec. 2.S6). The experiment shows that EVO can be applied to train complex generative models on large-scale realistic data sets; the structures of the learned generative fields (a variety of Gabor-like and a few globular fields) are in line with observations made in previous studies which applied SSSC models to whitened image patches (see, e.g., Sheikh and Lücke, 2016).

2.4 Performance Comparison on Denoising and Inpainting Benchmarks

So far, we have verified that EVO can be applied to the training of elementary generative models such as Noisy-OR Bayes Nets (NOR) and Binary Sparse Coding (BSC) but also to the more expressive model of spike-and-slab sparse coding (SSSC) which features a more flexible prior than BSC or standard sparse coding. To perform variational optimization for these models, no additional analytical steps were required and standard settings of optimization parameters were observed to work well. However, an important question is how well EVO is able to optimize model parameters compared to other methods. And more generally, how well do generative model algorithms perform in benchmarks if their parameters are optimized by EVO? Here we will address these questions using two standard and popular benchmarks: image denoising and image inpainting. These two benchmarks offer themselves for performance evaluation as benchmark data is available for a large range of different algorithms (including algorithms based on generative models among many others). The data for standard denoising and inpainting benchmarks is continuous. We will consequently focus on the BSC and SSSC generative models. In the following, we will refer to EVO applied to BSC as *Evolutionary Binary Sparse Coding* (EBSC) and to EVO applied to SSSC as *Evolutionary Spike-and-Slab Sparse Coding* (ES3C). As genetic operator combination for the EA, we will use “fitparents-randflips” for all subsequent experiments. The main reason to prefer this combination over combinations that also use crossover is computational efficiency (which is a major limiting factor for the following experiments).

2.4.1 Algorithm Types and Comparison Categories

We will use standard benchmarks to compare EBSC and ES3C w.r.t. (A) algorithms based on generative models equal or similar to the ones used in this work, (B) other probabilistic methods including mixture models and Markov random fields approaches, (C) non-local image specific methods and (D) deep neural networks. Tab. 2.1 gives an overview of the methods we compare to, sorted by algorithm type. To the knowledge of the authors, the listed papers represent the best performing approaches while we additionally included standard baselines such as BM3D, EPLL or MLP (see Tab. 2.1 for references).

The algorithms listed in Tab. 2.1 have different requirements that have to be fulfilled for them to be applicable. To facilitate comparison, we have grouped the benchmarked algorithms based on the prior knowledge they require/use (see Tab. 2.2 and Sec. 2.S5 for a related discussion). As can be observed, some algorithms (e.g., KSVD or BM3D) require the noise level to be known a-priori, for instance. Others require (often large amounts of) clean images for training, which typically applies for feed-forward deep neural networks (DNNs). If an algorithm is able to learn without information other than the corrupted image itself, it is commonly referred to as a ‘zero-shot’ learning algorithm (compare, e.g., Imamura et al., 2019; Shocher et al., 2018). Algorithms of the ‘zero-shot’ category we compare to are, e.g., MTMKL, BPFA, DIP and GSVAE. The EVO-based algorithms EBSC and ES3C also fall into the ‘zero-shot’ category. In Tab. 2.2, we have labeled the categories DE1-DE6 for denoising and IN1-IN6 for inpainting. DE2 could also be referred to as ‘zero-shot + noise level’ as the algorithms of that category also do not need additional images.

Acronym	Algorithm Name / Type	Reference
	Sparse Coding / Dictionary Learning	
MTMKL	Spike-and-Slab Multi-Task and Multiple Kernel Learning	Titsias and Lázaro-Gredilla (2011)
BPFA	Beta Process Factor Analysis	Zhou et al. (2012)
GSC	Gaussian Sparse Coding	Sheikh et al. (2014)
KSVD	Sparse and Redundant Representations Over Trained Dictionaries	Elad and Aharon (2006)
cKSVD	Sparse Representation for Color Image Restoration	Mairal et al. (2008)
LSSC	Learned Simultaneous Sparse Coding	Mairal et al. (2009)
BKSVD	Bayesian K-SVD	Serra et al. (2017)
	Other Probabilistic Methods (GMMs, MRFs)	
EPLL	Expected Patch Log Likelihood	Zoran and Weiss (2011)
PLE	Piecewise Linear Estimators	Yu et al. (2012)
FoE	Fields of Experts	Roth and Black (2009)
MRF	Markov Random Fields	Schmidt et al. (2010)
NLRMRF	Non-Local Range Markov Random Field	Sun and Tappen (2011)
	Non-Local Image Processing Methods	
BM3D	Block-Matching and 3D Filtering	Dabov et al. (2007)
NL	Patch-based Non-Local Image Interpolation	Li (2008)
WNNM	Weighted Nuclear Norm Minimization	Gu et al. (2014)
	Deep Neural Networks	
MLP	Multi Layer Perceptron	Burger et al. (2012)
DnCNN-B	Denoising Convolutional Neural Network	Zhang et al. (2017)
TNRD	Trainable Nonlinear Reaction Diffusion	Chen and Pock (2017)
MemNet	Deep Persistent Memory Network	Tai et al. (2017)
IRCNN	Image Restoration Convolutional Neural Network	Chaudhury and Roy (2017)
GSVAE	Gumbel-Softmax Variational Autoencoder	Jang et al. (2017)
FFDNet	Fast and Flexible Denoising Convolutional Neural Network	Zhang et al. (2018)
DIP	Deep Image Prior	Ulyanov et al. (2018)
DPDNN	Denoising Prior Driven Deep Neural Network	Dong et al. (2019)
BDGAN	Image Blind Denoising Using Generative Adversarial Networks	Zhu et al. (2019)
BRDNet	Batch-Renormalization Denoising Network	Tian et al. (2020)

Table 2.1: Algorithms used for comparsion on denoising and/or inpainting benchmarks. The algorithms have been grouped into four types of approaches. The table includes the best performing algorithms and standard baselines.

A Denoising

		Further a-priori assumptions		
		None	Noise Level	Test-Train Match
Clean training data required/used	No	DE1 MTMKL BPFA GSC GSVAE BKSVD EBSC ES3C	DE2 KSVD LSSC BM3D WNNM	DE3 n/a
	Yes	DE4 n/a	DE5 EPLL	DE6 MLP DnCNN-B TNRD MemNet FFDNet DPDNN BDGAN BRDNet

B Inpainting

		Further a-priori assumptions		
		None	Noise Level	Test-Train Match
Clean training data required/used	No	IN1 MTMKL BPFA BKSVD DIP ES3C	IN2 NL* PLE	IN3 n/a
	Yes	IN4 FoE MRF	IN5 cKSVD	IN6 NLRMRF IRCNN

Table 2.2: Algorithms for denoising and inpainting use different degrees and types of prior knowledge. In panel A, the algorithms listed in the column “Noise Level” assume access to the ground-truth noise level of the test data. The approaches listed in the column “Test-Train Match” in panel A are optimized either for one single or for several particular noise levels. In panel B, the algorithms listed in the column "Noise Level" (NL, PLE, cKSVD) use heuristics for noise level and/or sparsity estimation (*NL internally makes use of BM3D which requires a noise level (which is in turn set by a patch-by-patch heuristics); PLE and cKSVD employ manually set noise level/sparsity estimates). The column “Test-Train Match” in panel B lists algorithms that are optimized for a particular percentage of missing pixels (e.g. for images with 80% randomly missing values) or for a particular missing value pattern. The algorithms of category IN6 can be applied without providing a mask that indicates the pixels that are to be filled (see Sec. 2.S5 for details).

2.4.2 Image Denoising

To apply EBSC and ES3C for image denoising, we first preprocessed the noisy images that we aimed to restore by cutting the images into smaller patches: given an image of size $\mathcal{H}\times\mathcal{W}$ and using a patch size of $D=\mathcal{P}_{\mathrm{x}}\times\mathcal{P}_{\mathrm{y}}$, we cut all possible $N=(\mathcal{W}-\mathcal{P}_{\mathrm{x}}+1)\times(\mathcal{H}-\mathcal{P}_{\mathrm{y}}+1)$ patches by moving a sliding window over the noisy image. Patches were collected individually for each image and corresponded to the data sets $\mathcal{Y}$ which EBSC or ES3C were trained on. In other words, EBSC and ES3C leveraged nothing except of the information of the noisy image itself (no training on other images, no training on clean images, noise unknown a-priori). After model parameters had been inferred using EBSC and ES3C, we took expectations w.r.t. the posterior predictive distribution of the model in order to estimate the non-noisy image pixels. For each noisy patch, we estimated all its non-noisy pixel values, and, as commonly done, we repeatedly estimated the same pixel value based on different (mutually overlapping) patches that shared the same pixel. The different estimates for the same pixel were then gathered to determine the non-noisy value using a weighted average (for details see Sec. 2.2.3 and Sec. 2.S3; also see, e.g., Burger et al., 2012). Finally, we were interested in how well EBSC and ES3C could denoise a given image in terms of the standard peak-signal-to-noise ratio (PSNR) evaluation measure.

2.4.2.1 Relation Between PSNR Measure and Variational Lower Bound

We first investigated whether the value of the learning objective (the variational lower bound) of EBSC and ES3C was instructive about the denoising performance of the algorithms in terms of PSNR. While the PSNR measure requires access to the clean target image, the learning objective can be evaluated without such ground-truth knowledge. If learning objective and PSNR measure were reasonably well correlated, one could execute the algorithms several times on a given image and determine the best run based on the highest value of the learning objective without requiring access to the clean image. For the experiment, we used the standard “House” image degraded by additive white Gaussian (AWG) noise with a standard deviation of $\sigma=50$ (the clean and noisy images are depicted in Fig. 2.5). We applied ES3C for denoising using patches of size $D=8\times8$ and a dictionary with $H=256$ elements (EVO hyperparameters listed in Tab. 2.S2 in Sec. 2.S6.2). At several iterations during execution of the algorithm, we measured the value of the learning objective and the PSNR of the reconstructed image. The experiment was repeated five times, each time using the same noisy image but a different initialization of the algorithm. Fig. 2.4 illustrates the result of this experiment. As can be observed, PSNR values increase with increasing value of the objective $\mathcal{F}(\mathcal{K},\Theta)\,/\,N$ during learning. The truncated distributions which give rise to the lower bound are consequently well suited. Only at the end of learning, PSNR values slightly decrease while the lower bound still increases (Fig. 2.4, inset). The final PSNR decreases are likely due to the PSNR measure being not perfectly correlated with the data likelihood. However, decreases of the likelihood while the lower bound still increases cannot be excluded. While the final decreases of PSNR are small (see scale of inset figure of Fig. 2.4), their consequence is that the run with the highest final value for the variational bound is not necessarily the run with the highest PSNR. For Fig. 2.4, for instance, Run 1 has the highest final variational bound but Run 4 the highest final PSNR.

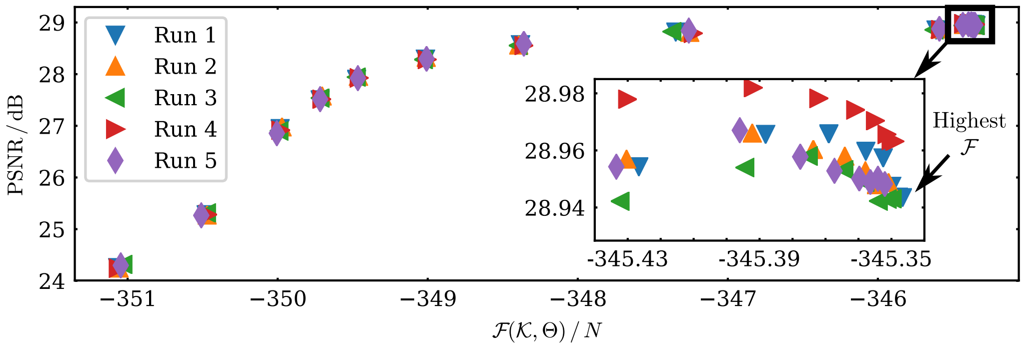


Figure 2.4: Variational bound and PSNR value pairs obtained from applying ES3C to the noisy “House” image (AWG noise with $\sigma = 50$). 2,000 EVO iterations were performed and PSNR values were measured at iterations 1, 2, 5, 10, 20, 50, 100, 200, 400, 600, 800, 1,000, 1,200, 1,400, 1,600, 1,800, 2,000. The experiment was repeated five times using the same noise realization in each run. The PSNR values at the last iteration were 28.94, 28.95, 28.94, 28.96, 28.95 (avg. rounded 28.95 ± 0.01; all values in dB).

2.4.2.2 Comparison of Generative Models and Approximate Inference

Before comparing EBSC and ES3C with a broad range of different denoising algorithms (further below), we first focused on approaches that are all based on essentially the same data model. Differences in measured performance can then be attributed more directly to differences in the method used for parameter optimization. Concretely, we first considered algorithms that are all based on a spike-and-slab sparse coding (SSSC) model. Namely, we compared the following algorithms:

- the ES3C algorithm which is trained using variational optimization of truncated posteriors (EVO),
- the MTMKL algorithm (Titsias and Lázaro-Gredilla, 2011) which uses a factored variation approach (i.e., mean field) for parameter optimization (also compare Goodfellow et al., 2012),
- the BPFA algorithm (Zhou et al., 2012) which uses sampling for parameter optimization,
- the GSC algorithm (Sheikh et al., 2014) which uses truncated posteriors constructed using preselection (i.e., no variational loop).

In addition, we included EBSC into the comparison. EBSC is based on the BSC generative model, and the BSC generative model can be optimized using different optimization approaches. Goodfellow et al. (2012), for instance, discuss the BSC model as a boundary case of their mean field optimization of the SSSC generative model. Furthermore, BSC can be optimized using encoding neural networks in conjunction with reparameterization and the Gumbel-Softmax trick (Jang et al., 2017). Using the Gumbel-Softmax trick, standard optimization techniques developed for VAEs can be maintained also if discrete latent variables are used[3]. Optimization using gradient ascent via amortized inference, reparameterization and Gumbel-Softmax trick can consequently also be used to optimize the BSC data model. We included such a Gumbel-Softmax optimized BSC model for comparison, and refer to it as $\mathrm{GSVAE}^{\mathrm{lin}}$ (for *Gumbel-Softmax Variational Autoencoder*). More concretely, we used the VAE setup suggested by Jang et al. (2017) but used a linear decoder to match the linear generative model of BSC (the superscript “lin” indicates the linearity); to match the binary latents of BSC, we used latents with only two categories and ensured that the category corresponding to “0” did not contribute to data generation. Our implementation of $\mathrm{GSVAE}^{\mathrm{lin}}$ is based on the source code provided by the original publication (Jang, 2016), and we conducted further control experiments also using GSVAE versions more closely aligned with the original VAE architectures (see Sec. 2.S6.3 for details).

For best comparability of the different optimization approaches, we first investigated the denoising performance of the algorithms under controlled conditions: We considered a fixed task (the “House” benchmark with $\sigma = 50$ AWG noise), and we compared algorithms for three different configurations of D and H (see Fig. 2.5 B). PSNR values for BPFA, MTMKL and GSC are taken from the corresponding original publications. The publication for GSC only reports values for $D = 8 \times 8$ and $H = 64$ as well as for $D = 8 \times 8$ and $H = 256$; MTMKL only for the former and BPFA only for the latter setting. We additionally cite the performance of MTMKL for $D = 8 \times 8$ and $H = 256$ as reported by Sheikh et al. (2014). For $\mathrm{GSVAE}^{\mathrm{lin}}$, EBSC and ES3C, PSNR values were obtained via patch averaging as described above (compare Sec. 2.4.2); the values reported correspond to average PSNRs of three runs of the respective algorithm (standard deviations were smaller or equal 0.13 dB PSNR; results of the individual runs are listed in Tab. 2.S1 in Sec. 2.S6). For EBSC and ES3C, the same EVO hyperparameters were used (hyperparameters, including those used for $\mathrm{GSVAE}^{\mathrm{lin}}$, are listed in Tab. 2.S2 in Sec. 2.S6.2).

Considering Fig. 2.5 B, BPFA and MTMKL (which both represented the state-of-the-art at the time of their publication) perform well but, in comparison, ES3C shows significant further improvements. As the major difference between the considered approaches is the parameter optimization method rather than the generative model, the performance increases of ES3C compared to the other algorithms can be attributed to the used evolutionary variational optimization (EVO). An important factor of the performance increases for ES3C is presumably its ability to leverage the available generative fields more effectively. Titsias and Lázaro-Gredilla (2011) report for MTMKL, for instance, that larger dictionaries ($H > 64$) do not result in performance increases; similarly Zhou et al. (2012) report that BPFA models with 256 and 512 dictionary elements yield similar performance. GSC has been shown to make better use of larger dictionaries than MTMKL (Sheikh et al., 2014). Performance of GSC is, however, significantly lower than ES3C for all investigated dictionary sizes, which provides strong evidence for the advantage of the fully variational optimization applied by the EVO algorithm. The evolutionary variational optimization also results in a notably strong performance of EBSC compared to previous approaches: it outperforms MTMKL (as well as GSC for $H = 64$). The strong performance may be unexpected as the underlying BSC model is less expressive than the SSSC data model. The performance gains using EVO thus outweigh the potentially better data model in this case.

---

[3] Essentially, a categorical distribution is annealed, i.e. becomes continuous. Reparameterization (which is otherwise not applicable to discrete latents) can now be used to estimate gradients. Reversal of the annealing process then recovers discrete latent values in the limit of low temperatures (e.g., Jang et al., 2017, for details).

In comparison, optimization of the BSC model using VAE-like optimization (GSVAE$^{\mathrm{lin}}$) resulted in much lower denoising performance: In all conditions (i.e. for all considered settings of D and H), the PSNR values obtained with GSVAE$^{\mathrm{lin}}$ were significantly lower compared to all other algorithms used for comparison (see Fig. 2.5 B). A reason for the lower PSNR values of GSVAE$^{\mathrm{lin}}$ compared to approaches such as MTMKL or EBSC is a much denser encoding, which, for the considered setting, is less advantageous than the sparse code learned by EBSC and other approaches (we elaborate further below). Finally, while PSNR values for EBSC can be higher than for methods based on spike-and-slab models, ES3C performs (given the same set of hyperparameters) much stronger than EBSC in all settings. The spike-and-slab-based ES3C algorithm can also make much more use of larger dictionaries and larger patches (see Fig. 2.5 B).

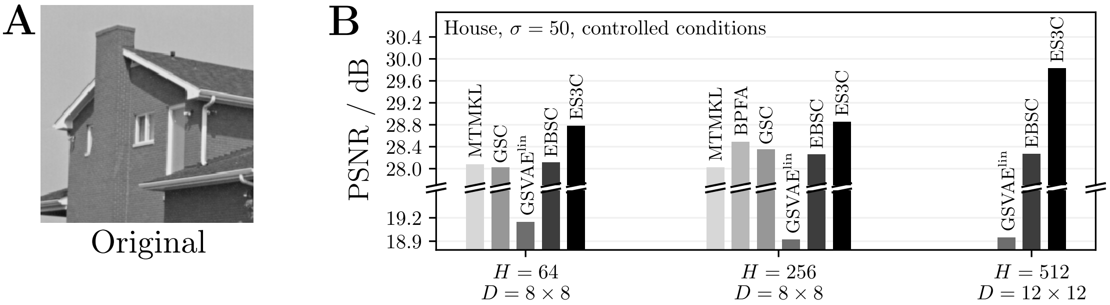
2.4.2.3 General Comparison of Denoising Approaches

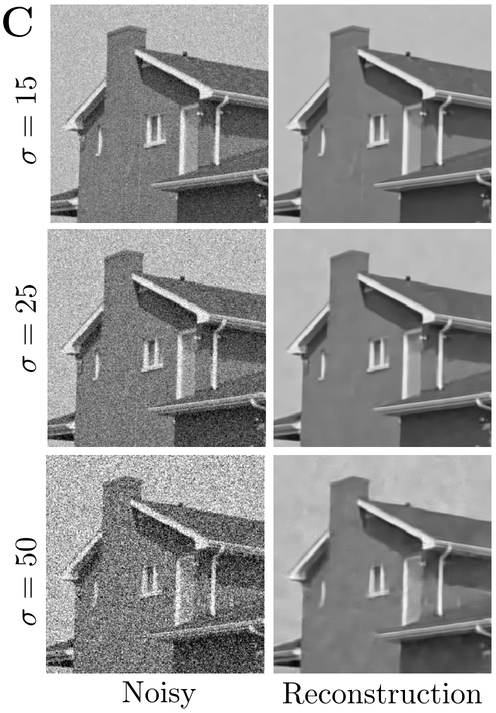
After having compared a selected set of algorithms under controlled conditions, we investigated denoising performance much more generally. We compared the best performing algorithms on standard denoising benchmarks irrespective of the general approach they follow, allowing for different task categories they require, and allowing for any type of hyperparameters they use. The methods we have used for comparison are listed in Tab. 2.2 A, where the algorithms are grouped according to the prior information they assume (compare Sec. 2.4.1 and Sec. 2.S5).

A limitation of most of the investigated algorithms is the computational resources they require. For EBSC and ES3C, model size (i.e., patch and dictionary sizes) are limited by available computational resources and so is the number of variational states S (Sec. 2.2.1) used to approximate posteriors. For ES3C, a good trade-off between performance vs. computational demand was observed for instance for $D = 12 \times 12$ and $H = 512$ with $S = 60$ (for the experiment of Fig. 2.5 D below); for EBSC a good trade-off between performance and computational demand was observed for $D = 8 \times 8$ and $H = 256$ with $S = 200$. We thus have used more variational states S for EBSC than for ES3C. Technical details including EVO hyperparameters are provided in Sec. 2.S6.

Figures 2.5 D and 2.6 list denoising performances obtained with algorithms from the different categories of Tab. 2.2. The listed PSNR values are taken from the respective original publications with the following exceptions: For WNNM, EPLL and MLP, values are taken from Zhang et al. (2017) and for TNRD and MemNet values equal the ones in Dong et al. (2019). PSNR values for GSVAE$^{\mathrm{lin}}$ were obtained based on the source code provided by the original publication (Jang, 2016) but using the same linear decoder as for Fig. 2.5 B. We observed the patch size and the number of binary latents to be significant hyperparameters for GSVAE$^{\mathrm{lin}}$; therefore, compared to the setting in Fig. 2.5 B, we used larger patch sizes and a higher number of latents for GSVAE$^{\mathrm{lin}}$ in the experiments of Figs. 2.5 D and 2.6 (see Tab. 2.S2 and Sec. 2.S6.3 for the hyperparameters used).

PSNR values reported in the original papers are often single PSNR values which may correspond to a single run of the investigated algorithm or to the best observed PSNR value. For deterministic methods with essentially no PSNR variations for a fixed input, it might seem natural to report a single PSNR value. For stochastic methods, reporting the best observed value is instructive about how well a given method can in principle perform. In practice, however, the run with highest PSNR value is less relevant than the average PSNR

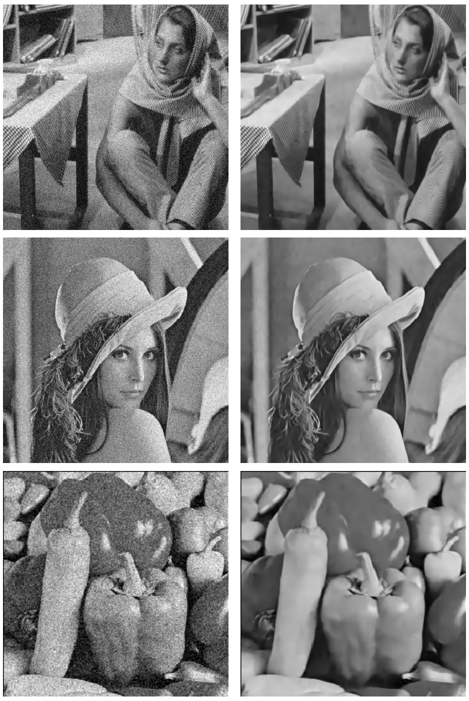




D

		House	
	$\sigma = 15$	$\sigma = 25$	$\sigma = 50$
DE1 MTMKL	34.29	31.88	28.08
DE1 BPFA	34.52	32.24	28.49
DE1 GSC	33.78	32.01	28.35
DE1 GSVAE$^{\text{lin}}$	29.39$^{\varnothing}$	28.32$^{\varnothing}$	25.73$^{\varnothing}$
DE1 EBSC	33.66$^{\varnothing}$	32.40$^{\varnothing}$	28.98$^{\varnothing}$
DE1 ES3C	**34.90**$^{\varnothing}$	**33.15**$^{\varnothing}$	**29.83**$^{\varnothing}$
DE2 KSVD	34.32$^{\varnothing}$	32.15$^{\varnothing}$	27.95$^{\varnothing}$
DE2 LSSC	**35.35**$^{\varnothing}$	33.15$^{\varnothing}$	30.04$^{\varnothing}$
DE2 BM3D	34.94	32.86	29.37
DE2 WNNM	35.13	**33.22**	**30.33**
DE5 EPLL	34.17†	32.17†	29.12†
DE6 MLP	n/a	32.56	29.64
DE6 DnCNN-B	34.93	33.05	30.02
DE6 TNRD	34.55	32.54	29.48
DE6 MemNet	35.10	33.25	30.70
DE6 FFDNet	35.01	33.27	30.43
DE6 DPDNN	**35.40**	**33.54**	**31.04**
DE6 BDGAN	34.57	33.28	30.61
DE6 BRDNet	35.27	33.41	30.53

Figure 2.5: Denoising results for the "House" image (clean original depicted in **A**). **B** Performance comparison for AWG noise with $\sigma = 50$ under controlled conditions (see text for further details). **D** Performance comparison w.r.t. to state-of-the-art denoising approaches and standard baselines using different optimized hyperparameters (see text for further details). Grouping and labeling according to Tab. 2.2; in each group the highest PSNR value is marked bold. Numbers marked with $^{\varnothing}$correspond to averages over multiple independent realizations of the experiment using different realizations of the noise (see text for further details). For EPLL, model selection was performed based on the learning objective of the algorithm (†markers; personal communication with D. Zoran). Panel **C** illustrates the reconstructed images obtained with ES3C in the run with the highest PSNR value.



Noisy Reconstruction

	Barbara $\sigma = 25$	Lena $\sigma = 25$	Peppers $\sigma = 25$
DE1 BPFA	29.88	31.63	30.00
DE1 GSVAE$^{\text{lin}}$	25.12$^{\varnothing}$	28.01$^{\varnothing}$	25.53$^{\varnothing}$
DE1 EBSC	28.38$^{\varnothing}$	30.88$^{\varnothing}$	28.93$^{\varnothing}$
DE1 ES3C	**30.19**$^{\varnothing}$	**31.95**$^{\varnothing}$	**30.27**$^{\varnothing}$
DE2 KSVD	29.60$^{\varnothing}$	31.32$^{\varnothing}$	29.73$^{\varnothing}$
DE2 LSSC	30.47$^{\varnothing}$	31.87$^{\varnothing}$	30.21$^{\varnothing}$
DE2 BM3D	30.72	32.08	30.16
DE2 WNNM	**31.24**	**32.24**	**30.42**
DE5 EPLL	28.61†	31.73†	30.17†
DE6 MLP	29.21	32.12	30.25
DE6 DnCNN-B	29.69	32.42	30.84
DE6 TNRD	29.41	32.00	30.55
DE6 MemNet	29.98	32.51	30.87
DE6 FFDNet	29.98	32.59	30.79
DE6 DPDNN	30.30	32.69	30.90
DE6 BDGAN	29.95	**32.77**	30.88
DE6 BRDNet	**30.34**	32.65	**31.04**

Figure 2.6: Denoising results for “Barbara”, “Lena” and “Peppers”. The table shows a performance comparison w.r.t. to state-of-the-art denoising approaches and standard baselines using different optimized hyperparameters (see text for further details). Grouping and labeling according to Tab. 2.2; in each group the highest PSNR value is marked bold. Numbers marked with $^{\varnothing}$correspond to averages over multiple independent realizations of the experiment using different realizations of the noise (see text for further details). For EPLL, model selection was performed based on the learning objective of the algorithm (†markers; personal communication with D. Zoran). On the left we illustrate the reconstructed images obtained with ES3C in the run with the highest PSNR value.

as selecting the run with best PSNR usually requires ground-truth (compare discussion in Sec. 2.S5). For KSVD and LSSC as well as for GSVAE$^{\text{lin}}$, EBSC and ES3C, the PSNR values listed in Figs. 2.5 D and 2.6 correspond to averages. For the two former approaches, averages were obtained from five runs of the algorithms with different noise realizations (standard deviations were reported to be negligibly small; see Elad and Aharon, 2006, Mairal et al., 2009). For GSVAE$^{\text{lin}}$, EBSC and ES3C, we performed three runs of the algorithms with different noise realizations (observed standard deviations of the resulting PSNRs were not larger than 0.06 dB for EBSC and ES3C and 0.23 dB for GSVAE$^{\text{lin}}$; compare Tab. 2.S1).

Considering Figs. 2.5 D and 2.6, it can be observed that the performance tends to increase the more a-priori information is used by an algorithm (as can be expected). Feed-forward DNNs report overall the best PSNR values but also use the most a-priori information (see category DE6, Tab. 2.2). In the 'zero-shot' category with no a-priori information (DE1), ES3C shows the best performance (Fig. 2.5 D). ES3C significantly increases the state-of-the-art PSNR values across all investigated settings. EBSC is very competitive to previous approaches in the "House" benchmark (but a bit less so for the images of Fig. 2.6). While being based on the same data model as EBSC, GSVAE$^{\text{lin}}$ was observed to result in much lower PSNRs than EBSC. Again, we observed a much denser encoding for GSVAE$^{\text{lin}}$ compared to other approaches used for the experiments. For the here considered benchmarks, such dense codes do not seem advantageous (while we remark that GSVAE$^{\text{lin}}$ and other GSVAE versions are observed to perform well on other data, see Sec. 2.S6.3 for details).

Notably, the spike-and-slab based ES3C approach is still competitive when compared to algorithms in categories DE2 and DE5 that use more a-priori information, but it is outperformed by some of these methods, especially at lower noise levels (Figs. 2.5 D and 2.6). Differences in performance between the categories get smaller for larger and more complex images (Fig. 2.6). For the image "Barbara", for instance, ES3C outperforms all DNNs except DPDNN and BRDNet, which are slightly better, while the best result is obtained for the image specific non-local method WNNM.

PSNR values achieved by the previous methods of BPFA or MTMKL have notably proven to be difficult to further improve. Only systems using more a-priori information (such as WNNM in category DE2 or DNNs in category DE6) have more recently reported higher PSNR values (also see discussion of difficult to improve BM3D values by Gu et al., 2014). In category DE1, the only more recent competitive approach is a Bayesian version of K-SVD (BKSVD; Serra et al., 2017). By extending K-SVD using Bayesian methodology, the noise level can be estimated from data. The BKSVD approach can thus be applied without a-priori information on the noise level, which makes it applicable in the 'zero-shot' category (see Serra et al., 2017, for a discussion). PSNR values for BKSVD reported by Serra et al. (2017) are computed on downscaled images, and are therefore not directly comparable to those of Figs. 2.5 D and 2.6. However, comparisons of BKSVD with MTMKL, K-SVD and BPFA reported by Serra et al. (2017) show that BKSVD performs competitively, especially for lower noise levels. For higher noise levels, though, (e.g. $\sigma = 25$ for "Lena" and "Peppers") BPFA results in higher PSNR values (and in similar PSNRs for "Barbara"). As ES3C outperforms BPFA for $\sigma = 25$ on "Barbara", "Lena" and "Peppers" on the full resolution images, BKSVD also does not seem to establish a new state-of-the-art in the DE1 category. We can, therefore, conclude that ES3C establishes a novel state-of-the-art in the 'zero-shot' category since a relatively long time.

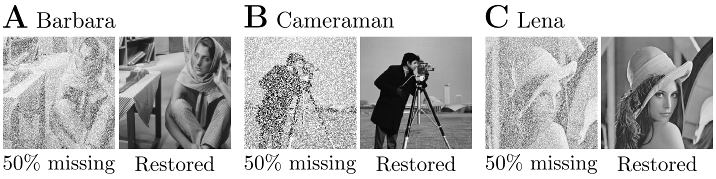
2.4.3 Image Inpainting

The final benchmark we used is image inpainting, i.e. the reconstruction of missing values in images. A standard setting for this task is the restoration of images that have been artificially degraded by uniformly randomly deleting pixels. Here we considered the standard test images "Barbara", "Cameraman", "Lena", "House" and "Castle" as examples for gray-scale and RGB images. For denoising, ES3C consistently performed better than EBSC, and the same we observed for inpainting. The performance difference between ES3C and EBSC on inpainting was, however, larger (sometimes several dB). We therefore focus on ES3C for comparison as EBSC is not competitive to the best performing methods we compare to.

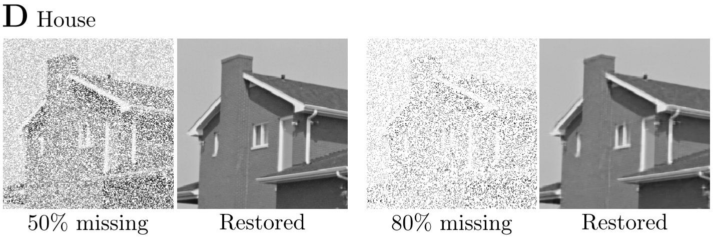
For inpainting with ES3C, we employed the same partitioning and averaging scheme as described in Sec. 2.4.2. For the RGB image, ES3C was applied jointly to all color channels, as this had been reported to be more beneficial than training individual models on different color layers (Mairal et al., 2008). We trained the model and restored missing values exclusively based on the incomplete data which we aimed to restore (following, e.g., Fadili and Starck, 2005; Little and Rubin, 1987); uncorrupted pixels were not modified. Details about the missing value estimator applied by ES3C are provided in Sec. 2.2.3 and in Sec. 2.S3. For inpainting, we used a SSSC model with fixed $\vec{\mu} = \vec{1}$ and $\Psi = \mathbb{1}$ as these parameters were observed to grow exponentially during learning which led to numerical instabilities of the training process. For each test scenario, we performed three runs of ES3C and used a different realization of missing values in each run (EVO hyperparameters are listed in Tab. 2.S2 in Sec. 2.S6.2). Compared to the denoising experiments (Sec. 2.4.2.3), we observed for inpainting slightly larger variations of the resulting PSNR values (standard deviations were smaller or equal 0.18 dB; results of individual runs listed in Tab. 2.S1 in Sec. 2.S6).

The inpainting results for "Barbara", "Cameraman", "Lena" (each 50% missing values) and "House" and "Castle" (each 50% and 80% missing values) are depicted in Fig. 2.7 A-E . The reference PSNR values are taken from the original publications except for NL and FoE for which we cite the numbers as reported by Yu et al. (2012). In general, inpainting is a less wide-spread benchmark than denoising presumably because additional estimation routines have to be developed and missing values pose, more generally, a considerable challenge to many approaches. DNN-based approaches have to define, for instance, how a missing pixel of the input image is treated (which can be challenging, e.g., if the positions of the missing pixels are different from patch to patch as for this benchmark). The challenge is faced by approaches such as GSVAE (and VAE approaches in general) because DNNs are used for VAE encoders. Considering Fig. 2.7 A-E, first observe that performance differences between categories are larger than for denoising. Compared to the best performing approaches in the literature, ES3C shows to be competitive in many settings. On the "Castle" benchmark, ES3C performs better than all other approaches in the literature, with the exception of PLE, which uses the noise level as a-priori information. For the "House" benchmark, ES3C establishes a novel state-of-the-art in general: it performs better than all other algorithms for 50% lost pixels and equal to PLE for 80% lost pixels (i.e., within PSNR standard deviation of ES3C). This novel state-of-the-art is notably reached without requiring clean training data or a-priori knowledge about noise level or sparsity (ES3C requires to know which pixels are missing; compare Tab. 2.2 and Sec. 2.S5).

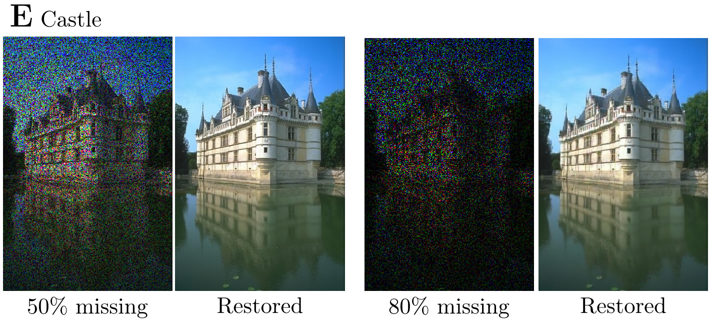
Finally, we also report inpainting performance on the well-known "New Orleans" inpainting benchmark (Fig. 2.7 F left-hand-side). The benchmark serves as an inpainting example



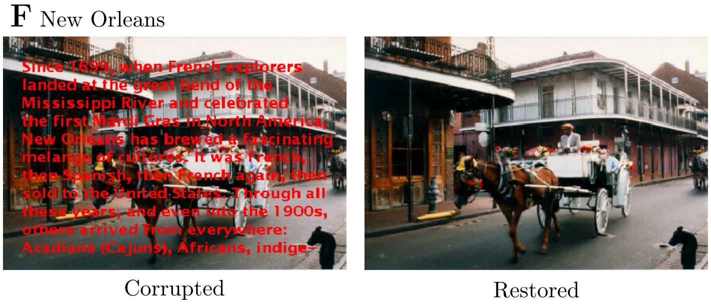
	Barbara	C.man	Lena
IN1 BPFA	33.17	28.90	36.94
IN1 DIP	32.22	29.80	36.16
IN1 ES3C	**35.44**$^{\varnothing}$	**30.69**$^{\varnothing}$	**37.54**$^{\varnothing}$
IN2 NL	36.40	n/a	**37.82**
IN2 PLE	**37.03**	n/a	37.78
IN4 FoE	29.47	n/a	36.66



	50%	80%
IN1 BPFA	38.02	30.12
IN1 DIP	39.16	n/a
IN1 ES3C	**39.59**$^{\varnothing}$	**33.06**$^{\varnothing}$
IN2 NL	**39.30**	32.87
IN2 PLE	38.97	**33.05**
IN4 FoE	37.99	31.28



	50%	80%
IN1 MTMKL	n/a	28.94
IN1 BPFA	36.45	29.12
IN1 ES3C	**38.23**$^{\varnothing}$	**29.66**$^{\varnothing}$
IN2 PLE	38.34	30.07
IN5 cKSVD	n/a	29.65
IN6 IRCNN	n/a	28.74


F New Orleans
Since 1699, when French explorers
landed at the great bend of the
Mississippi River and celebrated
the first Mardi Gras in North America,
New Orleans has brewed a fascinating
melange of cultures. It was French,
then Spanish, then French again, then
sold to the United States. Through all
these years, and even into the 1900s,
others arrived from everywhere:
Acadians (Cajuns), Africans, indige-
Corrupted
Restored

IN1 ES3C	32.17
IN4 FoE	**32.39**
IN4 MRF	31.69
IN5 cKSVD	32.45
IN6 NLRMRF	**32.59**
IN6 IRCNN	30.95

Figure 2.7: Inpainting results for "Barbara", "Cameraman", "Lena", "House", "Castle" and "New Orleans". For the performance comparison, different approaches are grouped and labeled according to Tab. 2.2. In each group the highest PSNR value is marked bold. Numbers marked with $^{\varnothing}$in **A** to **E** correspond to averages over multiple independent runs of the experiment using different realizations of missing values (see text for details; the performance of ES3C in the individual runs is reported in Tab. 2.S1 in Sec. 2.S6). On the left, we illustrate the reconstructed images obtained with ES3C in the run with the highest PSNR value.

where pixels are lost non-randomly. For the benchmark, the original image is artificially corrupted by adding overlaid text, which is then removed by applying inpainting. Compared to inpainting with randomly missing values (e.g., the benchmarks in Fig. 2.7 A-E), the "New Orleans" image contains relatively large contiguous regions of missing values. For our measurements, we used a publicly available data set[4]. The result of the experiment is illustrated in Fig. 2.7 F. Reference PSNR values are taken from respective original publications with exceptions of FoE and MRF for which we use values reported by Sun and Tappen (2011). Due to extensive runtimes, we report the result of a single run of ES3C for this benchmark (see Sec. 2.S6 for further details; EVO hyperparameters listed in Tab. 2.S2). As can be observed, ES3C also performs well on this task. We observed higher PSNR values for ES3C than for MRF and IRCNN. Values for FoE and cKSVD are higher, but more a-priori information is also used. Especially FoE is less sensitive to larger areas of lost pixels than ES3C (compare the lower performance of FoE in Fig. 2.7 A-D); presumably FoE can make better use of larger contexts. The state-of-the-art on this benchmark is established by NLRMRF.

2.5 Discussion

Efficient approximate inference is of crucial importance for training expressive generative models. Consequently, there exists a range of different methods with different assumptions and different features. As most generative models can only be trained using approximations, and usually many locally optimal solutions exist, the question of how well a generative model can potentially perform on a given task is difficult to answer. Sophisticated, mathematically grounded approaches such as sampling or variational optimization have been developed in order to derive sufficiently precise and efficient learning algorithms. As a contribution of this work, we have proposed a general purpose variational optimization that directly and intimately combines evolutionary algorithms (EAs) and EM. Considering Alg. 1, EAs are an integral part of variational EM where they address the key optimization problem (the variational loop) arising in the training of directed generative models. The EVO algorithm is defined by a set of optimization hyperparameters for the EA. Given the hyperparameters, EVO is applicable to a given generative model solely by using the model's joint probability $p(\vec{s}, \vec{y} \,|\, \Theta)$. No analytical derivations are required to realize variational optimization.

Relation to Other Approximate Inference Approaches. To show large-scale applicability of EVO, we considered high-dimensional image data and generative models with large (combinatorial) state spaces. Aside from the elementary generative models of Noisy-OR Bayes Nets (Jernite et al., 2013; Rotmensch et al., 2017; Šingliar and Hauskrecht, 2006) and binary sparse coding (Haft et al., 2004; Henniges et al., 2010), we used spike-and-slab sparse coding (SSSC) as a more expressive example. The SSSC data model has been of

---

[4] We downloaded the clean and the corrupted image from `https://lear.inrialpes.fr/people/mairal/resources/KSVD_package.tar.gz` and the text mask from `https://www.visinf.tu-darmstadt.de/media/visinf/software/foe_demo-1_0.zip`; all images were provided as PNG files. We verified that the corrupted image was consistent w.r.t. the mask and the original image contained in the data set: We applied the mask to the original image and compared the unmasked parts to the unmasked parts of the corrupted image. These were identical. After filling masked areas with red (i.e., replacing the corresponding pixels with (255,0,0) in RGB space) we measured a PSNR value of the corrupted image of 13.48 dB. This value slightly deviates from numbers reported in other studies; Chaudhury and Roy, for example, measured a PSNR of 15.05 dB.

considerable interest due to its strong performance for a range of tasks (Goodfellow et al., 2012; Sheikh et al., 2014; Titsias and Lázaro-Gredilla, 2011; Zhou et al., 2009, 2012). However, its properties require sophisticated probabilistic approximation methods for parameter optimization. Essentially three types of approximations have previously been applied to train the model: mean field (Goodfellow et al., 2012; Titsias and Lázaro-Gredilla, 2011), truncated posterior approximations (Sheikh et al., 2014), and MCMC sampling (Mohamed et al., 2012; Zhou et al., 2009, 2012). If we consider the BSC data model as a boundary case of a variational autoencoder with categorical latents (Jang et al., 2017), we can also include Gumbel-Softmax-based VAE optimization as a fourth optimization approach. EVO shares features with most of these previous approaches: like work by Sheikh et al. (2014), EVO uses truncated posterior distributions to approximate full posteriors, which contrasts with factored posterior approximations (i.e., mean field) used by Goodfellow et al. (2012); Titsias and Lázaro-Gredilla (2011). However, like work by Titsias and Lázaro-Gredilla (2011) and Goodfellow et al. (2012), EVO is a fully variational EM algorithm which is guaranteed to monotonically increase the variational lower bound. This contrasts with Sheikh et al. (2014) or Shelton et al. (2017) who used feed-forward estimates to construct approximate posteriors (without a guarantee to monotonically increase a variational bound). Furthermore, like sampling-based approaches (Mohamed et al., 2012; Zhou et al., 2009, 2012), EVO computes estimates of posterior expectations based on a finite set of latent states. However, unlike for sampling approximations, these states are not samples but are themselves variational parameters that are evolved using EAs (with a fitness defined by a variational bound).

VAE optimization using the Gumbel-Softmax trick is the most different to EVO and to all previous approaches. The investigated GSVAE algorithms use sampling-based approximations such as Zhou et al. (2009, 2012) or Mohamed et al. (2012), and they optimize a variational lower bound such as ES3C and EBSC. GSVAE is also similar to GSC as both methods use deterministic feed-forward mappings in their definition of variational distributions (amortization). However, GSVAE is, in contrast to the previously discussed methods, a further development of standard VAE optimization. More concretely, it uses a DNN for encoding and a "softening" (i.e. annealing) of discrete latents (the Gumbel-Softmax trick) in order to apply standard VAE optimization based on reparameterization and gradient ascent. Standard VAE optimization was, on the other hand, originally defined for latents with Gaussian prior, which typically gives rise to a dense code (and such codes will be of relevance for the discussion of denoising experiments below).

Denoising and Inpainting Benchmarks and Comparison with Other Approaches. For comparison with a range of other methods, we used standard denoising and inpainting benchmarks and evaluated EVO for the SSSC model (termed *ES3C*) and EVO for the BSC model (termed *EBSC*). Compared to competing methods in the 'zero-shot' category (see category DE1 and IN1 in Tab. 2.2), we observed ES3C to achieve the highest PSNR values on all of the benchmarks of Figs. 2.5 to 2.7 that we performed. Furthermore, ES3C also shows improvements w.r.t. methods that use more a-priori information in many test scenarios:

- The denoising results (Figs. 2.5 and 2.6) show that ES3C can outperform state-of-the-art sparse coding and non-local approaches including, e.g., LSSC and BM3D, which require ground-truth noise level information. ES3C also improves on the popular GMM-based EPLL method, which additionally uses clean training data. Furthermore, ES3C shows improvements compared to a number of recent, intricate DNNs which are tailored to image denoising and require (large amounts of) clean data for training. For instance, on the “Barbara” image (Fig. 2.6), ES3C is only outperformed by the very recent networks DPDNN (Dong et al., 2019) and BRDNet (Tian et al., 2020). At the same time, we note that the image noise for the benchmarks is Gaussian. Gaussian noise is commonly encountered in real data, but it also makes the SSSC model well suited for this task in general.

- For image inpainting, we observed ES3C to provide state-of-the-art results also in general (i.e. across all categories) in some settings: compared to the best performing algorithms on the “House” benchmark for inpainting, ES3C achieves the highest PSNR value (Fig. 2.7 D, best value shared with PLE for 80% lost pixels). The inpainting benchmark is, however, much less frequently applied than its denoising counterpart for the same image because a treatment of missing data is required.

The observation that probabilistic approaches such as SSSC-based algorithms perform well on the inpainting benchmarks of Fig. 2.7 A-E may not come as too much of a surprise. In the case of randomly missing values, the information provided by relatively small patches contains valuable information which an SSSC model can leverage well. Larger areas of missing values, as encountered in the “New Orleans” benchmark, require larger contexts, whose information may be better leveraged by less local methods such as Markov Random Fields (Fig. 2.7 F).

To answer the question of why EVO results in improvements, direct comparison with other optimization methods for the same or similar models are most instructive (Fig. 2.5 B). In general, the type of variational optimization can have a strong influence on the type of generative fields that are learned, e.g., for the SSSC model. Mean field methods (i.e., fully factored variational distributions) have a tendency to bias generative fields to be orthogonal (Ilin and Valpola, 2005; Mackay, 2001). Such effects are also discussed for other generative models (e.g. Turner and Sahani, 2011; Vértes and Sahani, 2018). Biases introduced by a variational approximation can thus lead to increasingly suboptimal representations (see, e.g., Titsias and Lázaro-Gredilla, 2011, Sheikh et al., 2014, for discussions). Sampling is in general more flexible but practical implementations may (e.g., due to insufficient mixing) also bias parameter learning towards learning posteriors with single modes. The denoising benchmark under controlled conditions shows that the sampling-based BPFA approach (Zhou et al., 2012) performs, for instance, better than the mean field approach MTMKL (Titsias and Lázaro-Gredilla, 2011) and the truncated EM approach GSC (Sheikh et al., 2014); but BPFA performs worse than ES3C, which is based on EVO. Likewise, also VAE-like training using Gumbel-Softmax is a specific optimization which impacts the learned representation. In our experiments, VAE-like optimization showed the strongest differences not only to EVO but also to all other approaches that were based on an SSSC or a BSC-like generative model.

In contrast to, e.g., MTMKL, GSC, ES3C and EBSC, the GSVAE approach resulted in a dense encoding of image patches. Concretely, instead of learning generative fields that resemble image structures such as edges or texture components, GSVAE learned fields with much less interpretable structures (and many of the learned fields represented relatively high spatial frequencies, see Fig. 2.S4). For reconstruction, GSVAE used large fractions of these fields (usually about half on average). In other words, GSVAE uses dense codes, while MTMKL, GSC, ES3C or EBSC use sparse codes for encoding (few fields are combined for reconstruction). Notably, ES3C or EBSC are by construction not constrained to learn a sparse code (parameters for the Bernoulli prior are learned). Neither is GSVAE by construction constrained to a dense code (prior parameters of GSVAE that correspond to a sparse code can be learned in principle). Still, in all applications to natural image data, ES3C and EBSC learned sparse codes; and in all applications to natural image data, GSVAE was observed to learn a dense code (for all GSVAE versions investigated, compare Sec. 2.S6.3). Comparison on denoising benchmarks clearly shows that the dense codes learned by GSVAE$^{\text{lin}}$ are much less competitive than the sparse codes of the other approaches (Figs. 2.5 and 2.6). At the same time, we remark that the used versions and implementations of GSVAE are competitive on other benchmarks. Compared to other methods in the literature (e.g. Park et al., 2019a), we observed dense codes of GSVAE to result in good benchmarking performance on data sets with many images of whole objects, for instance. On the other hand, for such data, algorithms giving rise to sparse codes such as EBSC are not learning edge or texture like components and are observed to be less competitive (see Sec. 2.S6.3 for such controls).

The differences between algorithms with dense and sparse codes may be an important reason why deep generative models have not been reported to perform competitively in denoising. For the benchmarks in Figs. 2.5 and 2.6, the recent BDGAN approach is, besides GSVAE, the only deep generative model. Except for BDGAN, neither for generative adversarial nets (GANs; Goodfellow et al., 2014) nor for GSVAE or other variational autoencoders competitive denoising or inpainting results are reported on these standard benchmarks (for GANs and VAEs, benchmarks other than those in Figs. 2.5 to 2.7 are often considered; see Sec. 2.S6.3 for further discussion). For BDGAN, Zhu et al. (2019) report performances on standard denoising benchmarks; the algorithm shows competitive performance but, like feed-forward DNNs, requires large image corpora for training. At least in principle, GANs and VAEs are also applicable to the 'zero-shot' category (DE1 and IN1 in Figs. 2.5 to 2.7), however (and GSVAE represents one such example). The BDGAN approach uses large and intricate DNNs whose parameters are presumably difficult to train using just one image, which may explain why it was not applied in the 'zero-shot' category. Recent work by, e.g. Shocher et al. (2018), explicitly discusses DNN sizes and the use of small DNNs for 'zero-shot' super-resolution.

For VAEs, there is (except of the results reported here for GSVAE) neither data for the 'zero-shot' category nor for the other categories available for the standard benchmarks we used (to the knowlege of the authors; but compare Prakash et al., 2021b). A reason may be that (like for BDGAN) very intricate DNNs as well as sophisticated sampling and training methods are required to be competitive. Experiments with standard (Gaussian) VAE setups that we conducted did, e.g. for denoising, not result in PSNR values close to those reported in Figs. 2.5 and 2.6. Another possible reason for the absence of competitive values for VAEs may, however, be related to the variational approximation used. VAEs for continuous data usually use Gaussians as variational distributions (Kingma and Welling, 2014; Rezende et al., 2014) that are in addition fully factored (i.e., mean field). The parameters of the VAE decoder may thus suffer from similar biasing effects as described for mean field approaches as used, e.g., by MTMKL (Titsias and Lázaro-Gredilla, 2011) for the SSSC model. That standard VAEs *do* show such biases has recently been pointed out, e.g., by Vértes and Sahani (2018). Furthermore, the dense codes learned by VAEs due to their usual Gaussian priors seem to lead to representations less suitable for denoising (but potentially well suited for other tasks). Even though sparse codes could be learned in principle (e.g., GSVAE can in principle learn sparse codes), the usual optimization methods for VAEs seem to favor dense codes.

Like other generative models, VAEs (which use DNNs as part of their encoders and decoders) are, consequently, likely to strongly profit from a more flexible encoding of variational distributions. In the context of deep learning, such flexible variational distributions have been suggested, e.g., in the form of normalizing flows (Rezende and Mohamed, 2015). In its standard version (Rezende and Mohamed, 2015), normalizing flows are making training significantly more expensive, however. Approximate learning using normalizing flows is centered around a sampling-based approximation of expectation values w.r.t. posteriors (Rezende and Mohamed, 2015; and also compare related approaches, e.g., Bigdeli et al., 2020; Huang et al., 2018). By following such a strategy, samples of increasingly complex variational distributions can be generated by successively transforming samples of an elementary distribution. The idea of successive transformations has also been used to directly parameterize the data distribution in what is now termed flow-based models (e.g. Dinh et al., 2017; Kingma and Dhariwal, 2018). An early example (Dinh et al., 2014) is a form of non-linear ICA (Hyvärinen and Pajunen, 1999). By using EAs, the EVO approach does, in contrast, follow a strategy that is notably very different from normalizing flows and other approaches. A consequence of these different strategies are very different sets of hyperparameters: EVO hyperparameters are centered around parameters for the EA such as population size as well as mutation, crossover and selection operators; hyperparameters of normalizing flows and related methods are centered around parameterized successive mappings (e.g., type of mapping, number of transformations) and (if applicable) hyperparameters for sampling. Furthermore, and most importantly for practical applications, the types of generative models addressed by EVO and normalizing flows are essentially complementary: EVO focuses on generative models with discrete latents while normalizing flows and others focus on models with continuous latents.

Conclusion. More generally, EVO can be regarded as an example of the significant potential in improving the accuracy of posterior approximations. Many current research efforts are focused on making parameterized models increasingly complex: usually an intricate DNN is used either directly (e.g., feed-forward DNNs/CNNs for denoising) or as part of a generative model. At least for tasks such as denoising or inpainting, our results suggest that improving the flexibility of approximation methods may be as effective in improving performance as increasing model complexity. While EVO shares the goal of a flexible and efficient posterior approximation with many other powerful and successful methods, it has a distinguishing feature: a direct link from variational optimization to a whole other research field focused on optimization: evolutionary algorithms. For generative models with discrete latents, new results in the field of EAs can consequently be leveraged to improve the training of future (elementary and complex) generative models.

Supplementary Material

2.S1 Log-Pseudo Joint Formulation

For efficient computation of log-joint probabilities $\log p\,(\vec{y}, \vec{s} \mid \Theta)$ (which are required to compute truncated expectations (2.3) and the variational lower bound (2.4)), we introduce the following reformulation. When computing $\log p\,(\vec{y}, \vec{s} \mid \Theta)$ for a concrete model, we identify the terms that only depend on the model parameters Θ (and not on the data points $\vec{y}$ and the variational states $\vec{s}$). These terms, denoted $\mathrm{C}(\Theta)$ in the following, only need to be evaluated once when computing $\log p\,(\vec{y}, \vec{s} \mid \Theta)$ for different data points and for different variational states. We refer to the remaining terms that depend on both $\vec{s}$, $\vec{y}$ and Θ as *log-pseudo joint* and denote this quantity by $\widetilde{\log p}(\vec{s}, \vec{y} \mid \Theta)$. The log-joint probability $\log p\,(\vec{y}, \vec{s} \mid \Theta)$ can then be reformulated as:

$$\log p\,(\vec{y}, \vec{s} \mid \Theta) = \widetilde{\log p}(\vec{s}, \vec{y} \mid \Theta) + \mathrm{C}(\Theta). \tag{2.S1}$$

The concrete expressions of $\mathrm{C}(\Theta)$ and $\widetilde{\log p}(\vec{s}, \vec{y} \mid \Theta)$ for the models introduced in Sec. 2.3.1 are listed below.

NOR.

$$\mathrm{C}(\Theta) = \sum_{h=1}^{H} \log(1 - \pi_h), \tag{2.S2}$$

$$\widetilde{\log p}(\vec{s}, \vec{y} \mid \Theta) = \sum_{d=1}^{D} \Big(y_d \log\big(N_d(\vec{s})\big) + (1 - y_d) \log\big(1 - N_d(\vec{s})\big) \Big) + \sum_{h=1}^{H} s_h \log\Big(\frac{\pi_h}{1 - \pi_h}\Big), \tag{2.S3}$$

$$N_d(\vec{s}) = 1 - \prod_{h=1}^{H} (1 - W_{dh} s_h), \tag{2.S4}$$

with the special exception of $\vec{s} = \vec{0}$ for which

$$\widetilde{\log p}(\vec{s} = \vec{0}, \vec{y}) = \begin{cases} 0 & \text{if } \vec{y} = \vec{0}, \\ -\inf & \text{otherwise.} \end{cases} \tag{2.S5}$$

For practical computations, we set $\widetilde{\log p}$ to an arbitrarily low value rather than the floating point infinite representation.

BSC.

$$\mathrm{C}(\Theta) = H \log(1-\pi) - \frac{D}{2}\log(2\pi\sigma^2), \tag{2.S6}$$

$$\widetilde{\log p}(\vec{s},\vec{y} \mid \Theta) = \log\left(\frac{\pi}{1-\pi}\right)\sum_{h=1}^{H} s_h - \frac{1}{2\sigma^2}(\vec{y}-W\vec{s})^{\mathrm{T}}(\vec{y}-W\vec{s}). \tag{2.S7}$$

SSSC.

$$\mathrm{C}(\Theta) = \sum_{h=1}^{H} \log(1-\pi_h) - \frac{D}{2}\log(2\pi), \tag{2.S8}$$

$$\widetilde{\log p}(\vec{s},\vec{y} \mid \Theta) = \sum_{h=1}^{H} s_h \log\left(\frac{\pi_h}{1-\pi_h}\right) - \frac{1}{2}\log|C_{\vec{s}}| - \frac{1}{2}(\vec{y}-\tilde{W}_{\vec{s}}\vec{\mu})^{\mathrm{T}}C_{\vec{s}}^{-1}(\vec{y}-\tilde{W}_{\vec{s}}\vec{\mu}). \tag{2.S9}$$

2.S2 Sparsity-Driven Bitflips

When performing sparsity-driven bitflips, we flip each bit of a particular child $\vec{s}$ with probability p_0 if it is 0, with probability p_1 otherwise. We call p_{bf} the average probability of flipping any bit in $\vec{s}$. We impose as constraints on p_0 and p_1 that $p_1 = \alpha p_0$ for some constant α and that the average number of "on" bits after mutation is set to $\widetilde{s}$. This yields the following expressions for p_0 and p_1:

$$p_0 = \frac{H p_{\mathrm{bf}}}{H + (\alpha-1)|\vec{s}|}, \qquad \alpha = \frac{(H-|\vec{s}|)\cdot(H p_{\mathrm{bf}} - \widetilde{s} + |\vec{s}|)}{(\widetilde{s} - |\vec{s}| + H p_{\mathrm{bf}})|\vec{s}|}. \tag{2.S10}$$

Trivially, random uniform bitflips correspond to the case $p_0 = p_1 = p_{\mathrm{bf}}$. For our numerical experiments (compare Sec. 2.3.2), we chose $\widetilde{s}$ based on the sparsity learned by the model (we set $\widetilde{s} = \sum_{h=1}^{H}\pi_h$ for NOR and SSSC and $\widetilde{s} = H\pi$ for BSC). We furthermore used an average bitflip probability of $p_{\mathrm{bf}} = \frac{1}{H}$.

2.S3 Data Estimator

Models With Binary Latents and Continuous Observables. Consider the posterior predictive distribution

$$p(\vec{y}^{\,\mathrm{est}} \mid \vec{y}^{\,\mathrm{obs}},\Theta) = \sum_{\vec{s}} p(\vec{y}^{\,\mathrm{est}},\vec{s} \mid \vec{y}^{\,\mathrm{obs}},\Theta) = \sum_{\vec{s}} p(\vec{y}^{\,\mathrm{est}} \mid \vec{s},\Theta)\, p(\vec{s} \mid \vec{y}^{\,\mathrm{obs}},\Theta). \tag{2.S11}$$

The second step in Eq. (2.S11) exploits the fact that $\vec{y}^{\,\mathrm{est}}$ and $\vec{y}^{\,\mathrm{obs}}$ are conditionally independent given the latents. In order to infer the value of y_d^{est} we will take expectations w.r.t. $p(\vec{y}^{\,\mathrm{est}} \mid \vec{y}^{\,\mathrm{obs}})$:

$$\mathbb{E}_{p(\vec{y}^{\,\mathrm{est}} \mid \vec{y}^{\,\mathrm{obs}},\Theta)}\big[y_d^{\mathrm{est}}\big] = \int_{\vec{y}^{\,\mathrm{est}}} y_d^{\mathrm{est}}\, p(\vec{y}^{\,\mathrm{est}} \mid \vec{y}^{\,\mathrm{obs}},\Theta)\,\mathrm{d}\vec{y}^{\,\mathrm{est}} = \int_{y_d^{\mathrm{est}}} y_d^{\mathrm{est}}\, p(y_d^{\mathrm{est}} \mid \vec{y}^{\,\mathrm{obs}},\Theta)\,\mathrm{d}y_d^{\mathrm{est}}. \tag{2.S12}$$

The second step in Eq. (2.S12) follows from marginalizing over $\vec{y}^{\,\mathrm{est}}\backslash y_d^{\mathrm{est}}$. Equations (2.S11) and (2.S12) can be combined:

$$
\begin{aligned}
\mathbb{E}_{p(\vec{y}^{\,\mathrm{est}}\,|\,\vec{y}^{\,\mathrm{obs}},\Theta)}\big[y_d^{\mathrm{est}}\big] &= \int\limits_{y_d^{\mathrm{est}}} y_d^{\mathrm{est}} \sum_{\vec{s}} p(y_d^{\mathrm{est}}\mid\vec{s},\Theta)\,p(\vec{s}\mid\vec{y}^{\,\mathrm{obs}},\Theta)\,\mathrm{d}y_d^{\mathrm{est}} && (2.S13)\\
&= \sum_{\vec{s}} \int\limits_{y_d^{\mathrm{est}}} y_d^{\mathrm{est}}\,p(y_d^{\mathrm{est}}\mid\vec{s},\Theta)\,\mathrm{d}y_d^{\mathrm{est}}\,p(\vec{s}\mid\vec{y}^{\,\mathrm{obs}},\Theta) && (2.S14)\\
&= \sum_{\vec{s}} \mathbb{E}_{p(y_d^{\mathrm{est}}\,|\,\vec{s},\Theta)}\big[y_d^{\mathrm{est}}\big]\,p(\vec{s}\mid\vec{y}^{\,\mathrm{obs}},\Theta) && (2.S15)\\
&= \mathbb{E}_{p(\vec{s}\,|\,\vec{y}^{\,\mathrm{obs}},\Theta)}\Big[\mathbb{E}_{p(y_d^{\mathrm{est}}\,|\,\vec{s},\Theta)}\big[y_d^{\mathrm{est}}\big]\Big]. && (2.S16)
\end{aligned}
$$

The inner expectation in Eq. (2.S16) is for the example of the BSC model (2.11) given by $\mathbb{E}_{p(y_d\,|\,\vec{s},\Theta)}\big[y_d\big] = \vec{W}_d\vec{s}$ with $\vec{W}_d$ denoting the d-th row of the matrix W. With this, Eq. (2.S16) takes for the BSC model the following form

$$
\mathbb{E}_{p(\vec{y}^{\,\mathrm{est}}\,|\,\vec{y}^{\,\mathrm{obs}},\Theta)}\big[y_d^{\mathrm{est}}\big] = \vec{W}_d\mathbb{E}_{p(\vec{s}\,|\,\vec{y}^{\,\mathrm{obs}},\Theta)}\big[\vec{s}\big]. \qquad (2.S17)
$$

The estimator (2.S17) can be efficiently computed by (i) approximating the full posterior distribution of the latents by using truncated posteriors (2.2) and by (ii) approximating the expectation w.r.t. the full posterior by applying truncated expectations (2.3).

Models With Binary-Continuous Latents and Continuous Observables. The upper derivation can be extended to be applicable to models with binary-continuous latents. Following the same line of reasoning, we again start with the posterior predictive distribution $p(\vec{y}^{\,\mathrm{est}}\mid\vec{y}^{\,\mathrm{obs}})$:

$$
\begin{aligned}
p(\vec{y}^{\,\mathrm{est}}\mid\vec{y}^{\,\mathrm{obs}},\Theta) &= \sum_{\vec{s}} \int\limits_{\vec{z}} p(\vec{y}^{\,\mathrm{est}},\vec{s},\vec{z}\mid\vec{y}^{\,\mathrm{obs}},\Theta)\,\mathrm{d}\vec{z} && (2.S18)\\
&= \sum_{\vec{s}} \int\limits_{\vec{z}} p(\vec{y}^{\,\mathrm{est}}\mid\vec{s},\vec{z},\Theta)\,p(\vec{s}\mid\vec{y}^{\,\mathrm{obs}},\Theta)\,p(\vec{z}\mid\vec{s},\vec{y}^{\,\mathrm{obs}},\Theta)\,\mathrm{d}\vec{z}. && (2.S19)
\end{aligned}
$$

We then follow the steps from Eqs. (2.S13) to (2.S16), i.e. we take expectations w.r.t. the posterior predictive distribution (2.S19):

$$
\begin{aligned}
\mathbb{E}_{p(\vec{y}^{\,\mathrm{est}}\,|\,\vec{y}^{\,\mathrm{obs}},\Theta)}\big[y_d^{\mathrm{est}}\big] &= \int\limits_{y_d^{\mathrm{est}}} y_d^{\mathrm{est}} \sum_{\vec{s}} \int\limits_{\vec{z}} p(y_d^{\mathrm{est}}\mid\vec{s},\vec{z},\Theta)\,p(\vec{s},\vec{z}\mid\vec{y}^{\,\mathrm{obs}},\Theta)\,\mathrm{d}\vec{z}\,\mathrm{d}y_d^{\mathrm{est}} && (2.S20)\\
&= \sum_{\vec{s}} \int\limits_{\vec{z}} \int\limits_{y_d^{\mathrm{est}}} y_d^{\mathrm{est}}\,p(y_d^{\mathrm{est}}\mid\vec{s},\vec{z},\Theta)\,\mathrm{d}y_d^{\mathrm{est}}\,p(\vec{s},\vec{z}\mid\vec{y}^{\,\mathrm{obs}},\Theta)\,\mathrm{d}\vec{z} && (2.S21)\\
&= \sum_{\vec{s}} \int\limits_{\vec{z}} \mathbb{E}_{p(y_d^{\mathrm{est}}\,|\,\vec{s},\vec{z},\Theta)}\big[y_d^{\mathrm{est}}\big]\,p(\vec{s},\vec{z}\mid\vec{y}^{\,\mathrm{obs}},\Theta)\,\mathrm{d}\vec{z} && (2.S22)\\
&= \mathbb{E}_{p(\vec{s},\vec{z}\,|\,\vec{y}^{\,\mathrm{obs}},\Theta)}\Big[\mathbb{E}_{p(y_d^{\mathrm{est}}\,|\,\vec{s},\vec{z},\Theta)}\big[y_d^{\mathrm{est}}\big]\Big]. && (2.S23)
\end{aligned}
$$

For the example of the SSSC model (2.12)–(2.13), the inner expectation in Eq. (2.S23) is given by $\mathbb{E}_{p(y_d\,|\,\vec{s},\vec{z},\Theta)}\big[y_d\big] = \vec{W}_d\,(\vec{s}\odot\vec{z})$ s.t. Eq. (2.S23) takes for the SSSC model the following form:

$$
\mathbb{E}_{p(\vec{y}^{\,\mathrm{est}}\,|\,\vec{y}^{\,\mathrm{obs}},\Theta)}\big[y_d^{\mathrm{est}}\big] = \vec{W}_d\mathbb{E}_{p(\vec{s},\vec{z}\,|\,\vec{y}^{\,\mathrm{obs}},\Theta)}\big[\vec{s}\odot\vec{z}\big]. \qquad (2.S24)
$$

The estimator (2.S24) can be efficiently computed based on Eq. (2.S31) by approximating the expectation w.r.t. the binary posterior by using truncated expectations (2.3).

2.S4 M-step Update Equations

NOR. For completeness, we report the NOR update rules here:

$$\pi_h = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\big[s_h\big], \qquad W_{dh} = 1 + \frac{\sum_{n=1}^{N}(y_d^{(n)}-1)\mathbb{E}_{q^{(n)}}\big[D_{dh}(\vec{s})\big]}{\sum_{n=1}^{N}\mathbb{E}_{q^{(n)}}\big[C_{dh}(\vec{s})\big]} \tag{2.S25}$$

where

$$D_{dh}(\vec{s}) := \frac{\widetilde{W}_{dh}(\vec{s})s_h}{N_d(\vec{s})(1-N_d(\vec{s}))},\ C_{dh}(\vec{s}) := \widetilde{W}_{dh}(\vec{s})D_{dh}(\vec{s}),\ \widetilde{W}_{dh}(\vec{s}) := \prod_{h'\neq h}(1-W_{dh'}s_{h'}). \tag{2.S26}$$

The update equations for the weights W_{dh} do not allow for a closed-form solution. We instead employ a fixed-point equation whose fixed point is the exact solution of the maximization step. We exploit the fact that in practice one single evaluation of Eq. (2.S26) is enough to (noisily, not optimally) move towards convergence to efficiently improve on the parameters W_{dh}.

BSC. As for NOR, we report the explicit forms of the M-step update rules for the BSC model here for completeness (compare, e.g., Henniges et al., 2010):

$$\begin{gathered}\pi = \frac{1}{NH}\sum_{n=1}^{N}\sum_{h=1}^{H}\mathbb{E}_{q^{(n)}}\big[s_h\big], \qquad \sigma^2 = \frac{1}{ND}\sum_{n=1}^{N}\mathbb{E}_{q^{(n)}}\Big[||\,\vec{y}^{\,(n)} - W\vec{s}\,||^2\Big], \\ W = \left(\sum_{n=1}^{N}\vec{y}^{\,(n)}\mathbb{E}_{q^{(n)}}\left[\vec{s}\right]^T\right)\left(\sum_{n=1}^{N}\mathbb{E}_{q^{(n)}}\left[\vec{s}\vec{s}^T\right]\right)^{-1}.\end{gathered} \tag{2.S27}$$

SSSC. We report the final expressions of the SSSC M-step update equations below. The derivations can be found in Sheikh et al. (2014).

$$\begin{gathered}W = \frac{\sum_{n=1}^{N}\vec{y}^{\,(n)}\mathbb{E}_{q^{(n)}}\big[\vec{s}\odot\vec{z}\,\big]^{\mathrm{T}}}{\sum_{n=1}^{N}\mathbb{E}_{q^{(n)}}\big[\,(\vec{s}\odot\vec{z}\,)\,(\vec{s}\odot\vec{z}\,)^{\mathrm{T}}\,\big]}, \qquad \vec{\pi} = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{q^{(n)}}\big[\vec{s}\,\big], \qquad \vec{\mu} = \frac{\sum_{n=1}^{N}\mathbb{E}_{q^{(n)}}\big[\vec{s}\odot\vec{z}\,\big]}{\sum_{n=1}^{N}\mathbb{E}_{q^{(n)}}\big[\vec{s}\,\big]}, \\ \Psi = \sum_{n=1}^{N}\Big[\mathbb{E}_{q^{(n)}}\big[\,(\vec{s}\odot\vec{z}\,)\,(\vec{s}\odot\vec{z}\,)^{\mathrm{T}}\,\big] - \mathbb{E}_{q^{(n)}}\big[\vec{s}\vec{s}^{\mathrm{T}}\big]\odot\vec{\mu}\vec{\mu}^{\mathrm{T}}\Big]\odot\left(\sum_{n=1}^{N}\Big[\mathbb{E}_{q^{(n)}}\big[\vec{s}\vec{s}^{\mathrm{T}}\big]\Big]\right)^{-1}, \\ \sigma^2 = \frac{1}{ND}\,\mathrm{Tr}\left(\sum_{n=1}^{N}\Big[\vec{y}^{\,(n)}(\vec{y}^{\,(n)})^{\mathrm{T}} - W\Big[\mathbb{E}_{q^{(n)}}\big[\vec{s}\odot\vec{z}\,\big]\mathbb{E}_{q^{(n)}}\big[\vec{s}\odot\vec{z}\,\big]^{\mathrm{T}}\Big]W^{\mathrm{T}}\Big]\right).\end{gathered} \tag{2.S28}$$

As defined in Eq. (2.15), the expectations in Eq. (2.S28) are taken w.r.t. the full binary-continuous latent space. Importantly for applying EVO, all these expectation values can be reformulated as expectations w.r.t. the posterior over the binary latent space:

$$\mathbb{E}_{q^{(n)}}\big[\vec{s}\,\big] = \sum_{\vec{s}} q^{(n)}(\vec{s};\Theta)\vec{s}, \tag{2.S29}$$

$$\mathbb{E}_{q^{(n)}}\big[\vec{s}\vec{s}^{\mathrm{T}}\big] = \sum_{\vec{s}} q^{(n)}(\vec{s};\Theta)\vec{s}\vec{s}^{\mathrm{T}}, \tag{2.S30}$$

$$\mathbb{E}_{q^{(n)}}\big[\vec{s}\odot\vec{z}\,\big] = \sum_{\vec{s}} q^{(n)}(\vec{s};\Theta)\vec{\kappa}_{\vec{s}}^{(n)}, \tag{2.S31}$$

$$\mathbb{E}_{q^{(n)}}\big[\,(\vec{s}\odot\vec{z}\,)\,(\vec{s}\odot\vec{z}\,)^{\mathrm{T}}\,\big] = \sum_{\vec{s}} q^{(n)}(\vec{s};\Theta)(\Lambda_{\vec{s}} + \vec{\kappa}_{\vec{s}}^{(n)}(\vec{\kappa}_{\vec{s}}^{(n)})^{\mathrm{T}}), \tag{2.S32}$$

where $q^{(n)}(\vec{s};\Theta)$ is (for the exact EM solution) the binary posterior given by

$$p\,(\vec{s}\mid\vec{y},\Theta) = \frac{\mathcal{B}(\vec{s};\vec{\pi})\;\mathcal{N}(\vec{y};\,\tilde{W}_{\vec{s}}\,\vec{\mu},C_{\vec{s}})}{\sum_{\vec{s}\,'}\mathcal{B}(\vec{s}\,';\vec{\pi})\;\mathcal{N}(\vec{y};\,\tilde{W}_{\vec{s}\,'}\,\vec{\mu},C_{\vec{s}\,'})}. \tag{2.S33}$$

In the above equations $\Lambda_{\vec{s}} = (\sigma^{-2}\tilde{W}_{\vec{s}}^{\mathrm{T}}\tilde{W}_{\vec{s}} + \Psi_{\vec{s}}^{-1})^{-1}$ and $\vec{\kappa}_{\vec{s}}^{(n)} = (\vec{s}\odot\vec{\mu}) + \sigma^{-2}\Lambda_{\vec{s}}\tilde{W}_{\vec{s}}^{\mathrm{T}}(\vec{y}^{\,(n)} - \tilde{W}_{\vec{s}}\vec{\mu})$. $\sum_{\vec{s}}$ denotes a summation over all binary vectors $\vec{s}$. Based on the reformulations, the expectations in Eqs. (2.S29) to (2.S32) can be approximated using truncated expectations (2.3).

2.S5 Evaluation Criteria for Image Restoration

A standard metric for the evaluation of image restoration (IR) methods is the peak-signal-to-noise ratio (PSNR; compare Sec. 2.4). In the context of IR benchmarks (e.g., denoising, inpainting, image super resolution), PSNR values are informative about the root-mean-square error between the restored image calculated by a specific IR algorithm and the respective clean target image. In addition to the PSNR measure, there are other criteria that can be used to compare IR approaches. Some criteria will be discussed in the following.

Clean Data. Supervised learning-based IR methods such as the denoising and inpainting methods of category DE6 and respectively IN6 in Tab. 2.2 require external data sets with clean images for training. In contrast, sparse coding and dictionary learning approaches such as the methods from category DE1 and IN1 (including the generative model algorithms EBSC and ES3C) are trained exclusively on the corrupted data that they aim to restore. The ability of learning solely from corrupted data is very valuable for scenarios in which clean data is not available or difficult to generate. Besides, the internal statistics of a test image were observed to be often more predictive than statistics learned from external data sets (compare Shocher et al., 2018). For deep neural networks, which often do require external clean training data, recent work seeks to provide methods to also allow them to be trained on noisy data alone (e.g. Lehtinen et al., 2018; Krull et al., 2019a; Ulyanov et al., 2018). Further related work proposes methods for ‘zero-shot’ super-resolution which can be trained exclusively on the corrupted test data (Shocher et al., 2018).

A-Priori Information and Robustness. For denoising, it is frequently assumed that a-priori knowledge of the ground-truth noise level is available. For instace KSVD, BM3D, WNNM, cKSVD or LSSC treat the noise level as input parameter of the algorithm. Other approaches that are trained using pairs of noisy/clean examples are typically optimized either for a single noise level (e.g., MLP, IRCNN, DnCNN-S, TNRD) or for several noise levels (e.g., MemNet, DnCNN-B, FFDNet, DPDNN, BDGAN). The denoising performance of noise-level-optimized approaches may deteriorate significantly if the algorithm is not provided with the appropriate (ground-truth) noise level of the test image (compare, e.g., Burger et al., 2012; Chaudhury and Roy, 2017; Zhang et al., 2018).

Similarly, for inpainting, it can be observed that certain methods exploit significant amounts of a-priori information, for example considering the text removal benchmark (Fig. 2.7 F): For NLRMRF and IRCNN, the training data is generated by overlaying specific text patterns on clean images. For IRCNN, several distinct font styles and font sizes are used (Chaudhury and Roy, 2017); for NLRMRF, training examples are generated using the identical text pattern of the test image (Sun and Tappen, 2011). In contrast, the generative model algorithms EBSC and ES3C require neither task-specific external training data sets (such as IRCNN or NLRMRF) nor do they require a-priori knowledge of the noise level (in contrast to, e.g., BM3D, MLP, IRCNN, FFDNET, DnCNN; compare Tab. 2.2). EBSC and ES3C learn the noise level parameter (which is part of the generative model) from the data in an unsupervised way.

Context Information. Another criterion for the evaluation of IR methods is the amount of context information that a particular algorithm exploits. The amount of context information is primarily determined by the patch size of the image segmentation. Increasing the effective patch size was reported to be beneficial for the performance of IR algorithms (compare Burger et al., 2012; Zhang et al., 2017). Zhang et al. reported that the effective patch size used by denoising methods can be found to vary greatly between different approaches (e.g., 36×36 used by EPLL, 50×50 by DnCNN-B, 70×70 by FFDNet, 361×361 by WNNM; numbers taken from Zhang et al., 2017, 2018). The denoising experiments with EBSC and ES3C were conducted using patches that did not exceed an effective size of 23×23 (compare Tab. 2.S2 in Sec. 2.S6.2), which is considerably smaller than the numbers reported by Zhang et al..

Stochasticity in the Acquisition of Test Data. For the denoising benchmarks considered in Sec. 2.4.2, there is stochastic variation in the test data due to the fact that for a given AWG noise level σ and for a given image, different realizations of the noise result in different noisy images. PSNR values can consequently vary even if the applied algorithm is fully deterministic. As all images investigated here are at least of size 256×256, variations due to different noise realizations are commonly taken as negligible for comparison (see, e.g., Mairal et al., 2009). The denoising results of EBSC and ES3C reported in Figs. 2.5 and 2.6 were obtained by executing the algorithm three times on each image using different noise realizations in each run. Observed PSNR standard deviations were smaller or equal 0.06 dB (compare Tab. 2.S1). For the inpainting experiments with randomly missing values (Fig. 2.7 A-E), we also performed three runs for each image using a different realization of missing values in each run. In these experiments, we observed slightly higher PSNR variations (standard deviations ranged from 0.02 to 0.18 dB). The different realizations of the noise (or of the missing values) employed for each execution of the algorithm might be relevant to explain the PSNR variations that we observed. Additionally, stochasticity in the EBSC and ES3C algorithms themselves cannot be excluded as a further factor.

Stochasticity in the Algorithm. The output of stochastic algorithms can differ from run to run even if the input remains constant. For learning algorithms, different PSNR values usually correspond to different optima of the learning objective which are obtained due to different initializations and/or due to stochasticity during learning. For stochastic algorithms, it is therefore the question which PSNR value shall be used for comparison. Two of the algorithms that we used for comparison in Figs. 2.5 to 2.7 report averages over different runs (KSVD and LSSC). All other algorithms do report a single PSNR value per denoising/inpainting setting without values for standard deviations. A single PSNR value is (for one realization of the corrupted image) naturally obtained for deterministic algorithms (e.g., BM3D, WNNM, NL). For the stochastic algorithms, a single PSNR may mean that either (A) just one run of the learning algorithm was performed, that (B) the best PSNR of all runs was reported, or that (C) one run of the learning algorithm was selected via another criterion. For DNN algorithms, for instance, the DNN with the lowest training or validation error could be selected for denoising or inpainting. Or for sparse coding algorithms, the model parameters with the highest (approximate) likelihood could be selected for denoising or inpainting. As the contributions in Figs. 2.5 to 2.7 which state just one PSNR value do not give details on how it was obtained from potentially several runs, it may be assumed that the best performing runs were reported (or sometimes the only run, e.g., in cases of a DNN requiring several days or weeks for training, compare Burger et al., 2012; Chaudhury and Roy, 2017; Jain and Seung, 2009; Zhang et al., 2018). Stating the best run is instructive as it shows how well an algorithm can perform under best conditions. For comparability, it should be detailed how the single PSNR value was selected, however (or if average and best values are essentially identical).

For practical applications, it is desirable to be able to select the best of several runs based on a criterion that can be evaluated without ground-truth knowledge. Our experimental data shows that for EBSC and ES3C the best denoising and inpainting performance in terms of PSNR cannot reliably be determined based on the learning objective, namely the variational bound (which is computable without ground-truth knowledge): The PSNR values of the runs with the highest bound may be smaller than the highest PSNR values observed in all runs. In a scenario with fixed noise realization, we observed the variational bound to be instructive about the PSNR value though (Fig. 2.4).

2.S6 Details on Numerical Experiments

We provide more details on used soft- and hardware, more details on the conducted experiments, and we provide control experiments for different algorithms.

2.S6.1 Soft- and Hardware

We implemented EBSC and ES3C in Python using MPI-based parallelization for execution on multiple processors. Model parameter updates were efficiently computed by distributing data points to multiple processors and performing calculations for batches of data points in parallel (compare Sec. 2.S4). Small scale experiments such as the bars tests described in

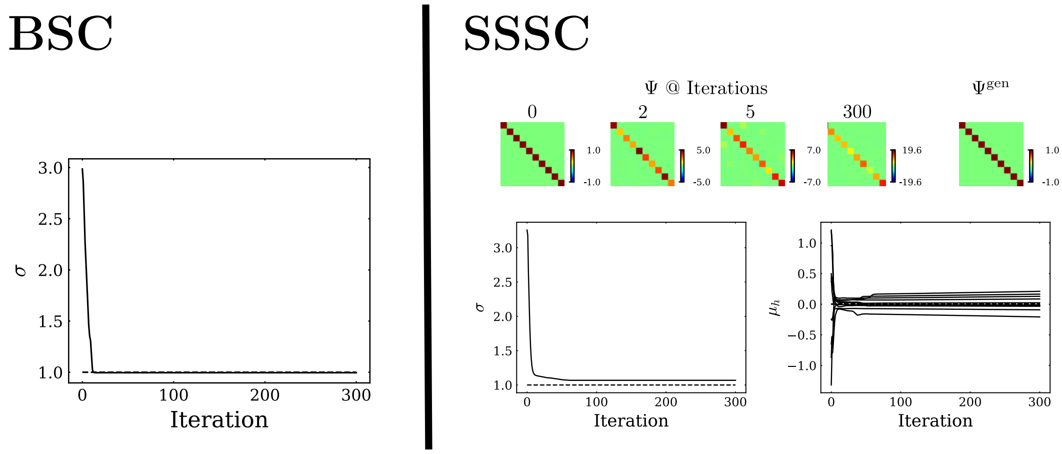


Figure 2.S1: BSC and SSSC model parameters learned from artificial data using EVO (see the description of the experiment in Sec. 2.3.2 for more details).

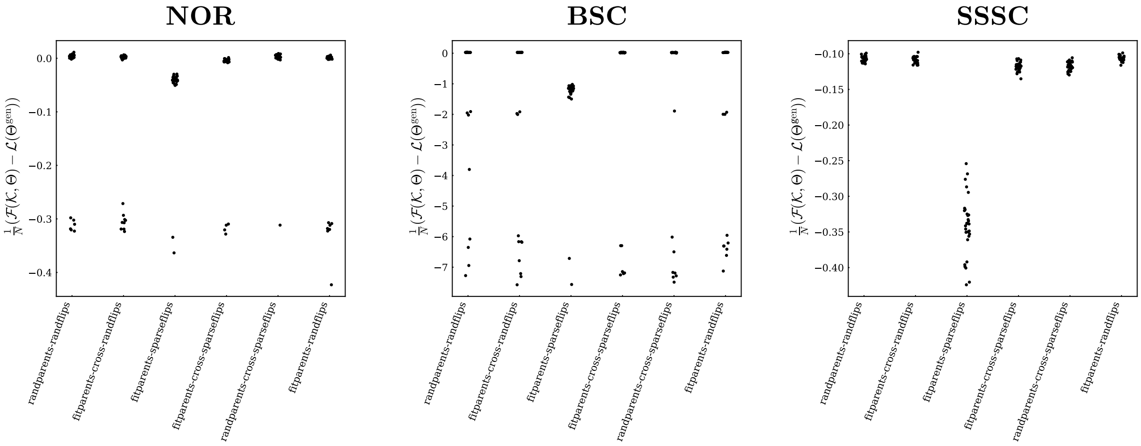


Figure 2.S2: Final free energies obtained by different EA configurations over 30 runs of EVO for NOR, BSC and SSSC models on 5×5 bars images (see the description of the experiment in Sec. 2.3.2 for more details).

Sec. 2.3.2 were performed on standalone workstations with four-core CPUs; runtimes were in the range of minutes. For most of the large-scale experiments described in Secs. 2.3.3 and 2.4, we used the HPC cluster CARL of the University of Oldenburg to execute our algorithms. The cluster is equipped with several hundred compute nodes with in total several thousands of CPU cores (Intel Xeon E5-2650 v4 12C, E5-2667 v4 8C and E7-8891 v4 10c). For each of the simulations, potentially different compute resources were employed (different numbers of CPU cores, different CPU types, different numbers of compute nodes, different numbers of employed cores per node), and hence runtimes were found to vary greatly between the simulations. For example for the denoising experiments with EBSC and ES3C on the “House” image (Fig. 2.5), the algorithms were executed using between 100 and 640 CPU cores, and runtimes ranged approximatly between six and one hundred hours for one run. Images larger than “House” required still longer runtimes or more CPU cores (we went up to a few thousand CPU cores on HPC clusters).

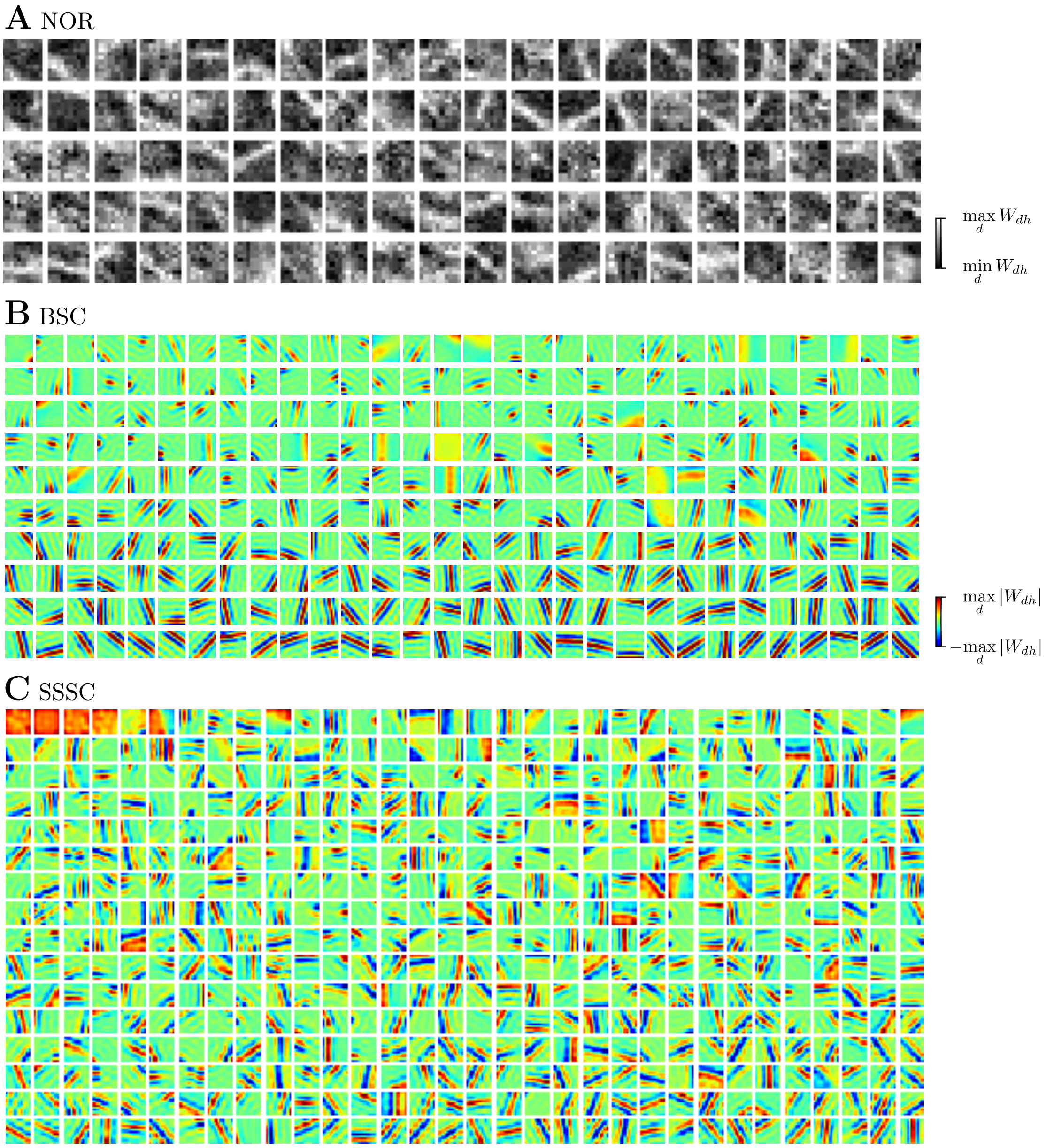


Figure 2.S3: Dictionaries learned from natural image patches using NOR, BSC and SSSC models (see Sec. 2.3.3). For NOR, we considered raw image patches and trained a model with $H = 100$ components. For BSC and SSSC, image patches were preprocessed using a whitening procedure. For BSC and SSSC, $H = 300$ and $H = 512$ generative fields were learned, respectively. In **B** and **C**, the fields are ordered according to their activation, starting with the fields corresponding to the most active hidden units.

A Denoising (Comparison of generative models and approximate inference under controlled conditions; Section 2.4.2.2)

Run	House $\sigma = 50$ ($H = 64$, $D = 8 \times 8$)			House $\sigma = 50$ ($H = 256$, $D = 8 \times 8$)			House $\sigma = 50$ ($H = 512$, $D = 12 \times 12$)		
	GSVAE$^{\text{lin}}$	EBSC	ES3C	GSVAE$^{\text{lin}}$	EBSC	ES3C	GSVAE$^{\text{lin}}$	EBSC	ES3C
1	19.14	28.30	28.87	18.93	28.29	28.89	18.98	28.35	29.88
2	19.16	28.04	28.74	18.92	28.27	28.86	18.98	28.29	29.85
3	19.14	27.99	28.72	18.89	28.22	28.79	18.91	28.16	29.74
∅	19.15	28.11	28.78	18.92	28.26	28.85	18.95	28.27	29.83

B Denoising (General comparison of denoising approaches; Section 2.4.2.3)

Run	House $\sigma = 15$			House $\sigma = 25$			House $\sigma = 50$			Barbara $\sigma = 25$			Lena $\sigma = 25$			Peppers $\sigma = 25$		
	GSVAE$^{\text{lin}}$	EBSC	ES3C	GSVAE$^{\text{lin}}$	EBSC	ES3C	GSVAE$^{\text{lin}}$	EBSC	ES3C	GSVAE$^{\text{lin}}$	EBSC	ES3C	GSVAE$^{\text{lin}}$	EBSC	ES3C	GSVAE$^{\text{lin}}$	EBSC	ES3C
1	29.38	33.70	34.86	28.62	32.39	33.16	25.88	29.03	29.88	24.97	28.33	30.17	27.93	30.91	31.90	25.52	29.01	30.35
2	29.37	33.65	34.93	28.06	32.47	33.20	25.68	28.91	29.85	25.21	28.39	30.14	28.10	30.90	31.94	25.54	28.87	30.22
3	29.41	33.63	34.93	28.29	32.34	33.09	25.62	29.01	29.74	25.19	28.42	30.24	28.00	30.84	32.01	25.52	28.90	30.26
∅	29.39	33.66	34.90	28.32	32.40	33.15	25.73	28.98	29.83	25.12	28.38	30.19	28.01	30.88	31.95	25.53	28.93	30.27

C Inpainting (Section 2.4.3)

Run	Barbara	Cameraman	Lena	House		Castle	
	50% missing	50% missing	50% missing	50% missing	80% missing	50% missing	80% missing
1	35.39	30.91	37.57	39.55	32.99	38.14	29.65
2	35.51	30.46	37.47	39.70	33.21	38.39	29.68
3	35.42	30.70	37.58	39.54	32.99	38.14	29.64
∅	35.44	30.69	37.54	39.59	33.06	38.23	29.66

Table 2.S1: PSNR values (in dB) measured for GSVAE$^{\text{lin}}$, EBSC and ES3C in the denoising and inpainting experiments described in Secs. 2.4.2.2, 2.4.2.3 and 2.4.3.

For the experiments with GSVAE, the code provided by the original publication (Jang, 2016) can execute optimization on GPU cores as is customary for deep models. When executing, for example, the CIFAR-10 experiment (Fig. 2.S4) on a single NVIDIA Tesla V100 16GB GPU, we observed runtimes of GSVAE$^{\text{lin}}$ on the order of a few seconds per iteration (on average approximately 10s); for the CIFAR-10 experiment, we observed GSVAE$^{\text{lin}}$ to converge within approximately 150 iterations. In comparison, to train EBSC on CIFAR-10, we executed our implementation in parallel on 768 CPU cores (Intel Xeon Platinum 9242) and performed 750 iterations of the algorithm. At the final iteration, the value of the lower bound was still slightly increasing, however not significantly anymore; the runtime per iteration was on the order of a few seconds (on average approximately 9s). In summary, GSVAE execution is more efficient. Comparison is difficult, however, because of the different setting. The more standard GSVAE approach uses conventional deep learning tools that can be expected to be well-optimized, while the distribution of EVO optimization across many CPUs (and cores) generates communication overhead, and the efficiency of implementation components can presumably be further enhanced.

2.S6.2 Hyperparameters

Table 2.S2 lists the hyperparameters employed in the numerical experiments on verification (Sec. 2.3.2), scalability (Sec. 2.3.3), denoising (Sec. 2.4.2) and inpainting (Sec. 2.4.3). EVO hyperparameters (S, N_p, N_m, N_g) were chosen s.t. they provided a reasonable trade-off between the accuracy of the approximate inference scheme and the runtime of the algorithm. For GSVAE$^{\text{lin}}$, we used the same neural network architecture for the encoding model and the same hyperparameters as in the source code provided by the original publication (Jang, 2016): The encoder network contained two hidden layers with 512 and 256 hidden units, respectively. The activation function of all network units was the identity (i.e., no non-linearity). The initial annealing temperature, the annealing rate and the minimal annealing temperature of the Gumbel-Softmax distribution were set to 1.0, $3 \cdot 10^{-5}$ and 0.5, respectively. The initial learning rate of the Adam optimizer was 10^{-3}. The patch sizes and number of latents used for GSVAE$^{\text{lin}}$ in the denoising experiments are, together with the corresponding values used for EBSC and ES3C, listed in Tab. 2.S2 C–D.

A Verification and scalability (Sections 2.3.2 and 2.3.3)

Experiment	N	D	H	EA	S	N_p	N_m	N_g	Iterations
Bars Test NOR/BSC/SSSC	5,000	5×5	10	randparents-randflips	20	5	4	2	300
Bars Test NOR/BSC/SSSC	5,000	5×5	10	fitparents-cross-randflips	20	5	-	2	300
Bars Test NOR/BSC/SSSC	5,000	5×5	10	fitparents-sparseflips	20	5	4	2	300
Bars Test NOR/BSC/SSSC	5,000	5×5	10	fitparents-cross-sparseflips	20	5	-	2	300
Bars Test NOR/BSC/SSSC	5,000	5×5	10	randparents-cross-sparseflips	20	5	-	2	300
Bars Test NOR/BSC/SSSC	5,000	5×5	10	fitparents-randflips	20	5	4	2	300
van Hateren Images NOR	30,000	10×10	100	fitparents-cross-sparseflips	120	8	-	2	200
van Hateren Images BSC	100,000	16×16	300	fitparents-cross-sparseflips	200	10	-	4	4,000
van Hateren Images SSSC	100,000	12×12	512	fitparents-cross-sparseflips	60	6	-	2	2,000

B Denoising (Relation between PSNR measure and variational lower bound; Section 2.4.2.1)

Image	Size	Noise Level σ	D	H	EA	S	N_p	N_m	N_g	Iterations
House	256×256	50	8×8	256	fitparents-randflips	60	6	5	2	2,000

C Denoising (Comparison of generative models and approximate inference under controlled conditions; Section 2.4.2.2)

Image	Size	Noise Level σ	D	H	EBSC- and ES3C-specific						GSVAE$^{\text{lin}}$-specific
					EA	S	N_p	N_m	N_g	Iterations	Iterations
House	256×256	50	8×8	64	fitparents-randflips	60	60	1	1	4,000	100
House	256×256	50	8×8	256	fitparents-randflips	60	60	1	1	4,000	100
House	256×256	50	12×12	512	fitparents-randflips	60	60	1	1	4,000	100

D Denoising (General comparison of denoising approaches; Section 2.4.2.3)

Image	Size	Noise Level σ	EBSC-specific						ES3C-specific						EBSC- and ES3C-specific		GSVAE$^{\text{lin}}$-specific		
			D	H	S	N_p	N_m	N_g	D	H	S	N_p	N_m	N_g	EA	Iterations	D	H	Iterations
House	256×256	15	8×8	256	200	10	9	4	12×12	512	60	60	1	1	fitparents-randflips	4,000	16×16	512	100
House	256×256	25	8×8	256	200	10	9	4	12×12	512	60	60	1	1	fitparents-randflips	4,000	32×32	512	100
House	256×256	50	8×8	256	200	10	9	4	12×12	512	60	60	1	1	fitparents-randflips	4,000	32×32	512	100
Barbara	512×512	25	8×8	256	200	10	9	4	11×11	512	60	60	1	1	fitparents-randflips	3,000	16×16	512	100
Lena	512×512	25	8×8	256	200	10	9	4	11×11	512	60	60	1	1	fitparents-randflips	4,000	16×16	512	100
Peppers	256×256	25	8×8	256	200	10	9	4	10×10	800	40	30	1	1	fitparents-randflips	6,000	32×32	512	100

E Inpainting (Section 2.4.3)

Image	Size	Missing Data Ratio	D	H	EA	S	N_p	N_m	N_g	Iterations
Barbara	512×512	50%	12×12	512	fitparents-randflips	30	20	1	1	4,000
Cameraman	256×256	50%	12×12	512	fitparents-randflips	30	20	1	1	4,000
Lena	512×512	50%	12×12	512	fitparents-randflips	30	20	1	1	4,000
House	256×256	50%	12×12	512	fitparents-randflips	30	20	1	1	4,000
House	256×256	80%	15×15	512	fitparents-randflips	30	20	1	1	500
Castle	481×321	50%	7×7	900	fitparents-randflips	30	20	1	1	2,000
Castle	481×321	80%	7×7	900	fitparents-randflips	60	60	1	1	200
New Orleans	297×438	Text Mask	14×14	900	fitparents-randflips	60	60	1	1	3,000

Table 2.S2: Hyperparameters employed in the numerical experiments.

2.S6.3 Comparison to Other Generative Models – Details

For our experiments with the Gumbel-Softmax Variational Autoencoder (GSVAE), we leveraged the source code provided by the original publication (Jang, 2016). For GSVAE$^{\text{lin}}$, we chose a categorical distribution with just two categories ("0" or "1") which matches the Bernoulli prior of BSC. Also to match the BSC generative model, we used a shallow, linear decoder (with weights W and without bias terms) connected only to the category "1" units. The scalar variance σ^2 of the Gaussian was treated as a trainable parameter (optimized alongside the other parameters), and we used the same prior parameter π for all $h = 1, \ldots, H$ as for BSC. The decoding model of GSVAE thus becomes equivalent to the generative model of BSC in Eq. (2.11). As controls, we used different versions of GSVAE including versions with (i) deep decoders connected only to category "1" units (below and in Tab. 2.S3, we refer to this GSVAE version "Binary-Deep"), (ii) shallow decoders connected to both categories (referred to as "Categorical-Shallow"), and (iii) deep decoders connected to both categories (referred to as "Categorical-Deep"; below and in Tab. 2.S3, we refer to GSVAE$^{\text{lin}}$, i.e., the version used for the denoising benchmarks in Secs. 2.4.2.2 and 2.4.2.3,

	Run	EBSC	GSVAE			
			Binary-Shallow	Binary-Deep	Categorical-Shallow	Categorical-Deep
Barbara	1	43.72	36.78	37.43	37.79	36.78
	2	**43.85**	37.49	37.71	37.55	36.88
	3	43.61	38.33	37.49	37.96	37.75
	∅	43.73	37.53	37.54	37.77	37.14
CIFAR-10	1	2,027.55	2,325.95	1,919.39	170.74	2,376.67
	2	2,021.42	2,307.94	513.90	2,545.02	2,559.23
	3	2,019.11	2,301.24	2,380.29	**2,618.58**	2,544.10
	∅	2,022.70	2,311.71	1,604.53	1,778.11	2,493.33

Table 2.S3: Free energies (ELBOs) per datapoint obtained with EBSC and GSVAE on image patches of the noisy "Barbara" image ($\sigma = 25$) and on CIFAR-10 (see text for details). For "Barbara", the values listed denote free energies on the noisy image patches which make up the training data set; for CIFAR-10, the values listed denote free energies on the test data set. The highest free energy obtained for each benchmark is marked bold.

as the "Binary-Shallow" version). The deep decoders had two hidden layers with 256 and 512 units. For GSVAE$^{\text{lin}}$ and all controls, we used the same encoding model as in the original implementation (architecture and hyperparameters listed in Sec. 2.S6.2). Also, for consistency with the original implementation, we used identity activation functions for all GSVAE versions.

To confirm the functionality of the implementation used, we first applied GSVAE$^{\text{lin}}$ and the different control versions to two standard data sets: image patches of the "Barbara" image with AWG noise and images of whole objects from the CIFAR-10 data set. For "Barbara", we used a noise level of $\sigma = 25$, $D = 8 \times 8$ patches and $H = 256$; for CIFAR-10, the patch size was $D = 32 \times 32 \times 3$, and we used $H = 1{,}024$. The CIFAR-10 images had amplitudes in the range $[0, 1]$; accordingly, we also scaled the clean "Barbara" image (that we used to generate noisy image patches) and the noise level to the range $[0, 1]$. Tab. 2.S3 lists the values of the lower bound obtained in different runs of the algorithms for the two data sets. As can be observed, there are significant differences between different runs of the GSVAE algorithms, especially for CIFAR-10. On this data set, the best runs resulted in values of the lower bound in the range of 2,300 (obtained with the "Binary-Shallow" version of GSVAE) to 2,620 (obtained with the "Categorical-Shallow" version; see Tab. 2.S3, bottom four rows). Already the GSVAE versions with shallow decoder performed relatively well on this benchmark. For comparison, the highest log-likelihood values reported in Park et al. (2019a) for their VLAE approaches on CIFAR-10 are 2,392 for a shallow version and 2,687 for a deep version. The "Categorical-Shallow" GSVAE control improved on the "Binary-Shallow" version (GSVAE$^{\text{lin}}$); at the same time, the former version has twice as many weight parameters. Deep decoders also tended to improve performance on CIFAR-10 but not very significantly. The original GSVAE implementation used linear activation units. When we substituted linear activation by ReLU units, performance even tended to decrease (which is presumably the reason for the linear activations in the original publication (Jang et al., 2017)).

While performance of GSVAE in terms of lower bounds is relatively high on CIFAR-10, performance on image patches of the noisy "Barbara" image is relatively low. GSVAE$^{\text{lin}}$

and also all controls resulted in significantly lower values of the bound than values obtained by EBSC. On the other hand, EBSC values are significantly lower on average on CIFAR-10. On image patches of the noisy "Barbara" image, EBSC (and also other approaches such as MTMKL) are able to learn representations with generative fields (GFs) being much more aligned with actual image structures (Fig. 2.S4 B); and EBSC uses a sparse code to combine only few of the fields for reconstruction (on "Barbara", EBSC used approximately 1.5 out of 256 fields, on CIFAR-10 approximately 19 out of 1,024). GSVAE$^{\mathrm{lin}}$ uses the same generative model as EBSC, so we can directly compare the GFs (i.e., the columns of W). In contrast to EBSC, the GFs of GSVAE$^{\mathrm{lin}}$ are less interpretable and contain higher spatial frequencies (Fig. 2.S4 E). On average, GSVAE$^{\mathrm{lin}}$ used about half of these fields for reconstruction (i.e., 128 out of 256 fields on "Barbara"). GSVAE$^{\mathrm{lin}}$ also learned such fields and dense codes for CIFAR-10 (Fig. 2.S4F; approximately 512 out of 1,024 fields were used on average). On CIFAR-10, the dense codes of GSVAE$^{\mathrm{lin}}$ and of the controls seem comparably effective, while it seems more difficult to encode whole object images using sparse codes.

For GSVAE we can, of course, not exclude that settings can be found for which sparse codes and high PSNR values are achieved on "Barbara" and other denoising benchmarks. However, no such setting and competitive denoising performance has been reported in the literature for either GSVAE or other VAEs (to the knowledge of the authors; but compare Prakash et al., 2021b). For other data, sparse codes may not necessarily result in better performance (Tab. 2.S3). For standard denoising and especially for the 'zero-shot' setting which we focused on, good image patch models are required, however, and sparse codes are observed to be advantageous in this case.

Regarding the above discussion, note that other deep generative models *have* been applied to denoising and inpainting. Creswell and Bharath report reconstruction performance on AWG noise removal tasks; however in Creswell and Bharath (2018), (i) the considered noise levels are significantly smaller than the ones in Figs. 2.5 and 2.6, and (ii) data sets of whole objects such as Omniglot or CelebA are employed rather than the test images of Figs. 2.5 and 2.6. Related contributions on GANs frequently consider the task of semantic inpainting, i.e., the reconstruction of large areas in images in a semantically plausible way (see, e.g., Cao et al., 2019, for an overview). In these contributions, typically data sets such as CelebA, Street View House Numbers images or ImageNet are employed, and frequently the reconstruction of a single or a few (e.g., squared or rectangular) holes is considered as a benchmark (compare, e.g., Iizuka et al., 2017; Li et al., 2017; Pathak et al., 2016; Yang et al., 2017; Yeh et al., 2017). Yeh et al. also consider the task of restoring randomly missing values, however neither Yeh et al. (2017) nor the aforementioned publications report performance on the test images of Fig. 2.7. The benchmarks of denoising and inpainting we use (Figs. 2.5 to 2.7) are (A) very natural for EVO because of available benchmark results for mean field and sampling approaches, and (B) the benchmarks allow for comparison with state-of-the-art feed-forward DNNs (Figs. 2.5 and 2.6).

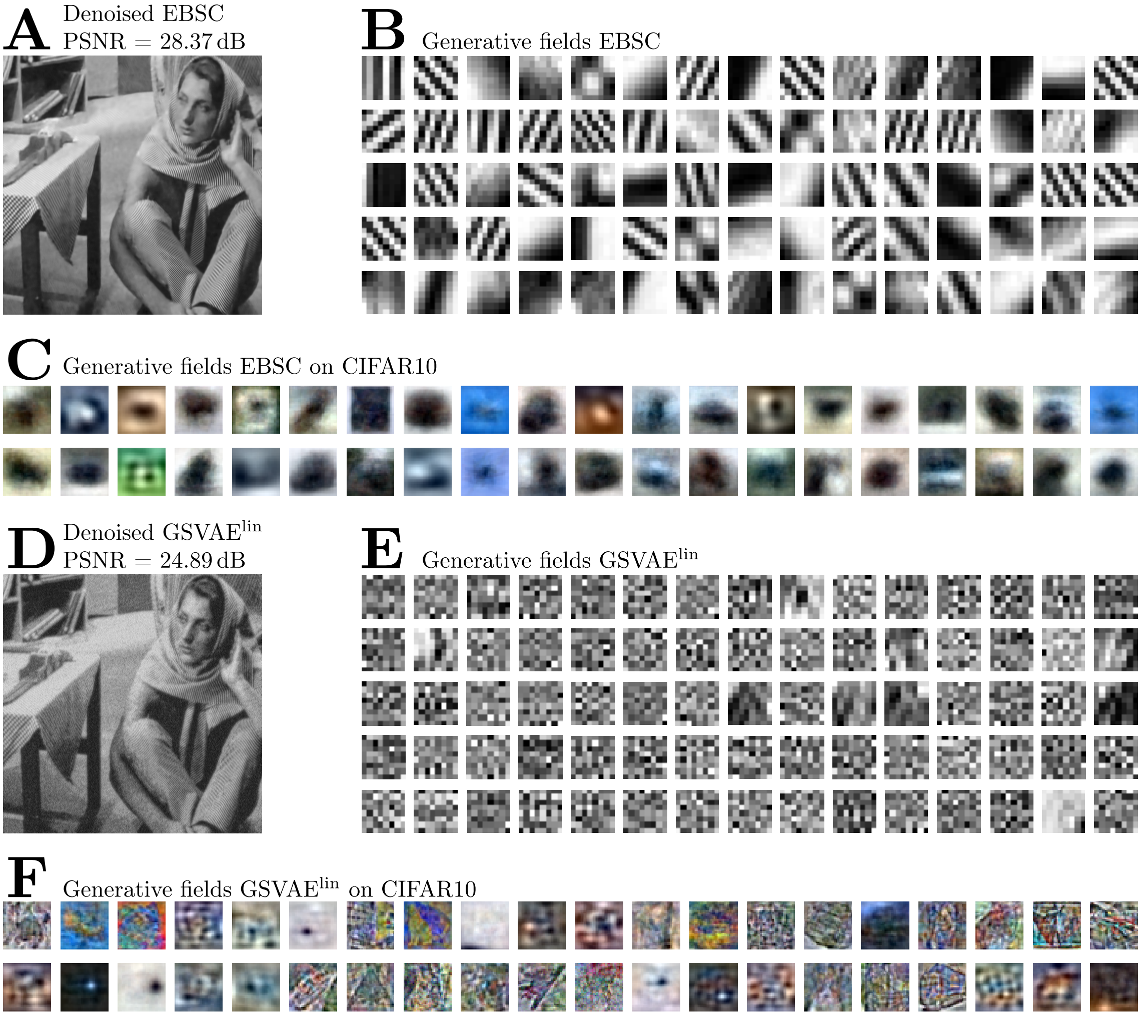


Figure 2.S4: Denoising and generative fields of EBSC and GSVAE (compare text and Tab. 2.S3). **A, D** Denoising results after EBSC and GSVAE$^{\text{lin}}$ were used for denoising (input image was "Barbara" with $\sigma = 25$; depicted are the results of the run with highest lower bound). **B, C** Selection of GFs of EBSC for the run with highest lower bound on "Barbara" and CIFAR-10 data, respectively. **E, F** Selection of GFs of GSVAE$^{\text{lin}}$ for the run with highest lower bound on "Barbara" and CIFAR-10 data, respectively.

Chapter 3

Zero-Shot Microscopy Image Enhancement

This chapter is in revision for publication as: Jakob Drefs*, Sebastian Salwig*, Jörg Lücke. Enhancing Microscopy Images of SARS-CoV-2 Infection Scenes: Applications of Novel Data-Driven Algorithms Operating at the High-Resolution Limit. *Joint first authorship.

Abstract. Microscopy images of infected tissues serve to diagnose a disease, and they can contribute to our understanding of infection processes. As a consequence of the corona pandemic, a large number of microscopy images of SARS-CoV-2 viruses and their interaction with infected tissues have been made available. In order to visualize the viruses, high resolution electron microscopy (EM) has been applied; and in order to visualize details of the viruses and infected tissues, EM recording devices had to operate at the limit of the currently available resolution. Recordings at such limits result in a high degree of noise which negatively effects both image interpretation by experts and further automated processing. However, the deteriorating effects of strong noise can be very much alleviated by novel machine learning (ML) algorithms for image enhancement which can operate in the high resolution limit of microscopy images. Because of the inherent noise at high resolution, a requirement for ML algorithms is their trainability on noisy images or on just one noisy image. Our main focus will be on two novel probabilistic such ML algorithms based on generative approaches, but we also survey other novel methods including recent neural networks which were adapted to operate on noisy images. The ML approaches are then applied to EM images of SARS-CoV-2 infection scenes to investigate their image enhancement capabilities. For further validation and evaluation, the approaches are, moreover, applied to fluorescence microscopy images for which ground-truth related comparisons are available. We find that the denoised images resulting from all applied approaches can reveal structural details that are difficult to identify in the raw noisy images. Furthermore, by using the focused-on probabilistic generative approaches, we show that details such as the SARS-CoV-2 spike protein can be enhanced still further by using non-standard higher-order statistics. Finally, we discuss the advantages and disadvantages of all here investigated approaches and point to future developments. In general, we argue that data-driven image enhancement algorithms represent tools which can decisively help in extracting knowledge from microscopy data, and thus can contribute to better understanding SARS-CoV-2 and other infections in the future.

Author Contributions. All authors designed the study, and preliminary numerical investigations on higher-order image reconstruction statistics were done by Jakob Drefs (JD). All authors contributed to the literature research. Sebastian Salwig (SS) performed the mathematical derivations related to one of the used generative models (the Gamma-Poisson mixtures, Secs. 3.S1.2 to 3.S1.3) and implemented that model. JD and SS performed the numerical experiments. JD and Jörg Lücke (JL) wrote the manuscript with contributions from SS. JD and SS prepared the figures and tables with input from JL. All authors discussed the results and their presentation and approved the final version of the manuscript.

Acknowledgments. This work was funded by the German Ministry of Research and Education (BMBF) under grant "Denoising and detail enhancement of SARS-CoV-2 EM recordings by probabilistic generative models". The grant was provided as an emergency top-up funding for COVID-19 research as part of the grant "Small data problems in digital pathology" (SPAplus; 05M20MOA, TP3). Furthermore, we acknowledge funding by the DFG project 352015383 (SFB 1330, B2) which provided source code to train generative models, and we acknowledge support by the Oldenburg High Performance Compute Cluster (CARL) and by the North German Supercomputing Alliance (HLRN) under grant nim00006.

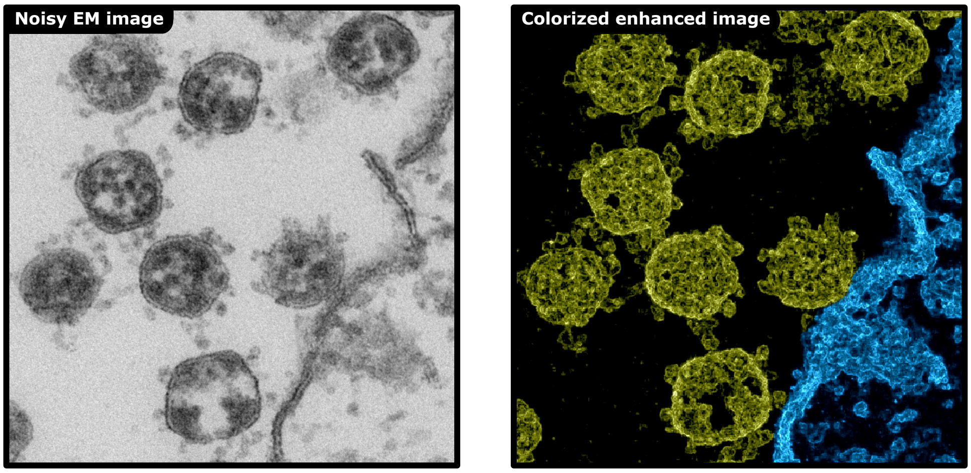


Figure 3.1: Example of enhancing an electron microscopy image of SARS-CoV-2 viruses in Vero cell cultures. Left: Original transmission electron microscopy of ultrathin plastic sections (Laue et al., 2021). Right: Detail enhancement of the original image by visualizing the pixel variances during the application of a probabilistic denoising approach (see Methods). The denoising algorithm uses a representation of image patches learned by a probabilistic generative model trained solely on the single, noisy original image ('zero-shot' learning). The final displayed image was obtained after contrast enhancement and colorization (structures that we manually identified as belonging to a cell were colored in blue (lower right area), the remainder was colorized in yellow).

3.1 Introduction

The SARS-CoV-2 virus which causes the COVID-19 disease is currently infecting a high percentage of humans world-wide, which has dramatically impacted society and economy. Electron microscopy (EM) imaging is an established method to visualize viruses and infection processes. Such images can contribute to our understanding of how a given virus infects a cell, how it binds to cells and different tissues, and how it spreads in the body (Ivanova et al., 2016; Lamers et al., 2020). In this context, it is of high importance to appropriately visualize fine details at the nanoscale. Otherwise, nanoscale structures other than SARS-CoV-2 viruses can be confused with the viruses leading to undesirable misinterpretations of infection scenes (Dittmayer et al., 2020). Transmission electron microscopy (TEM) allows for the visualization of fine details on the scale of a few nanometers. SARS-CoV-2 viruses with their characteristic spike proteins can in principle be identified at this scale. However, because of the low electron intensity at such high resolution, EM images contain significant amounts of noise (Fig. 3.1, left).

To address the problem of high noise, algorithms for noise removal are a very established tool, and they can significantly improve further image processing as well as the interpretation of images by humans. In the case of EM images at high resolution, denoising is particularly challenging, since typically neither noise-free images nor noise level information are available. Standard denoising algorithms such as BM3D (Dabov et al., 2007) or WNNM (Gu et al., 2014), which both rely on a-priori noise level information, can consequently not be applied directly. Similarly, the direct use of state-of-the-art neural networks (Dong et al., 2019; Tai et al., 2017; Tian et al., 2020; Zhang et al., 2017) is not possible as clean (i.e., non-noisy) images are required for training. For data-driven approaches, waiving the requirement of clean training images has been subject of current research: Evaluated on benchmarks with standard test images, Noise2Noise (N2N; Lehtinen et al., 2018), for instance, was shown to improve the performance of BM3D-based denoisers while trained on noisy data (and without requiring the noise level a-priori). A prerequisite of N2N is, however, that a training dataset with different noisy realizations of the same underlying clean image can be constructed. Such an in practice usually unrealistic assumption motivated follow-up work in the form of Noise2Void (N2V; Krull et al., 2019a) which dropped the requirement of paired noisy images as in N2N training. On standard macroscopic test images, N2V showed competitive denoising performance, yet peak-signal-to-noise ratios (PSNRs) reported by Krull et al. (2019a) were not on par with BM3D and N2N. Examples such as N2N or N2V highlight that direct applicability to noisy data has emerged as an increasingly important feature of denoising algorithms, and ideas exploited by these methods have consequently been taken up in various ways (Bepler et al., 2020; Prakash et al., 2021b; Quan et al., 2020a; Wang et al., 2020).

For images of SARS-CoV-2 viruses, a further goal has been the reconstruction of the virus or virus parts in 3D from EM data. On the most detailed level, work, e.g., by Wrapp et al. (2020) and Ke et al. (2020) has used large amounts of TEM images of SARS-CoV-2 particles in order to reconstruct the protein's 3D shape on a molecular level. It was thus possible to visualize a 3D model of the protein in different conditions which can help to better understand the protein binding process. On the next lower resolution level, work by Nanographics has released a 3D image of the SARS-CoV-2 virus using cryo-electron tomography (Nanographics, 2021; Yao et al., 2020). The approach is based on a large number of TEM images of the same virus recorded as the sample is tilted along an axis and merged to a 3D image using technology developed for computer tomography (Doerr, 2017).

In this work, we investigate methods that can be applied for microscopy image restoration under 'zero-shot' (Chen et al., 2020; Shocher et al., 2018; Soh et al., 2020) conditions, i.e., in a setting where only a single noisy image is available. This implies that for the application to SARS-CoV-2 TEM recordings, the considered approaches (A) can *not* leverage large numbers of different images of different spike proteins (as would be required for the methods by Wrapp et al. (2020) and Ke et al. (2020)), (B) they can not use many images of the same virus (as would be required for the approach of Nanographics (2021)), and (C) they can not assume availability of multiple noisy versions of a hypothetically 'clean' underlying image (as assumed, e.g., for N2N). The 'zero-shot' algorithms we here concretely investigate are: (i) BM3D as a standard (and not data-driven) baseline, (ii) N2V and Self2Self (S2S; Quan et al., 2020a) as recent deep neural networks (DNNs) that, respectively, use blind-spot and dropout strategies to learn a denoising mapping, (iii) DivNoising (DivN; Prakash et al., 2021b) as a recent deep generative model approach, and (iv) Evolutionary Spike-and-Slab Sparse Coding (ES3C; Drefs et al., 2022) and a novel variant of the recent Gamma-Poisson Mixture Model (GPMM) of Monk et al. (2018) representing two novel probabilistic generative approaches that we here for the first time apply to microscopy image data.

Applying BM3D in a 'zero-shot' setting requires additional methodology for blind noise level estimation. The alternative N2V approach uses a U-Net architecture, and with large such nets the approach can benefit from the availability of increasingly much (noisy) training data. Consequently, the here considered setting in which training data needs to be derived from a single image represents a challenging scenario for the algorithm. DivN relies on a noise model of the imaging device, i.e., an expression for the distribution of noisy pixels given clean image pixels. Prakash et al. (2021b) discuss different strategies for estimating such a model including strategies that make use of non-noisy data as well as an approach applicable without clean data. One of the former strategies relies on the availability of a sequence of noisy images recorded using the same field of view (i.e., static recording conditions; Krull et al., 2020a). These images, referred to as calibration data, are used to estimate a non-noisy image by averaging the image sequence. As an alternative for a scenario without calibration data, Prakash et al. discuss a strategy referred to as bootstrapping which leverages an external denoising algorithm for estimating a non-noisy image. In both the calibration and bootstrapping scenarios, the noise model is defined in terms of a Gaussian mixture model which uses a parameterization depending on both clean and noisy data (Prakash et al., 2020). Furthermore, Prakash et al. discuss a strategy referred to as fully unsupervised which uses a Gaussian noise model with a variance parameterized as linear function of the modeled clean pixel value (Prakash et al., 2021b). This approach involves a user-defined parameter that determines the lower limit of the estimated variance. The S2S approach, unlike N2V and DivN, is designed for a 'zero-shot' setting and has as such comparable requirements to ES3C and GPMM, i.e., S2S, ES3C and GPMM can all directly be applied to a single noisy image. A common property of the three approaches DivN, ES3C and GPMM, on the other hand, is that they are all based on probabilistic generative models. Generative models are very established tools with very competitive performance for a variety of image enhancement tasks including denoising, deblurring or inpainting (Papyan and Elad, 2016; Parameswaran et al., 2018; Titsias and Lázaro-Gredilla, 2011; Zoran and Weiss, 2011). Generative approaches have also successfully been used to decrease EM scan time and electron beam exposure (Ede and Beanland, 2020) using training settings for inpainting (a setting different from the here considered task). Furthermore, generative models are, usually, well suited for task settings with few data, and, typically, they can naturally be optimized on noisy data (Goodfellow et al., 2012; Sheikh et al., 2014; Titsias and Lázaro-Gredilla, 2011; Zhou et al., 2012). Such properties consequently suggest them as suitable tools for image enhancement applications as we investigate here.

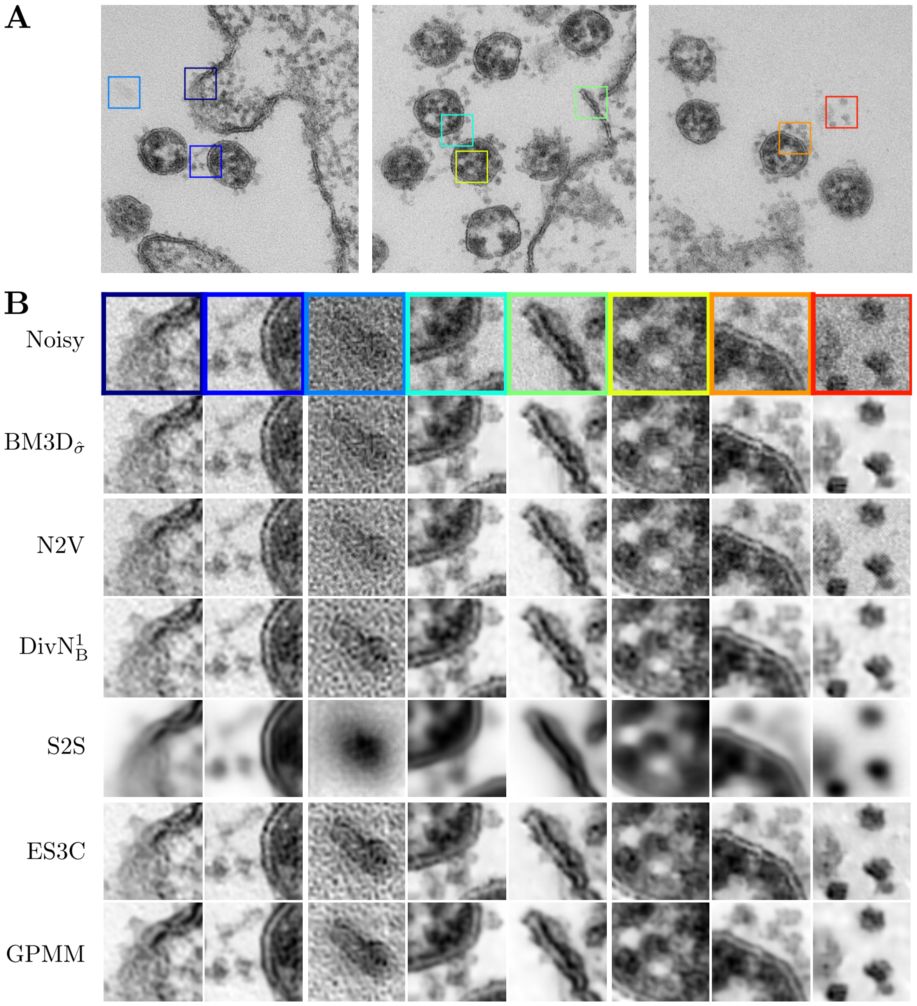
The 'zero-shot' setting we consider represents, as described, a challenging scenario for an enhancement algorithm; yet, data representations learned based on single observations such as a single noisy image can also be advantageous. It has recently been argued, for instance, that intrinsic image statistics contain higher quality information compared to statistics learned from large image corpora (Shocher et al., 2018). BM3D and its many extensions (Azzari and Foi, 2016; Ehret and Arias, 2021), for example, make relatively direct use of single image statistics. Generative approaches, on the other hand, first learn a probabilistic representation of image patches, which can subsequently be used for estimating the underlying non-noisy image. Conventionally, non-noisy images are reconstructed through the estimation of *mean* intensity values, i.e., based on the first moments of the modeled pixel distribution. During first-order image reconstruction, the learned representation can, however, also be used to estimate pixel reconstruction *variances*. Here, we argue that, alongside standard denoising, such higher-order statistical information can reveal more structural details in reconstructed images. Both, first-order and higher-order image reconstructions of 'zero-shot' approaches may consequently help medical experts in identifying important structural image details.

3.2 Results

3.2.1 Denoising TEM Images of SARS-CoV-2 Infected Cell Cultures

We considered publicly available TEM images of SARS-CoV-2 viruses in Vero cell cultures (Laue et al., 2021) and investigated ‘zero-shot’ denoising of nine selected, preprocessed test images (data-related details in Sec. 3.S2.1). To apply BM3D, N2V, S2S, and DivN to these images, we used the implementations that were made publicly available together with the official publications, and used the default hyperparameters of those implementations (for details and references see Sec. 3.S2.2). To estimate the noise level for BM3D, we applied the method of Chen et al. (2015a) which assumes additive white Gaussian noise. For DivN, we used the denoised images obtained with N2V for bootstrapping. For image denoising with ES3C and GPMM, we computed first-order estimations of pixel means (compare Methods and see Sec. 3.S2.2 for hyperparameters and initializations used). All investigated algorithms were applied individually to each test image. The results of the experiment are depicted in Fig. 3.2.

Qualitatively, BM3D, DivN, ES3C and GPMM result in similarly strong noise suppression while largely preserving image sharpness. S2S achieves still stronger noise suppression, however, the denoised images obtained with the algorithm appear severely blurred (Fig. 3.2 B). Due to the lack of ‘clean’ reference images, we could not quantify denoising performance in terms of the standard PSNR measure. Instead, we adopted objective metrics for noise suppression and image quality that could be evaluated based on a single image (Fig. 3.2 C). Concretely, we (i) assessed noise suppression based on estimated noise levels (using the method of Chen et al. (2015a); $\hat{\sigma}$ values in Fig. 3.2 C), (ii) performed a signal-to-noise quantification based on hand-labeled signal and background regions (using the method of Bepler et al. (2020); SNR values in Fig. 3.2 C), and (iii) carried out an image quality ranking based on spatial and spectral image statistics (Koho et al., 2016a). For the quality ranking, we considered four of the statistics discussed in the related publication (Koho et al., 2016a, also see Methods): coefficient of variation (denoted CV in Fig. 3.2 C; supposed to be instructive about ensemble or statistical noise (Yan et al., 2016)), inverse of power spectrum standard deviation (denoted invSTD; supposed to be instructive about non-noisiness (Koho et al., 2016a)), spatial entropy (denoted Entropy; supposed to be instructive about contrast (Koho et al., 2016a)), and bin mean (denoted MeanBin; supposed to be instructive about blur (Koho et al., 2016a)). Averaged over all test images, best SNR and invSTD scores are obtained by S2S. Effectiveness of S2S is also suggested by the $\hat{\sigma}$ values; at the same time, this measure evaluates the performance of S2S to be comparable to DivN, GPMM and ES3C. Comparing these quantitative results with a (non-expert) visual assessment of the qualitative results depicted in Fig. 3.2 B makes $\hat{\sigma}$, SNR and invSTD, for the test images considered, appear insightful with respect to the visual impression of noise suppression, but less instructive with respect to image sharpness. More closely aligned with the visual impression of image sharpness, in turn, appear the measures CV and MeanBin, which both assign the lowest ranking to S2S and best rankings to ES3C (CV) and BM3D (MeanBin).



C

Algorithm	$\Delta\hat{\sigma}\downarrow$	ΔSNR$\uparrow$	Quality Ranking			
			CV	invSTD	Entropy	MeanBin
BM3D$_{\hat{\sigma}}$	-18.76±5.31	10.74±4.63	4th	6th	6th	**1st**
N2V	-13.95±5.12	7.53±2.52	5th	5th	4th	2nd
DivN$^{1}_{\mathrm{B}}$	-21.24±4.28	11.62±2.76	2nd	3rd	3rd	4th
S2S	**-21.37±4.29**	**16.78±3.03**	6th	**1st**	2nd	6th
ES3C	-21.15±4.27	12.89±2.62	**1st**	4th	5th	3rd
GPMM	-21.21±4.29	14.72±2.02	3rd	2nd	**1st**	5th

Figure 3.2: Results for ‘zero-shot’ denoising TEM images of SARS-CoV-2 infected cell cultures. A: Three of the in total nine test images used. B: Qualitative comparison of denoised image parts showing virus and cell structures obtained with the investigated algorithms. BM3D is listed with subscript $\hat{\sigma}$ to indicate that the algorithm was applied in combination with an external method for noise level estimation. For DivN, we use the suffix $^{1}_{\mathrm{B}}$ to indicate that the approach was trained on a single noisy image using bootstrapping. C: Quantification of denoising performance using no-reference evaluation metrics. For $\hat{\sigma}$ and SNR, we report the delta between the reconstructed and the noisy image, averaged over all test images (up and down arrows indicate that performance improves as values increase and decrease, respectively). The results of the quality ranking were obtained by first ranking the investigated algorithms individually for each test image and then averaging the results across all test images and translating them to ranks again (the individual results for each test image are listed in Fig. 3.S1). Best performances in terms of average value of each measure are marked bold. See Sec. 3.2.1 for details.

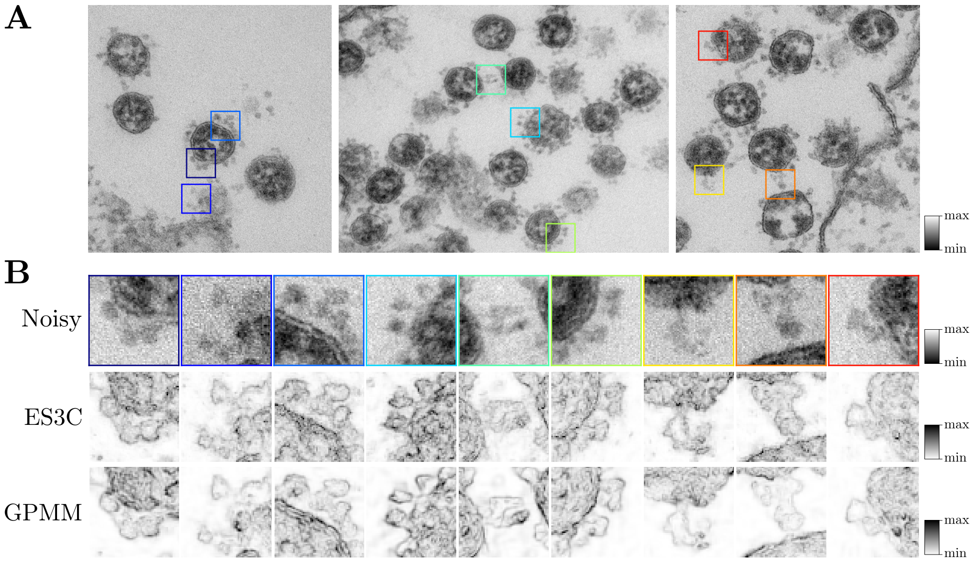


Figure 3.3: Visualizing variances during estimation of non-noisy image pixels of TEM images of SARS-CoV-2 infected cell cultures. A: Noisy TEM images. B: Qualitative comparison of variance reconstructions obtained with ES3C and GPMM for image parts showing spike proteins. Note that the bottom two rows are displayed with a reversed color map for improved visibility. See Sec. 3.2.2 for details.

3.2.2 Variances of Non-Noisy Pixel Estimations

The ES3C and GPMM approaches process images on a patch-by-patch basis: a non-noisy image pixel is computed based on a set of estimates obtained from image patches enclosing the given pixel (see Methods and Fig. 3.5). For the ES3C and GPMM results in Fig. 3.2, we computed non-noisy pixels by taking the average, i.e. the *mean* of sets of pixel estimates (Fig. 3.5, top center). If we, instead, compute the *variance* of each pixel's mean estimations (i.e., a higher-order reconstruction statistics; Fig. 3.5, top right), we obtain images as shown in Fig. 3.3. Displaying the variance of pixel estimations removes the direct information on matter density from the TEM image. Information about boundaries between high and low density matter is maintained because at boundaries image shapes vary more strongly than in volumes with uniform matter distribution. Visualizing variances (instead of means) of pixel estimations based on a sufficiently sophisticated generative model of image patches as in Fig. 3.3 allows for a much sharper representation of edge-like structures. This may help, for example, to measure the size of the SARS-CoV-2 virus and its spike proteins. Comparing the second order reconstructions obtained with ES3C and GPMM in Fig. 3.3 shows that the generative model itself changes the appearance of the higher-order reconstruction and (to a lesser extend) also the appearance of the conventionally denoised image (Fig. 3.2). Note that the introductory Fig. 3.1 visualizes variances of pixel estimations similarly to Fig. 3.3, but it uses additional manual coloring (details in caption of Fig. 3.1). Further visualizations of variances of non-noisy pixel estimations are provided in Sec. 3.S2.3.

3.2.3 Zero-Shot Denoising Fluorescence Microscopy Images

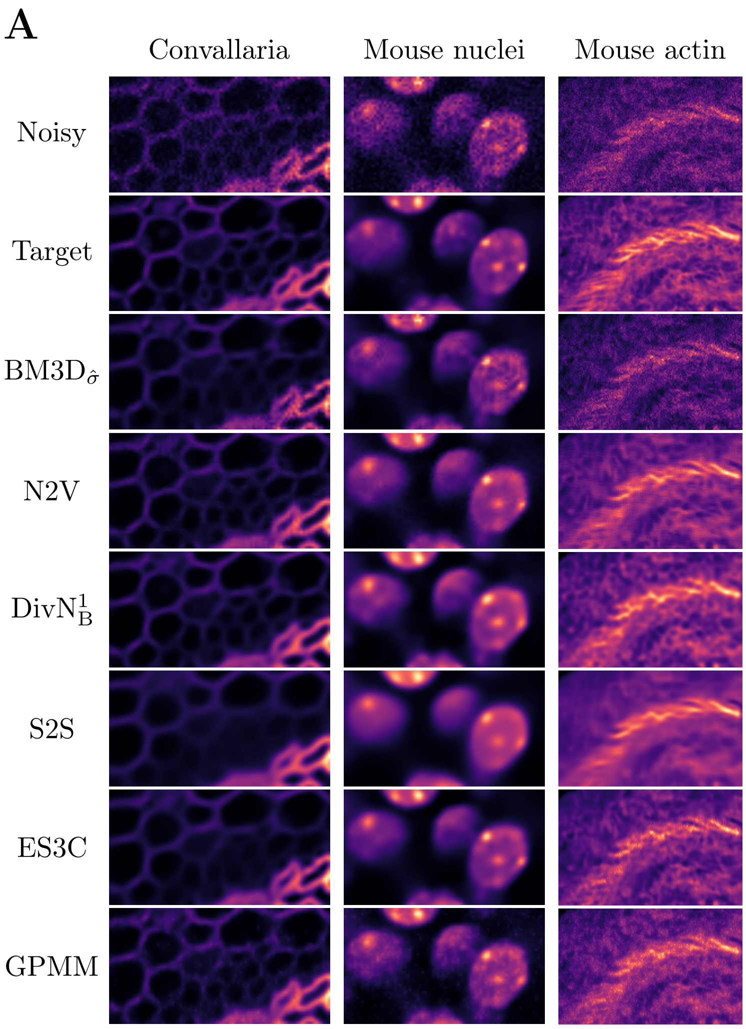
Thorough benchmarking of denoising performance for the TEM images of SARS-CoV-2 infected cell cultures considered in Fig. 3.2 proved difficult due to the unavailability of clean reference data, as discussed above. For the no-reference objective measures that we considered in Fig. 3.2 C, it holds that they come with their own sets of hyperparameters: the blind noise level estimator, for instance, operates on a patch-by-patch basis (similar to, e.g., ES3C and GPMM) and hence requires to choose a patch size (we used the defaults of the publicly available implementation, details in Sec. 3.S2.2). Similarly, the signal-to-noise quantification relies on manually labeled signal-background regions whose positions and sizes obviously impact the resulting SNRs. Likewise, the image quality ranking method allows to set a threshold to adjust the sensitivity of internally calculated measures, which, in turn, we observed to impact the resulting ranking. While we observed similarities between particular objective scores and our (non-expert) subjective visual assessment, the data of Fig. 3.2 did not seem well suited to us in order to objectively compare the denoising performance of the investigated algorithms. Consequently, we performed further evaluations using more established data sets.

Concretely, we adopted the recent microscopy image denoising benchmark discussed by Prakash et al. (2021b) and applied ES3C and GPMM to the publicly available fluorescence microscopy images FU-PN2V Convallaria, FU-PN2V Mouse actin, and FU-PN2V Mouse nuclei (see Sec. 3.S2.1 for references). Each of these datasets contains a sequence of different noisy realizations of a static scene for which a clean reference can be estimated by averaging all images in the sequence (Prakash et al., 2020). To consider a ‘zero-shot’ setting, we adopted the procedure discussed by Prakash et al. (2021b) and used a single randomly selected image from each data set to train and test ES3C and GPMM (see Sec. 3.S2.1 for data-related details and Sec. 3.S2.2 for initializations and hyperparameters used). We then compared to BM3D, N2V, DivN and S2S by running the algorithms ourselves on the selected test images (using publicly available source code as for Fig. 3.2, details in Sec. 3.S2.2). Figure 3.4 visualizes the results of this experiment.

On the Convallaria image, N2V achieves the highest PSNR on average, closely followed by ES3C, GPMM and DivN. A larger gap can be observed when comparing to BM3D and S2S which result in comparably clearly lower PSNRs. For both images Mouse nuclei and Mouse actin, the highest average PSNRs are achieved by ES3C, and, in contrast to the Convallaria image, the PSNR gap between ES3C on the one hand, and N2V and DivN on the other is more pronounced. Further experiments that we conducted reveal that PSNRs of DivN can be significantly improved by providing the algorithm with external calibration images for noise model estimation and a complete image sequence for training (Tab. 3.S2). Compared to such ‘non-zero-shot’ variants of DivN, the PSNRs obtained by ES3C for the Mouse nuclei and Mouse actin images nevertheless remain competitive and in individual runs also higher (with ES3C still being applied in a ‘zero-shot’ setting).

3.3 Discussion

Motivated by the global thread posed by the COVID-19 pandemic, our goal here was to ask how the most recent developments of data enhancement algorithms can be leveraged for the specific task of improving the visualization of details in SARS-CoV-2 infection scenes.


A
Convallaria
Mouse nuclei
Mouse actin
Noisy
Target
$\mathrm{BM3D}_{\hat{\sigma}}$
N2V
$\mathrm{DivN}^{1}_{\mathrm{B}}$
S2S
ES3C
GPMM

B

	Convallaria	Mouse nuclei	Mouse actin
Noisy	32.01	29.08	25.10
$\mathrm{BM3D}_{\hat{\sigma}}$	35.88	34.65	30.24
N2V	**38.86 ± 0.14**	36.52 ± 0.15	34.49 ± 0.05
$\mathrm{DivN}^{1}_{\mathrm{B}}$	38.46 ± 0.19	35.96 ± 0.16	35.12 ± 0.14
S2S	35.71 ± 0.01	33.96 ± 0.12	33.30 ± 0.03
ES3C	38.51 ± 0.00	**37.35 ± 0.00**	**35.72 ± 0.00**
GPMM	38.49 ± 0.00	35.61 ± 0.01	31.63 ± 0.00

Figure 3.4: Results for 'zero-shot' denoising of fluorescence microscopy images (benchmark adopted from Prakash et al. (2021b)). We used full-size versions of each test image (in contrast to the methodology applied by Prakash et al., personal communication). A: Qualitative comparison of denoised image parts (image sections and colormap choosen to match Fig. 2 in Prakash et al. (2021b)). B: Denoising performance quantified in terms of PSNR in dB. Reported are averages and standard deviations over three runs of each algorithm (except for deterministic BM3D). Best performances in terms of average PSNR are marked bold. See Sec. 3.2.3 for details.

In this respect, our work follows many other contributions on subtasks of SARS-CoV-2 research which seek to rapidly leverage methods and results from different fields for research on the SARS-CoV-2 virus. Recent enhancement algorithms and a standard baseline were applied to electron microscopy (EM) images that were recorded at the limits of currently possible resolutions. At such limits, EM images are deteriorated by strong noise, however, which can hinder experts to appropriately identify potentially important structures. Modern Machine Learning algorithms can decisively reduce noise and enhance structural details. By doing so, algorithms can decisively facilitate further processing and can help experts in understanding infection processes.

An immediate challenge posed by the images of SARS-CoV-2 infection scenes at high resolution is the availability of exclusively noisy images. As a consequence, current mainstream approaches for denoising such as supervised deep neural networks (DNNs) cannot be used directly because they require large datasets with clean images (e.g., Dong et al., 2019; Tai et al., 2017; Tian et al., 2020; Zhang et al., 2017). For high resolution microscopy images, alternative approaches are consequently required. Recent approaches that are applicable include: the recent DNN-based approaches N2V and S2S, the standard BM3D denoiser, and the generative algorithms DivN, ES3C and GPMM. All these methods can be applied when clean data is absent and even in cases when only a single noisy image is available. The BM3D baseline can by definition be applied to single noisy images without training (given that a-priori noise level information is available). Besides BM3D, a well-known approach is also represented by EPLL (Zoran and Weiss, 2011). However, training using the EPLL approach is usually done based on clean images, which makes the BM3D approach the more suitable baseline for the here considered setting. In contrast to BM3D, N2V and S2S first train DNNs that are then used to map noisy to denoised images. In order to train DNNs when only noisy data is available, N2V and S2S have to significantly amend standard supervised training, however. BM3D and the generative algorithms DivN, ES3C and GPMM (as well as previous generative approaches) are by definition unsupervised, and can as such be applied to noisy data directly. Furthermore, BM3D and the shallow generative models used by ES3C and GPMM typically require much fewer data to learn appropriate representations compared to approaches based on large DNNs. For the applications considered here, noisy data in the form of patches extracted from a single microscopy image proved sufficient. The effectiveness of generative approaches in image restoration tasks has been demonstrated by numerous contributions; in particular, they have been shown to establish state-of-the-art performances in ‘zero-shot’ settings of standard image denoising and inpainting benchmarks (Drefs et al., 2022; Guiraud et al., 2020; Sheikh et al., 2014; Titsias and Lázaro-Gredilla, 2011; Ulyanov et al., 2018; Zhou et al., 2012). The evaluations performed here (Figs. 3.2 and 3.4) highlight that the effectiveness of generative approaches in noise suppression carries over to high-resolution microscopy recordings.

In addition to the estimation of denoised images, we have investigated higher-order reconstruction statistics. Using the variances of pixel estimations based on generative models, images can be computed which highlight details by improving the visualization of edge-like structures (Fig. 3.3). As for the conventionally denoised images, we computed the images displaying higher-order reconstruction statistics based on a single, noisy TEM image of an infection scene (Fig. 3.5). This stands in contrast to the procedure that Nanographics (2021) have recently applied to produce 3D reconstructions of a single SARS-CoV-2 virus. On the one hand, such a tomogram reconstruction of the virus is more detailed and can visualize the spike protein structure and the spike protein distribution on the virus surface very well. On the other hand, the method relies on many TEM images of the same single virus, and many images are more difficult to obtain than single images. Furthermore, whole infection scenes are more challenging to reconstruct than a single virus, and 3D tomogram reconstructions of whole such scenes have (to our knowledge) not been reported so far. Importantly, different approaches for 3D reconstruction come with different artifacts. Tomogram approaches can misestimate surfaces especially for noisy data, while the here shown reconstructions based on generative models can introduce artifacts, e.g., because rare structures are underrepresented. Furthermore, due to being single-image based, actual surfaces are not estimated by the generative approaches investigated here. An advantage is, however, that surface structures do not occlude other structures – the here shown images (Figs. 3.1 and 3.3, and Figs. 3.S2 and 3.S3) have a 'glassy' effect. A disadvantage is that, e.g., internal structures of the virus can get entangled with surface structures. To identify single spikes across the whole surface of the virus, the 3D reconstruction method of Nanographics (2021) is consequently preferable. In general, however, as artifacts are different for the different approaches, the methods can be used to mutually confirm the true nano-scale structures that are visualized. Furthermore, different methods can potentially be combined in hybrid systems. For instance, 'zero-shot' denoising of single images by generative models could be used to generate denoised images for improved 3D surface reconstruction with methods used by Nanographics (2021); or 'zero-shot' denoising could be used in conjunction with other EM imaging methods (Titze and Genoud, 2016).

Finally, different approaches are applicable to different types of available data. Methods for protein reconstruction such as the approaches of Wrapp et al. (2020) or Ke et al. (2020) combine TEM images of many protein molecules of many different viruses, which neither the tomogram method of Nanographics (2021) nor the approach we investigated is capable of. Also, unlike the surface reconstruction by Nanographics, none of the here considered approaches can combine many TEM images of the same virus to generate a 3D tomogram. However, neither the method of Wrapp et al. (2020), Ke et al. (2020) nor of Nanographics (2021) can be used to denoise a single image, while all here investigated approaches can. Images showing virus infections spatially have been made available previously, e.g., by using scanning electron microscopes (NIAID, 2020). Such spatial images are at a lower resolution, however, and do therefore not show the characteristic details of SARS-CoV-2 viruses, for instance. In contrast, we were here interested in methods operating at resolutions that allow for revealing structures as small as SARS-CoV-2 spike proteins. Future work could, for instance, compare different EM recording techniques (transmission and scanning EM) of the same SARS-CoV-2 infection scenes. Such comparisons would help in better understanding the 'glassy' effect we observe, gaining more insight in the details that can be displayed via higher-order reconstruction statistics, and how internal and external structures of the virus get entangled with this approach.

In general and for standard denoising, we observed that essentially all here investigated approaches can decisively enhance important structural details of SARS-CoV-2 infection scenes (Figs. 3.1 to 3.4). Differences between algorithms were often subtle, but we also observed strong dependencies of the algorithms' performance on the specific type of data that was processed. Furthermore, a strong dependence on the type of used evaluation measure was observed, especially for measures that can be used without ground-truth. However, as may be evident by considering the provided examples, the enhanced images with their easy-to-identify details can all serve to distinguish structures at the nanoscale more reliably than the original noisy images. Current research on enhancement algorithms and different ways of visualization can, therefore, help in further understanding the COVID-19 disease. Moreover, vivid and detailed images of SARS-CoV-2 infections may be perceived as more realistic by the general population compared to the more indirect tomogram approach, for instance. Enhanced images may, therefore, generally increase the awareness for the thread posed by COVID-19 now and in the near future.

3.4 Methods

3.4.1 Generative Model Algorithms

One focus of the investigated microscopy image enhancement are two recent generative model algorithms: Evolutionary Spike-and-Slab Sparse Coding (ES3C; Drefs et al., 2022) and Gamma-Poisson Mixture Models (GPMM). The spike-and-slab sparse coding (SSSC) data model has been studied intensively for image processing and other tasks (Goodfellow et al., 2012; Sheikh et al., 2014; Titsias and Lázaro-Gredilla, 2011; Yoshida and West, 2010; Zhou et al., 2012), and the ES3C approach, in particular, has recently been observed to establish state-of-the-art performances on standard image denoising benchmarks in the 'zero-shot' category (Drefs et al., 2022). Here, we, for the first time, apply and benchmark the approach on microscopy image data. The GPMM model, in contrast, represents a novel approach based on recent work by Monk et al. (2018) which we here developed further for the purpose of image enhancement.

ES3C uses a linear sparse coding model with binary-continuous latents $\vec{s} \in \{0,1\}^H$, $\vec{z} \in \mathbb{R}^H$ and continuous, Gaussian-distributed observables $\vec{y} \in \mathbb{R}^D$:

$$\begin{aligned} p\,(\vec{s} \mid \Theta) = \mathcal{B}(\vec{s};\vec{\pi}), \qquad\qquad & p\,(\vec{z} \mid \Theta) = \mathcal{N}(\vec{z};\vec{\mu},\Psi), \\ p\,(\vec{y} \mid \vec{s},\vec{z},\Theta) = \prod_{d=1}^{D} \mathcal{N}\big(y_d; & \sum_{h=1}^{H} W_{dh} s_h z_h, \sigma^2\big), \end{aligned} \tag{3.1}$$

with $\mathcal{B}(\vec{s};\vec{\pi}) = \prod_{h=1}^{H} \pi_h^{s_h}(1-\pi_h)^{1-s_h}$ denoting the Bernoulli prior with $p(s_h=1)=\pi_h$ and $\pi_h \in [0,1]$, $\vec{\mu}$ and Ψ denoting, respectively, the mean and covariance of the H-dimensional multivariate Gaussian prior, $W_{dh} \in \mathbb{R}$ and $W = \{\vec{W}_h\}_{h=1}^{H}$ with $\vec{W}_h = \{W_{dh}\}_{d=1}^{D}$ denoting the generative fields setting the mean of the D-dimensional, isotropic Gaussian over the observables given the latents with variance parameter σ^2. The parameters of the SSSC model (3.1) are $\Theta = \{\vec{\pi},\vec{\mu},\Psi,W,\sigma\}$. GPMMs, on the other hand, are based on Poisson distributed observables $\vec{y} \in \mathbb{N}_0^D$ and assume data points to originate from a single cause, i.e., a particular mixture component. Unlike standard Poisson mixture models, GPMMs use a continuous, Gamma-distributed latent variable $z \in \mathbb{R}^+$ to model mixture components' intensities:

$$p\,(c \mid \Theta) = \pi_c, \qquad p\,(z \mid \Theta) = \mathrm{Gam}(z;\,\alpha,\beta), \qquad p\,(\vec{y} \mid c,z,\Theta) = \prod_{d=1}^{D} \mathrm{Poiss}\left(y_d; zW_{cd}\right), \tag{3.2}$$

with $c = 1,\dots,C$ denoting the index of the mixture components and $\vec{\pi} = \{\pi_c\}_{c=1}^{C}$ with $\pi_c \in [0,1]$ and $\sum_{c=1}^{C} \pi_c = 1$ denoting the corresponding prior activations, $\alpha, \beta, W_{cd} \in \mathbb{R}^+$, and $W = \{\vec{W}_c\}_{c=1}^{C}$ with $\vec{W}_c = \{W_{cd}\}_{d=1}^{D}$ denoting the generative fields of the model. The model parameters of the GPMM are $\Theta = \{\vec{\pi},\alpha,\beta,W\}$.

Given a set of data points $\{\vec{y}^{(n)}\}_{n=1\ldots N}$, we seek parameters $\Theta^* = \mathrm{argmax}_\Theta\, \mathcal{L}(\Theta)$ that optimize the data log-likelihood. A direct optimization of the likelihood is usually challenging, and, instead, efficient algorithms use, for instance, variational Expectation Maximization approaches (Neal and Hinton, 1998; Saul and Jordan, 1995) and optimize a variational lower bound

$$\mathcal{F}(q,\Theta) \leq \mathcal{L}(\Theta) = \sum_{n=1}^{N} \log\big(p(\vec{y}^{\,(n)} \,|\, \Theta)\big) \tag{3.3}$$

of the log-likelihood (the variational lower bound is also known as free energy or evidence lower bound – ELBO). For the SSSC generative model (Eq. (3.1)), $p(\vec{y}\,|\Theta) = \sum_{\{\vec{s}\}} \int_{\vec{z}} p(\vec{y} \mid \vec{s}, \vec{z}, \Theta) p(\vec{s} \mid \Theta) p(\vec{z} \mid \Theta)\, d\vec{z}$ with $\sum_{\{\vec{s}\}}$ running over all possible configurations of the binary latent vector $\vec{s}$, and for GPMMs (Eq. (3.2)), $p(\vec{y}\,|\Theta) = \sum_{c=1}^{C} \int_z p(\vec{y} \mid c, z) p(c \mid \Theta) p(z \mid \Theta) dz$. The ES3C approach exploits that (i) parameter update equations (M-steps) take analytically tractable forms (Sheikh et al., 2014) and that (ii) efficient variational E-steps can be realized by applying evolutionary algorithms to optimize variational parameters (see Sec. 3.S1 and Drefs et al. (2022) for details). For GPMMs, the Gamma distributed latents and Poisson observables may suggest analytical intractabilities at first sight, yet, closed-form expressions for the posterior can be derived. For the derivation, we make use of theoretical results by Monk et al. (2018), who studied a more general class of such models for intensity-sensitive classification. For the GPMM model (Eq. (3.2)), the following M-step equations can be derived (a detailed derivation is provided in Sec. 3.S1):

$$\begin{aligned}
W_{cd} &= \frac{(\beta + \sum_{d' \neq d} W_{cd'}) \sum_{n=1}^{N} p(c|\vec{y}^{\,(n)}, \Theta) y_d^{(n)}}{\sum_{n=1}^{N} p(c|\vec{y}^{\,(n)}, \Theta)(\alpha + \hat{y}^{(n)} - y_d^{(n)})}, \\
\pi_c &= \frac{1}{N} \sum_{n=1}^{N} p(c \,|\, \vec{y}^{\,(n)}, \Theta), \\
\alpha &= \exp\Big(\frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{p(c|\vec{y}^{\,(n)},\Theta)} \big[\log\beta - g(\alpha) + \psi(\hat{y}^{(n)} + \alpha) + \log(\beta + \hat{W}_c)\big]\Big), \\
\beta &= \frac{N \cdot \alpha}{\sum_{n=1}^{N} \mathbb{E}_{p(c|\vec{y}^{\,(n)},\Theta)} \big[(\hat{y}^{(n)} + \alpha)(\beta + \hat{W}_c)^{-1}\big]}, \\
\text{with} \quad g(\alpha) &= -\frac{1}{2\alpha} - \frac{1}{12\alpha^2} + \frac{1}{120\alpha^4} - \frac{1}{256\alpha^6} + \frac{1}{240\alpha^8},
\end{aligned} \tag{3.4}$$

and where ψ denotes the digamma function. While there are closed-form solutions for the parameters W_{cd} and π_c, the parameters α and β are updated via fixed-point iteration of the form $x = f(x)$. The M-steps (Eq. (3.4)) depend on the posterior $p(c \,|\, \vec{y}, \Theta)$, which can be derived by marginalizing the joint posterior $p(c, z \,|\, \vec{y}, \Theta)$ over the continuous latent z (details in Sec. 3.S1). The posterior $p(c \,|\, \vec{y}, \Theta)$ computed in the E-step is given by:

$$\begin{aligned}
p(c \,|\, \vec{y}, \Theta) &= \frac{p(\vec{y}\,|c, \Theta) p(c \,|\, \Theta)}{\sum_{c'=1}^{C} p(\vec{y}\,|\, c', \Theta) p(c' \,|\, \Theta)}, \\
\text{with} \quad p(\vec{y} \,|\, c, \Theta) &= \left(\prod_{d=1}^{D} \frac{W_{cd}^{y_d}}{y_d!}\right) \frac{\hat{y}!}{\hat{W}_c^{\hat{y}}} \mathrm{NB}\left(\hat{y}; \alpha, \frac{\hat{W}_c}{\beta + \hat{W}_c}\right),
\end{aligned} \tag{3.5}$$

with NB denoting the negative binomial distribution, $\hat{W}_c = \sum_{d=1}^{D} W_{cd}$, and $\hat{y} = \sum_{d=1}^{D} y_d$. As the continuous latents can be marginalized, the resulting EM algorithm maintains the computational complexity of a mixture model. We can consequently efficiently train the GPMM model (Eq. (3.2)) without applying variational approximations.

3.4.2 Probabilistic Data Estimation

Given a set of optimized parameters Θ, we can use the data representations learned with ES3C and GPMM for probabilistic data estimation: for instance, we can estimate the most likely, non-noisy version $\vec{y}^{\,\mathrm{est}}$ of a given noisy data point $\vec{y}$ by estimating the first moment of the modeled pixel distribution. The data estimator we apply here is derived based on the posterior predictive distribution $p(\vec{y}^{\,\mathrm{est}}|\vec{y},\Theta)$. The concrete expression we use here for ES3C is given by (Drefs et al., 2022):

$$y_d^{\mathrm{est}} \leftarrow \mathbb{E}_{p(\vec{y}^{\,\mathrm{est}}|\vec{y},\Theta)}\big[y_d^{\mathrm{est}}\big] = \mathbb{E}_{p(\vec{s},\vec{z}\,|\vec{y},\Theta)}\Big[\mathbb{E}_{p(y_d^{\mathrm{est}}|\vec{s},\vec{z},\Theta)}\big[y_d^{\mathrm{est}}\big]\Big] = (\vec{W}_d)^{\mathrm{T}}\mathbb{E}_{p(\vec{s},\vec{z}\,|\vec{y},\Theta)}\big[\vec{s}\odot\vec{z}\big], \quad (3.6)$$

with $\vec{W}_d = \{W_{dh}\}_{h=1}^{H}$, $\odot$ denoting point-wise multiplication, and $\mathbb{E}[\cdot]$ denoting expectation. Efficient computation of Eq. (3.6) exploits that the expectation w.r.t. the binary-continuous posterior $p(\vec{s},\vec{z}\mid\vec{y},\Theta)$ can be reformulated as an expectation taken only w.r.t. the binary latent space (Sheikh et al., 2014), which, in turn, can be approximated efficiently using truncated posteriors (see Sec. 3.S1 and Drefs et al. (2022) for details). We follow the same line of reasoning in order to derive a data estimator for GPMM; the final expression takes the following form (details in Sec. 3.S1):

$$y_d^{\mathrm{est}} \leftarrow \mathbb{E}_{p(\vec{y}^{\,\mathrm{est}}|\vec{y},\Theta)}\big[y_d^{\mathrm{est}}\big] = (\alpha+\hat{y})\,\mathbb{E}_{p(c\,|\,\vec{y},\Theta)}\left[\frac{W_{cd}}{\beta+\hat{W}_c}\right]. \quad (3.7)$$

3.4.3 Estimation of Non-Noisy Image Pixels

To enhance a given microscopy image using ES3C and GPMM, we extract overlapping patches (Burger et al., 2012; Elad and Aharon, 2006; Mairal et al., 2008; Zhou et al., 2012) from the image and treat the obtained set of image patches as our training dataset $\mathcal{Y} = \{\vec{y}^{\,(n)}\}_{n=1}^{N}$ (patch sizes used are listed in Sec. 3.S2.2). After training, we apply Eqs. (3.6) and (3.7) to each data point in $\mathcal{Y}$ (i.e., to each image patch). Due to mutually overlapping patches, this results in multiple denoised estimates for a given image pixel, and, here, we consider both means and variances of pixel estimations for image reconstruction (see Fig. 3.5 for an illustration).

3.4.4 Evaluation Metrics

A variety of objective measures for quantifying image enhancement performance exists, with the requirement for reference information being one distinguishing criterion. In benchmarks with artificially degraded images (e.g. Gaussian denoising), processed images can be compared to the available respective ‘clean’ image. Image restoration performance can then be quantified based on measures such as peak-signal-to-noise ratio (PSNR) or structural similarity (Wang et al., 2004) which both rely on the ground-truth image as input. For the SARS-CoV-2 TEM microscopy data considered in this study, clean reference images are, however, not available, and we are consequently restricted to evaluation metrics that can be calculated based on a single image (also referred to as no-reference metrics). For our purposes,

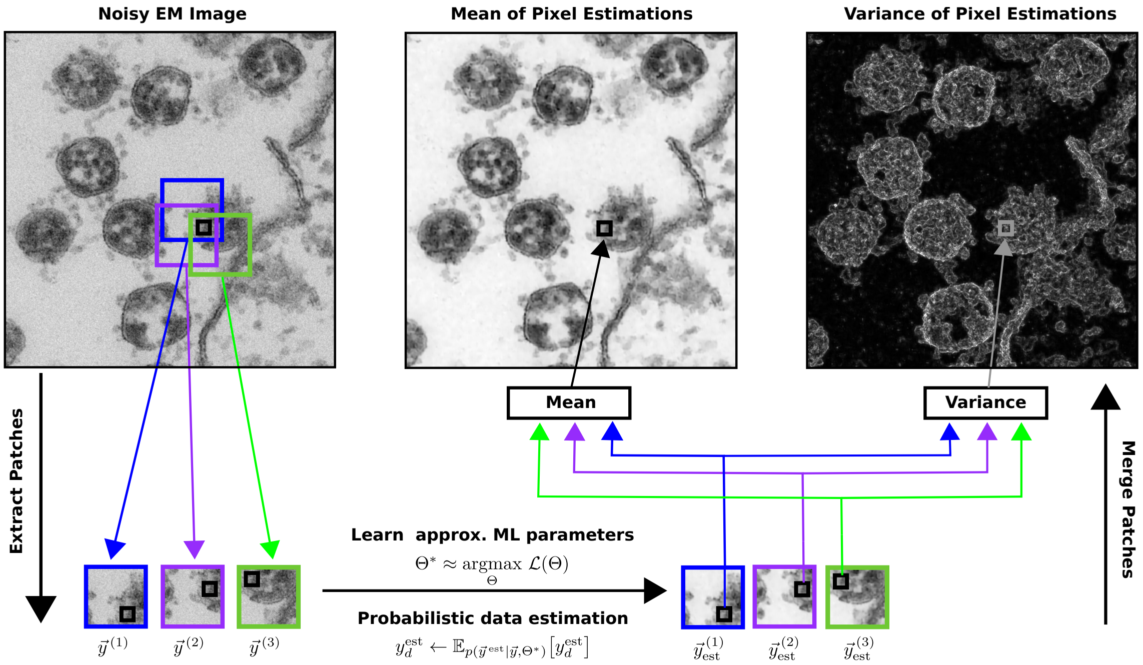

Noisy EM Image
Mean of Pixel Estimations
Variance of Pixel Estimations
Extract Patches
Mean
Variance
Merge Patches
Learn approx. ML parameters
$\Theta^* \approx \underset{\Theta}{\mathrm{argmax}}\, \mathcal{L}(\Theta)$
Probabilistic data estimation
$y_d^{\mathrm{est}} \leftarrow \mathbb{E}_{p(\vec{y}^{\,\mathrm{est}}|\vec{y},\Theta^*)}[y_d^{\mathrm{est}}]$
$\vec{y}^{(1)}$
$\vec{y}^{(2)}$
$\vec{y}^{(3)}$
$\vec{y}^{(1)}_{\mathrm{est}}$
$\vec{y}^{(2)}_{\mathrm{est}}$
$\vec{y}^{(3)}_{\mathrm{est}}$

Figure 3.5: ‘Zero-shot’ image enhancement method using probabilistic generative model algorithms for denoising (see Sec. 3.4 for details): As input (top left), here we use pre-processed versions of publicly available transmission electron microscopy (TEM) images of SARS-CoV-2 infected cell cultures (Laue et al., 2021) (see Sec. 3.S2.1 for data-related details). Using patches extracted from the noisy TEM image (bottom left), we first learn a probabilistic representation of the image using an appropriately chosen data model (here we apply linear spike-and-slab sparse coding models and Gamma-Poisson mixtures). We can then apply the learned representation to probabilistically reconstruct each image patch (bottom right). Finally, we generate reconstructed TEM images by computing means and variances of pixel estimates obtained from mutually overlapping patches (top center and right). The TEM image enhancement approach does not require (clean) training data and can directly be applied to a single noisy image.

we consider blind noise level estimation (Chen et al., 2015a), signal-to-noise quantification based on paired hand-labeled signal and background regions (Bepler et al., 2020), and image quality ranking (Koho et al., 2016a). For the signal-to-noise quantification, we follow Bepler et al. (2020) and label M pairs of signal and background regions $\vec{x}_s^{(m)}$ and $\vec{x}_b^{(m)}$ for each test image, respectively, to calculate SNR $= \frac{10}{M}\sum_{m=1}^{M}\log_{10}(s^{(m)}) - \log_{10}(\sigma_b^{(m)})$, with $s^{(m)} = (\mu_s^{(m)} - \mu_b^{(m)})^2$, $\mu_s^{(m)}$ and $\mu_b^{(m)}$ denoting the mean of $\vec{x}_s^{(m)}$ and $\vec{x}_b^{(m)}$, respectively, and $\sigma_b^{(m)}$ denoting the variance of $\vec{x}_b^{(m)}$. For the quality ranking, we first evaluate the denoised images obtained with the investigated algorithms for each test image individually based on the normalized quality scores computed using publicly available source code (Koho et al., 2016b). In total, six algorithms are considered such that the rankings correspond to integer numbers between one and six. Finally, we average ranks across test images and translate the resulting numbers into integers again. We perform separate rankings based on the measures CV, invSTD, Entropy and MeanBin.

Supplementary Material

3.S1 Derivations

3.S1.1 EM Update Rules Spike-and-Slab Sparse Coding

To efficiently optimize the parameters of the SSSC model (Eq. (3.1)), the here applied ES3C approach exploits Evolutionary Variational Optimization (EVO; Drefs et al., 2022), a fully variational Expectation Maximization approach which uses truncated posteriors (Hirschberger et al., 2022; Lücke and Eggert, 2010; Lücke and Forster, 2019; Sheikh et al., 2014; Shelton et al., 2017) of the form

$$\begin{aligned} q^{(n)}(\vec{s} \mid \mathcal{K}^{(n)}, \Theta) &:= \frac{p\big(\vec{s} \mid \vec{y}^{\,(n)}, \Theta\big)}{\sum\limits_{\vec{s}\,' \in \mathcal{K}^{(n)}} p\big(\vec{s}\,' \mid \vec{y}^{\,(n)}, \Theta\big)} \delta\big(\vec{s} \in \mathcal{K}^{(n)}\big), \\ \text{with} \quad \delta\big(\vec{s} \in \mathcal{K}^{(n)}\big) &= \begin{cases} 1 \text{ if } \vec{s} \in \mathcal{K}^{(n)}, \\ 0 \text{ otherwise,} \end{cases} \end{aligned} \tag{3.S1}$$

as variational distributions. Truncated posteriors as defined by Eq. (3.S1) are proportional to the exact posterior $p\big(\vec{s} \mid \vec{y}^{\,(n)}, \Theta\big)$ on a subset of the latent space defined by a set of latent states, denoted $\mathcal{K}^{(n)}$, (i) that are supposed to accumulate most posterior mass, and (ii) whose size $S = |\mathcal{K}^{(n)}|$ is chosen much smaller than the full latent space of size 2^H. Crucial for the variational E-step is that the free energy objective can be expressed as

$$\mathcal{F}(\mathcal{K}^{(1\ldots N)}, \Theta) = \sum_{n=1}^{N} \log\Big(\sum_{\vec{s} \in \mathcal{K}^{(n)}} p\big(\vec{y}^{\,(n)}, \vec{s} \mid \Theta\big)\Big). \tag{3.S2}$$

The states in the sets $\mathcal{K}^{(n)}$ are variational parameters of the EVO approach, and they are optimized using evolutionary computation: To increase Eq. (3.S2) in the variational E-step, the sets $\mathcal{K}^{(n)}$ are varied by replacing states $\vec{s} \in \mathcal{K}^{(n)}$ with new states $\vec{s}^{\,\mathrm{new}} \notin \mathcal{K}^{(n)}$ that fulfill the criterion

$$p(\vec{s}^{\,\mathrm{new}}, \vec{y}^{\,(n)} \mid \Theta) \;>\; p(\vec{s}, \vec{y}^{\,(n)} \mid \Theta)\,, \tag{3.S3}$$

and new states are evolved by elementary genetic operations, namely selection, mutation and crossover (see Drefs et al. (2022) for details). Equations (3.S2) and (3.S3) can efficiently be evaluated given that the joint $p\big(\vec{y}^{\,(n)}, \vec{s} \mid \Theta\big)$ of the SSSC model (Eq. (3.1)) has a computationally tractable form:

$$p\big(\vec{y}^{\,(n)}, \vec{s} \mid \Theta\big) = \mathcal{B}(\vec{s}; \vec{\pi})\; \mathcal{N}(\vec{y}^{\,(n)};\, \tilde{W}_{\vec{s}}\, \vec{\mu}, C_{\vec{s}}), \tag{3.S4}$$

with $C_{\vec{s}} = \sigma^2 \mathbb{1} + \tilde{W}_{\vec{s}} \Psi \tilde{W}_{\vec{s}}^{\mathrm{T}}$ and $(\tilde{W}_{\vec{s}})_{dh} = W_{dh} s_h$ s.t. $W(\vec{s} \odot \vec{z}) = \tilde{W}_{\vec{s}} \vec{z}$. The M-step update rules for the SSSC model have been derived before (Sheikh et al., 2014), and we report the final expressions here for completeness:

$$
\begin{aligned}
W &= \frac{\sum_{n=1}^{N} \vec{y}^{(n)} \mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s} \odot \vec{z}\big]^{\mathrm{T}}}{\sum_{n=1}^{N} \mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\,(\vec{s} \odot \vec{z}\,)\,(\vec{s} \odot \vec{z}\,)^{\mathrm{T}}\,\big]}, \\
\vec{\pi} &= \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s}\big], \\
\vec{\mu} &= \frac{\sum_{n=1}^{N} \mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s} \odot \vec{z}\big]}{\sum_{n=1}^{N} \mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s}\big]}, \\
\Psi &= \sum_{n=1}^{N} \Big[\mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\,(\vec{s} \odot \vec{z}\,)\,(\vec{s} \odot \vec{z}\,)^{\mathrm{T}}\,\big] - \mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s}\vec{s}^{\mathrm{T}}\big] \odot \vec{\mu}\vec{\mu}^{\mathrm{T}}\Big] \odot \left(\sum_{n=1}^{N} \mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s}\vec{s}^{\mathrm{T}}\big]\right)^{-1}, \\
\sigma^2 &= \frac{1}{ND} \operatorname{Tr}\left(\sum_{n=1}^{N} \Big[\vec{y}^{(n)} (\vec{y}^{(n)})^{\mathrm{T}} - W\big[\mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s} \odot \vec{z}\big] \cdot \mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s} \odot \vec{z}\big]^{\mathrm{T}}\big] W^{\mathrm{T}}\Big]\right).
\end{aligned}
\tag{3.S5}
$$

The expressions of the form $\mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}[\cdot]$ in Eq. (3.S5) denote expectation values w.r.t. the full binary-continuous posterior of the SSSC model. These expectations can be reformulated as functions of expectation values $\mathbb{E}_{q^{(n)}(\vec{s})}[\cdot]$ that are taken w.r.t. the posterior over the binary (rather than the full binary-continuous) latent space (Sheikh et al., 2014):

$$
\begin{aligned}
\mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s}\big] &= \mathbb{E}_{q^{(n)}(\vec{s})}\big[\vec{s}\big], \\
\mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s}\vec{s}^{\mathrm{T}}\big] &= \mathbb{E}_{q^{(n)}(\vec{s})}\big[\vec{s}\vec{s}^{\mathrm{T}}\big], \\
\mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\vec{s} \odot \vec{z}\big] &= \mathbb{E}_{q^{(n)}(\vec{s})}\big[\vec{\kappa}_{\vec{s}}^{(n)}\big], \\
\mathbb{E}_{q^{(n)}(\vec{s},\vec{z})}\big[\,(\vec{s} \odot \vec{z}\,)\,(\vec{s} \odot \vec{z}\,)^{\mathrm{T}}\,\big] &= \mathbb{E}_{q^{(n)}(\vec{s})}\big[\Lambda_{\vec{s}} + \vec{\kappa}_{\vec{s}}^{(n)} (\vec{\kappa}_{\vec{s}}^{(n)})^{\mathrm{T}}\big],
\end{aligned}
\tag{3.S6}
$$

where $\Lambda_{\vec{s}} = (\sigma^{-2} \tilde{W}_{\vec{s}}^{\mathrm{T}} \tilde{W}_{\vec{s}} + \Psi_{\vec{s}}^{-1})^{-1}$, and $\vec{\kappa}_{\vec{s}}^{(n)} = (\vec{s} \odot \vec{\mu}) + \sigma^{-2} \Lambda_{\vec{s}} \tilde{W}_{\vec{s}}^{\mathrm{T}} (\vec{y}^{(n)} - \tilde{W}_{\vec{s}} \vec{\mu})$. Eq. (3.S6) can efficiently be evaluated by approximating the expectations w.r.t. the exact binary posterior $q^{(n)}(\vec{s}) = p\big(\vec{s} \mid \vec{y}^{(n)}, \Theta\big)$ by expectations taken w.r.t. a truncated posterior distribution as in Eq. (3.S1). The expectation values that are ultimately computed in the M-step take the following form:

$$
\mathbb{E}_{q^{(n)}(\vec{s} \mid \mathcal{K}^{(n)}, \Theta)}\big[g(\vec{s})\big] = \frac{\sum\limits_{\vec{s} \in \mathcal{K}^{(n)}} g(\vec{s})\, p\big(\vec{y}^{(n)}, \vec{s} \mid \Theta\big)}{\sum\limits_{\vec{s}' \in \mathcal{K}^{(n)}} p\big(\vec{y}^{(n)}, \vec{s}' \mid \Theta\big)},
\tag{3.S7}
$$

with $p\big(\vec{y}^{(n)}, \vec{s} \mid \Theta\big)$ given by Eq. (3.S4) (Drefs et al., 2022; Sheikh et al., 2014).

3.S1.2 EM Update Rules Gamma-Poisson Mixtures

To efficiently train the GPMM (Eq. (3.2)), we apply Expectation Maximization. The here considered GPMM has been defined specifically for the purposes of this paper. Consequently, all the following derivations are specific to this work. The derivations make use of earlier theoretical results by Monk et al. (2018) about a similar model developed for biologically plausible classification with class-specific intensities. Before deriving E- and M-step update rules, we reformulate the product of the conditional probability $p(\vec{y}\,|z,c,\Theta)$ and the prior probability $p(z|\Theta)$ in the following way:

$$p(\vec{y}\,|z,c,\Theta)p(z|\Theta) = \left(\prod_d \frac{W_{cd}^{y_d}}{y_d!}\right) z^{\hat{y}+\alpha-1}\frac{\beta^\alpha}{\Gamma(\alpha)}\exp\left(-z(\beta+\hat{W}_c)\right) \tag{3.S8}$$

Introducing the factors $\Gamma\left(\hat{y}+\alpha\right)^{-1}$ and $\left(\beta+\hat{W}_c\right)^{\hat{y}+\alpha}$, we can recognize the underlined term as a Gamma distribution:

$$\begin{aligned}p(\vec{y}\,|z,c,\Theta)p(z|\Theta) &= \left(\prod_d \frac{W_{cd}^{y_d}}{y_d!}\right)\frac{\Gamma\left(\hat{y}+\alpha\right)}{\Gamma(\alpha)}\frac{\beta^\alpha}{\left(\beta+\hat{W}_c\right)^{\hat{y}+\alpha}} \\ &\quad \underbrace{\cdot\, z^{\hat{y}+\alpha-1}\exp\left(-z(\beta+\hat{W}_c)\right)\frac{\left(\beta+\hat{W}_c\right)^{\hat{y}+\alpha}}{\Gamma\left(\hat{y}+\alpha\right)}}_{=\mathrm{Gam}(z;\alpha+\hat{y},\beta+\hat{W}_c)}\end{aligned} \tag{3.S9}$$

$$= \left(\prod_d W_{cd}^{y_d}\right)\frac{\Gamma\left(\hat{y}+\alpha\right)}{\Gamma(\alpha)}\frac{\beta^\alpha}{\left(\beta+\hat{W}_c\right)^{\hat{y}+\alpha}}\cdot \mathrm{Gam}(z;\alpha+\hat{y},\beta+\hat{W}_c) \tag{3.S10}$$

Multiplying with $\hat{y}!\hat{y}!^{-1}$ and $\hat{W}_c^{\hat{y}}\hat{W}_c^{-\hat{y}}$, we can recognize a negative binomial distribution, which yields to our final result:

$$p(\vec{y}\,|z,c,\Theta)p(z|\Theta) = \left(\prod_d \frac{W_{cd}^{y_d}}{y_d!}\right)\frac{\hat{y}!}{\hat{W}_c^{\hat{y}}}\mathrm{NB}\left(\hat{y};\alpha,\frac{\hat{W}_c}{\beta+\hat{W}_c}\right)\cdot \mathrm{Gam}(z;\alpha+\hat{y},\beta+\hat{W}_c) \tag{3.S11}$$

$$= p(\vec{y}\,|c,\Theta)\mathrm{Gam}(z;\alpha+\hat{y},\beta+\hat{W}_c) \tag{3.S12}$$

Given a set of datapoints $\mathcal{Y} = \left\{\vec{y}^{\,(n)}\right\}_{n=1,\dots,N}$, we seek parameters Θ^* that optimize the data likelihood which for the GPMM (Eq. (3.2)) is given by:

$$\mathcal{L}(\Theta) = \sum_{n=1}^{N}\log\bigl(p(\vec{y}^{\,(n)}|\Theta)\bigr) = \sum_{n=1}^{N}\log\left(\sum_{c=1}^{C}\int p(\vec{y}^{\,(n)}|z,c,\Theta)p(z|\Theta)p(c|\Theta)dz\right) \tag{3.S13}$$

$$\overset{(3.S12)}{=} \sum_{n=1}^{N}\log\left(\sum_{c=1}^{C}p(\vec{y}^{\,(n)}|c,\Theta)p(c|\Theta)\cdot\int \mathrm{Gam}(z;\alpha+\hat{y}^{(n)},\beta+\hat{W}_c)dz\right) \tag{3.S14}$$

$$= \sum_{n=1}^{N}\log\left(\sum_{c=1}^{C}p(\vec{y}^{\,(n)}|c,\Theta)p(c|\Theta)\right) \tag{3.S15}$$

$$\geq \sum_{n=1}^{N}\sum_{c=1}^{C}q^{(n)}\left(c,\Theta\right)\Big[\log p(\vec{y}^{\,(n)}|c,\Theta)+\log p(c|\Theta)\Big] + \mathcal{H}(\Theta) \tag{3.S16}$$

$$= \mathcal{F}(q,\Theta) \tag{3.S17}$$

A direct optimization of $\mathcal{L}(\Theta)$ in the form of Eq. (3.S15) is intractable. Following Saul and Jordan (1995), Neal and Hinton (1998), we introduce variational distributions $q^{(n)}(c,\Theta)$ (in Eq. (3.S16)) and apply Jensen's inequality (from Eqs. (3.S15) to (3.S16)) in order to derive the free energy $\mathcal{F}(q,\Theta)$; the free energy is a lower bound of the likelihood and it allows for efficient optimization. $\mathcal{H}\left(q^{(n)},\Theta\right)$ in Eq. (3.S16) denotes the Shannon entropy of the distributions $q^{(n)}(c,\Theta)$. For our purposes, we use exact posteriors as variational distributions, i.e. we set $q^{(n)}(c,\Theta) = p(c|\vec{y}^{\,(n)},\Theta)$. With this choice, the inequality in Eq. (3.S16) becomes an equality. The free energy than takes the following form:

$$\mathcal{F}(q,\Theta_t,\Theta_{t-1}) = \sum_{n=1}^{N}\sum_{c'=1}^{C} p(c|\vec{y}^{\,(n)},\Theta_{t-1})\left[\log p(\vec{y}^{\,(n)}|c',\Theta_t) + \log p(c'|\Theta_t)\right] + \mathcal{H}(\Theta_{t-1}) \tag{3.S18}$$

In the E-step, the posterior probability $p(c|\vec{y}^{\,(n)},\Theta)$ is computed which can be derived by applying Bayes' rule:

$$p(c|\vec{y}^{\,(n)},\Theta) = \frac{p(\vec{y}\,|c,\Theta)p(c|\Theta)}{\sum_{c'}\int p(\vec{y}\,|c',\Theta)p(c'|\Theta)dz} \tag{3.S19}$$

In the M-step new parameters are computed, which maximize the free energy using the current estimate of the posterior probability distribution. Each new parameter, including W, π_c, α and β, will be computed using update rules that are derived via $\frac{\partial}{\partial\Theta}\mathcal{F}(q,\Theta)$. For the update rule of the parameter W, a closed-form solution can be obtained:

$$\begin{aligned}
0 &\overset{!}{=} \frac{\partial}{\partial W_{cd}}\mathcal{F}(q,\Theta)\\
&= \sum_{n=1}^{N}\sum_{c'=1}^{C} p(c'|\vec{y}^{\,(n)},\Theta)\left(\sum_{d'=1}^{D} y_{d'}^{(n)}\frac{\partial}{\partial W_{cd}}\log W_{c'd'} - (\hat{y}^{(n)}+\alpha)\frac{\partial}{\partial W_{cd}}\log(\beta+\hat{W}_{c'})\right)\\
&= \sum_{n=1}^{N}\sum_{c'=1}^{C} p(c'|\vec{y}^{\,(n)},\Theta)\left(\sum_{d'=1}^{D} y_{d'}^{(n)}\frac{1}{W_{c'd'}}\delta_{dd'}\delta_{cc'} - \frac{\hat{y}^{(n)}+\alpha}{\beta+\hat{W}_{c'}}\delta_{cc'}\right)\\
&= \frac{1}{W_{cd}}\sum_{n=1}^{N} p(c|\vec{y}^{\,(n)},\Theta)y_d^{(n)} - \frac{1}{\beta+\hat{W}_c}\sum_{n=1}^{N} p(c|\vec{y}^{\,(n)},\Theta)(\hat{y}^{(n)}+\alpha)
\end{aligned} \tag{3.S20}$$

$$\begin{aligned}
\Leftrightarrow \sum_{n=1}^{N} p(c|\vec{y}^{\,(n)},\Theta)(\hat{y}^{(n)}+\alpha) &= \frac{\beta+\sum_{d'}W_{cd'}}{W_{cd}}\sum_{n=1}^{N} p(c|\vec{y}^{\,(n)},\Theta)y_d^{(n)}\\
&= \left(\frac{\beta+\sum_{d'\neq d}W_{cd'}}{W_{cd}}+1\right)\sum_{n=1}^{N} p(c|\vec{y}^{\,(n)},\Theta)y_d^{(n)}
\end{aligned} \tag{3.S21}$$

$$\Leftrightarrow \sum_{n=1}^{N} p(c|\vec{y}^{\,(n)},\Theta)(\hat{y}^{(n)}+\alpha-y_d^{(n)}) = \frac{\beta+\sum_{d'\neq d}W_{cd'}}{W_{cd}}\sum_{n=1}^{N} p(c|\vec{y}^{\,(n)},\Theta)y_d^{(n)} \tag{3.S22}$$

$$\Leftrightarrow W_{cd} = \frac{(\beta+\sum_{d'\neq d}W_{cd'})\sum_{n=1}^{N} p(c|\vec{y}^{\,(n)},\Theta)y_d^{(n)}}{\sum_{n=1}^{N} p(c|\vec{y}^{\,(n)},\Theta)(\alpha+\hat{y}^{(n)}-y_d^{(n)})} \tag{3.S23}$$

The corresponding update rule for the parameter π_c is obtained in a similar manner, with the additional constraint that $\sum_c \pi_c = 1$, enforced using a Lagrange multiplier λ:

$$0 \overset{!}{=} \frac{\partial}{\partial \pi_c} \mathcal{F}(q,\Theta) = \sum_{n=1}^{N}\sum_{c'=1}^{C} p(c|\vec{y}^{(n)},\Theta) \frac{\partial}{\partial \pi_c} \log \pi_{c'} + \lambda \left(\sum_{c'=1}^{C} \frac{\partial}{\partial \pi_c} \pi_{c'} - 1 \right) \tag{3.S24}$$

$$= \sum_{n=1}^{N}\sum_{c'=1}^{C} p(c|\vec{y}^{(n)},\Theta) \frac{1}{\pi_{c'}} \delta_{cc'} + \lambda \sum_{c'=1}^{C} \delta_{cc'} = \sum_{n=1}^{N} p(c|\vec{y}^{(n)},\Theta) \frac{1}{\pi_c} + \lambda \tag{3.S25}$$

$$\Rightarrow \lambda \pi_c = - \sum_{n=1}^{N} p(c|\vec{y}^{(n)},\Theta) \tag{3.S26}$$

By summing Eq. (3.S26) over c, we obtain $\lambda = -N$. This yields to the final solution of the update rule of π_c:

$$-N\pi_c = - \sum_{n=1}^{N} p(c|\vec{y}^{(n)},\Theta) \tag{3.S27}$$

$$\pi_c = \frac{1}{N} \sum_{n=1}^{N} p(c|\vec{y}^{(n)},\Theta) \tag{3.S28}$$

The update rules of the parameters α and β can not be derived in closed-form. Therefore, we apply fixed-point iterations of the form $x = f(x)$ in order to update these parameters. The update rule for α looks as follows:

$$0 \overset{!}{=} \frac{\partial}{\partial \alpha} \mathcal{F}(q,\Theta) \tag{3.S29}$$

$$\begin{aligned} = \sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\vec{y}^{(n)},\Theta) \Bigg(& \frac{\partial}{\partial \alpha} \alpha \log \beta - \frac{\partial}{\partial \alpha} \log \Gamma(\alpha) \\ & + \frac{\partial}{\partial \alpha} \log \Gamma(\hat{y}^{(n)} + \alpha) - \frac{\partial}{\partial \alpha} \alpha \log(\beta + \hat{W}_c) \Bigg) \end{aligned} \tag{3.S30}$$

$$= \sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\vec{y}^{(n)},\Theta) \left(\log \beta - \psi(\alpha) + \psi(\hat{y}^{(n)} + \alpha) - \log(\beta + \hat{W}_c) \right) \tag{3.S31}$$

where ψ denotes the digamma function. To solve this equation, we aproximate $\psi(\alpha)$ with:

$$\psi(\alpha) \approx \log(\alpha) \underbrace{- \frac{1}{2\alpha} - \frac{1}{12\alpha^2} + \frac{1}{120\alpha^4} - \frac{1}{256\alpha^6} + \frac{1}{240\alpha^8}}_{g(\alpha)} = \log(\alpha) + g(\alpha) \tag{3.S32}$$

By inserting this aproximation into Eq. (3.S31), the update rule of α yields to:

$$\Rightarrow \log(\alpha) = \frac{1}{N} \sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\vec{y}^{(n)},\Theta) \left(\log \beta - g(\alpha) + \psi(\hat{y}^{(n)} + \alpha) - \log(\beta + \hat{W}_c) \right) \tag{3.S33}$$

$$\Rightarrow \alpha = \exp \left(\frac{1}{N} \sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\vec{y}^{(n)},\Theta) \left(\log \beta - g(\alpha) + \psi(\hat{y}^{(n)} + \alpha) - \log(\beta + \hat{W}_c) \right) \right) \tag{3.S34}$$

The corresponding update of β is obtained in a similar manner:

$$0 \overset{!}{=} \frac{\partial}{\partial \beta}\mathcal{F}(q,\Theta) = \sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\vec{y}^{(n)},\Theta)\left(\alpha\frac{\partial}{\partial\beta}\log\beta - (\hat{y}^{(n)}+\alpha)\frac{\partial}{\partial\beta}\log\left(\beta+\hat{W}_c\right)\right) \tag{3.S35}$$

$$= N\frac{\alpha}{\beta} - \sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\vec{y}^{(n)},\Theta)(\hat{y}^{(n)}+\alpha)(\beta+\hat{W}_c)^{-1} \tag{3.S36}$$

$$\Rightarrow \beta = \frac{N\cdot\alpha}{\sum_{n=1}^{N}\sum_{c=1}^{C} p(c|\vec{y}^{(n)},\Theta)(\hat{y}^{(n)}+\alpha)(\beta+\hat{W}_c)^{-1}} \tag{3.S37}$$

3.S1.3 Data Estimator Gamma-Poisson Mixtures

As described in Sec. 3.4, we apply a probabilistic data estimator based on the posterior predictive distribution to estimate the most likely, non-noisy version $\vec{y}^{\mathrm{est}}$ of a given noisy data point $\vec{y}$. This data estimator is model-specific, and, for the SSSC model, a concrete expression has been derived by Drefs et al. (2022). However, for the GPMM, such concrete expression still needs to be derived. For the GPMM, the posterior predictive distribution can be written as follows:

$$p(\vec{y}^{\mathrm{est}}|\vec{y}\,) = \sum_{c}\int p(\vec{y}^{\mathrm{est}}|c,z,\Theta)p(c,z|\vec{y},\Theta) \tag{3.S38}$$

The noisy pixel values are then replaced by the expectation values of this posterior predictive distribution:

$$y_d^{\mathrm{est}} \leftarrow \mathbb{E}_{p(\vec{y}^{\mathrm{est}}|\vec{y},\Theta)}\left[y_d^{\mathrm{est}}\right] = \mathbb{E}_{p(c,z|\vec{y},\Theta)}\left[\mathbb{E}_{p(\vec{y}^{\mathrm{est}}|c,z,\Theta)}\left[y_d^{\mathrm{est}}\right]\right] \tag{3.S39}$$

The inner expectation is the first moment of the noise model (Eq. (3.2), right). This expectation is given by $\mathbb{E}_{p(\vec{y}^{\mathrm{est}}|c,z,\Theta)}\left[y_d^{\mathrm{est}}\right] = zW_{cd}$. Furthermore, the posterior distribution $p(c,z|\vec{y},\Theta)$ can be reformulated as follows:

$$p(c,z|\vec{y},\Theta) = \frac{p(\vec{y}|c,z,\Theta)p(z|\Theta)p(c|\Theta)}{\sum_{c'}\int p(\vec{y}|c',z',\Theta)p(z|\Theta)p(c'|\Theta)dz} \tag{3.S40}$$

$$\overset{(3.S12)}{=} \frac{p(\vec{y}|c,\Theta)p(c|\Theta)\mathrm{Gam}(z;\alpha+\hat{y},\beta+\hat{W}_c)}{\sum_{c'} p(\vec{y}|c',\Theta)p(c'|\Theta)\int \mathrm{Gam}(z;\alpha+\hat{y},\beta+\hat{W}_c)dz} \tag{3.S41}$$

$$= \frac{p(\vec{y}|c,\Theta)p(c|\Theta)}{\sum_{c'} p(\vec{y}|c',\Theta)p(c'|\Theta)}\mathrm{Gam}(z;\alpha+\hat{y},\beta+\hat{W}_c) \tag{3.S42}$$

$$= p(c|\vec{y},\Theta)\mathrm{Gam}(z;\alpha+\hat{y},\beta+\hat{W}_c) \tag{3.S43}$$

File		Vertical crop		Horizontal crop		Rescale	Resulting size	
Name	ID	Start	End	Start	End		Height	Width
Dataset_02_SARS-CoV-2_007.tif	1	0	1032	0	1376	0.5	516	688
Dataset_02_SARS-CoV-2_009.tif	2	0	1032	0	1376	0.5	516	688
Dataset_02_SARS-CoV-2_038.tif	3	0	1032	0	1376	0.5	516	688
Dataset_03_SARS-CoV-2_043.tif	4	0	1032	165	1375	0.5	516	605
Dataset_03_SARS-CoV-2_070.tif	5	0	1032	0	1110	0.5	516	555
Dataset_03_SARS-CoV-2_080.tif	6	0	900	0	1270	0.5	450	635
Dataset_07_SARS-CoV-2_036.tif	7	0	1032	384	1376	0.5	516	496
Dataset_07_SARS-CoV-2_077.tif	8	0	1032	360	1376	0.5	516	508
Dataset_07_SARS-CoV-2_102.tif	9	0	1032	358	1374	0.5	516	508

Table 3.S1: File names and preprocessing settings for the selected TEM images of SARS-CoV-2 infected cell cultures (see Sec. 3.S2.1 for description).

Inserting these results into Eq. (3.S39), the data estimator yields to:

$$\mathbb{E}_{p(c,z|\vec{y},\Theta)}\big[zW_{cd}\big] = \sum_c \int zW_{cd}\, p(c,z|\vec{y},\Theta)dz \tag{3.S44}$$

$$= \sum_c W_{cd}\, p(c|\vec{y},\Theta) \int z\,\mathrm{Gam}(z;\alpha+\hat{y},\beta+\hat{W}_c)dz \tag{3.S45}$$

$$= \sum_c W_{cd}\, p(c|\vec{y},\Theta)\frac{\alpha+\hat{y}}{\beta+\hat{W}_c} \tag{3.S46}$$

$$= (\alpha+\hat{y})\mathbb{E}_{p(c|\vec{y},\Theta)}\left[\frac{W_{cd}}{\beta+\hat{W}_c}\right] \tag{3.S47}$$

3.S2 Details on Numerical Experiments

3.S2.1 Datasets

Transmission Electron Microscopy Images of SARS-CoV-2 Infected Cell Cultures. We consulted three publicly available datasets containing transmission electron microscopy (TEM) images of ultrathin plastic sections (45 nm and 60–70 nm) through extracellular SARS-CoV-2 particles in Vero cell cultures (Laue et al., 2020a,b,c) to select nine test images. From the original 1032×1376 images, we cut off areas with little structure (e.g., areas containing only noise) and, due to computational limitations of our implementation, particularly of ES3C (compare Drefs et al. (2022)), down-scaled the resolution by omitting every second pixel in width and height (using GIMP's 'Scale Image' method with 'Interpolation=None'). This enabled to train ES3C and GPMM with a sufficiently large number of generative fields and using patches sizes that captured sufficiently much virus structure. Table 3.S1 lists the file names of the selected images and the pixel indices of the selected image crops.

Fluorescence Microscopy Images. We downloaded the data sets FU-PN2V Convallaria, FU-PN2V Mouse actin and FU-PN2V Mouse nuclei from publicly available repositories (Krull et al., 2020b; Prakash et al., 2019a,b). The former two data sets each contained 100 images with 1024×1024 pixels, the latter 200 images with 512×512 pixels. In our 'zero-shot' denoising experiments, we applied BM3D, N2V, DivN, S2S, ES3C, and GPMM to a single randomly selected image from each data set (following a procedure used by Prakash et al. (2021b); in our case, the 75th, 41st, and 177th image from FU-PN2V Convallaria, FU-PN2V Mouse actin, and FU-PN2V Mouse nuclei, respectively, were selected). Following the publicly available DivNoising source code (Prakash et al., 2021a), we measured denoising performance in terms of PSNR, using the average image of each sequence as the target and the difference between the minimum and maximum amplitude of the target image as the peak value.

3.S2.2 Implementations and Hyperparameters

BM3D. We used the official BM3D Python software (Mäkinen et al., 2019) and executed the algorithm using 'stage_arg=BM3DStages.ALL_STAGES'. The algorithm requires as input an estimate of the noise level of the data. To compute such estimate, we used the method of Chen et al. (2015a) and the respective publicly available implementation (Chen et al., 2015b) together with default hyperparameters.

N2V. We used the official Noise2Void implementation (Krull et al., 2019b) and executed the algorithm with the configuration of the 'BSD68_reproducibility' example available in the respective GitHub repository.

DivN. We used the official DivNoising implementation (Prakash et al., 2021a) and executed the algorithm with the configuration of the 'Convallaria' and 'Mouse_nuclei' example available in the respective GitHub repository. For Mouse actin and SARS-CoV-2 experiments, we adopted the settings of the Convallaria example and changed image-specific settings accordingly. To create a noise model, we used the bootstrapping method based on the respective denoised outcome of the N2V algorithm, as also suggested in the repository.

S2S. We used the official Self2Self implementation (Quan et al., 2020b) and executed the algorithm with the configuration of the 'demo_denoising' script available in the respective GitHub repository. We set the dropout rate to 0.3 (as also suggested in examples in the 'demo_denoising' script), and the parameters 'sigma' and 'is_realnoisy' to '-1' and 'True', respectively. We modified the preprocessing pipeline by dividing the pixel amplitudes with the maximum pixel amplitude of a given input image rather than with the value 255 (compare utils.py/ line 26). Accordingly, we modified the postprocessing pipeline by multiplying the pixel amplitudes of the output image with the previously determined maximum value and then saved the image with single precision (compare demo_denoising.py/ line 56).

ES3C. To train ES3C on TEM images of SARS-CoV-2 infected cell cultures, we used a patch size and a dictionary size of $D = 6 \times 6$ and $H = 512$, respectively. For the evolutionary variational optimization (EVO; Drefs et al., 2022), we chose $S = |\mathcal{K}^{(n)}| = 40$ and used the 'fitparents-randflip' algorithm with $N_{\text{parents}} = 30$ and $N_{\text{children}} = N_{\text{generations}} = 1$. ES3C was trained for 150 epochs for each TEM image. For the fluorescence microscopy images, we used a slightly larger patch size of $D = 12 \times 12$, changed EVO hyperparameters to $S = 30$ and $N_p = 20$ to keep the computational demand at a reasonable level, and performed 200 training epochs. The algorithm was initialized as follows (compare Drefs et al., 2022):

Priors $\{\pi_h^{\text{init}}\}_{h=1}^H$ were uniformly randomly drawn from the interval $[0.1, 0.5]$. Latent means $\{\mu_h^{\text{init}}\}_{h=1}^H$ were sampled from a zero-mean unit variance Gaussian. The latent covariance matrix was set to the unit matrix, i.e. $\Psi^{\text{init}} = \mathbb{1}$. The columns of the matrix W^{init} were initialized with the mean of the data points pertubed by a small amount of Gaussian noise. The variance $(\sigma^{\text{init}})^2$ was set to the variance of the data points averaged over the observed dimensions. The initial states $\vec{s}$ in the sets $\{\mathcal{K}^{(n)}\}_{n=1}^N$ were sampled from a Bernoulli distribution with $p(s_h = 1) = 0.1$. We used an implementation that allows for parallelized execution on potentially large numbers of CPU cores[5]. Such parallelization alleviates the in general relatively high computational demand of the ES3C approach (see Drefs et al. (2022) for discussion).

GPMM. To train GPMM on TEM images of SARS-CoV-2 infected cell cultures, we used a patch size and a codebook size of $D = 6 \times 6$ (same value as used for ES3C) and $H = 1000$, respectively (compare Sec. 3.4). GPMM was trained for 200 epochs for each image. For the fluorescence microscopy images, we used $D = 5 \times 5$ patches (we found this value to yield the best performance of the algorithm in terms of denoising PSNR) and performed 500 training epochs for the Convallaria dataset and 400 for the Mouse actin and Mouse nuclei datasets. We initialized GPMM as follows: Priors $\{\pi_c^{\text{init}}\}_{c=1}^C$ were uniformly randomly drawn from the interval $]0, 1[$. The columns of the matrix W^{init} were initialized with the mean of the data points pertubed by a small amount of Poisson noise and scaled between 0 and 1. The initial parameters of the Gamma distribution were set to $\alpha^{\text{init}} = 100$ and $\beta^{\text{init}} = 1$. As for ES3C, we used a parallelized implementation of GPMM that allows for distributed execution (cf. Footnote 5).

3.S2.3 Further Results

Image Quality Ranking. Figure 3.S1 lists the individual scores of the image quality measures considered in Fig. 3.2 C for each test image.

ES3C. Figure 3.S2 depicts further colorized visualizations of variances of non-noisy pixel estimations including results obtained based on an unscaled SARS-CoV-2 TEM image. Figure 3.S3 presents results for 'zero-shot' enhancements of a SARS-CoV-1 EM recording, i.e., the virus that caused the outbreak of the SARS disease in 2002/2003.

DivNoising. For the experiments described in Sec. 3.2, we executed DivN using N2V-based bootstrapping for noise model estimation and a single noisy image for training. In further control experiments that we conducted with the fluorescence microscopy data, we also investigated applications of DivN with calibration data for noise model estimation and training on full image series. For calibration, we used the calibration images made publicly available by Krull et al. (2020b); Prakash et al. (2019a,b). In total, we performed four different types of experiments per image, which we here refer to as DivN_C^1, DivN_B^1, $\text{DivN}_\text{C}^\text{all}$, and $\text{DivN}_\text{B}^\text{all}$. The superscripts DivN^1 and DivN^all denote the variants of the algorithm that use training on a single and on all images of a given dataset, respectively; the subscripts DivN_C and DivN_B indicate noise model estimation using calibration data and bootstrapping,

---

[5] The source code will be made publicly available at `https://github.com/tvlearn` once the manuscript has been accepted for publication.

A CV

Algorithm	File ID								
	1	2	3	4	5	6	7	8	9
$\mathrm{BM3D}_{\hat{\sigma}}$	**0.983**	**1.071**	1.020	1.040	1.094	0.914	1.031	1.118	1.097
N2V	0.657	0.619	0.474	0.579	0.944	0.578	0.904	0.575	0.301
$\mathrm{DivN}^{1}_{\mathrm{B}}$	0.697	0.858	**2.062**	**1.816**	**1.507**	0.906	**2.158**	**1.680**	1.150
S2S	0.345	0.339	0.695	0.550	0.481	0.497	0.772	0.429	0.396
ES3C	0.894	1.059	1.503	1.411	1.468	**1.363**	1.610	1.297	**1.321**
GPMM	0.659	0.851	1.448	1.362	1.393	1.262	1.566	1.169	1.208

B InvSTD

Algorithm	File ID								
	1	2	3	4	5	6	7	8	9
$\mathrm{BM3D}_{\hat{\sigma}}$	0.339	0.344	0.367	0.306	0.156	0.220	0.123	0.459	0.713
N2V	0.509	0.525	0.542	0.467	0.266	0.455	0.270	0.586	0.714
$\mathrm{DivN}^{1}_{\mathrm{B}}$	0.865	0.882	0.632	0.723	0.792	0.953	0.516	0.786	0.961
S2S	**0.947**	**0.968**	**0.990**	**0.986**	**0.985**	**0.985**	**0.985**	**0.979**	**0.993**
ES3C	0.767	0.787	0.831	0.769	0.692	0.770	0.704	0.805	0.866
GPMM	0.867	0.867	0.819	0.842	0.823	0.866	0.849	0.872	0.944

C Entropy

Algorithm	File ID								
	1	2	3	4	5	6	7	8	9
$\mathrm{BM3D}_{\hat{\sigma}}$	0.960	0.972	0.912	0.948	0.971	0.958	**0.979**	0.948	0.802
N2V	0.972	0.979	0.920	0.951	0.978	0.971	0.974	0.950	**0.901**
$\mathrm{DivN}^{1}_{\mathrm{B}}$	0.975	0.980	0.947	0.973	0.988	**1.000**	0.958	0.960	0.818
S2S	**1.000**	**1.000**	**1.000**	**1.000**	0.980	0.965	0.932	**0.994**	0.753
ES3C	0.974	0.975	0.937	0.966	0.973	0.962	0.954	0.968	0.844
GPMM	0.983	0.992	0.970	0.986	**1.000**	0.989	0.954	0.984	0.862

D MeanBin

Algorithm	File ID								
	1	2	3	4	5	6	7	8	9
$\mathrm{BM3D}_{\hat{\sigma}}$	**0.416**	**0.431**	**0.541**	**0.594**	**0.762**	**0.673**	**0.799**	**0.349**	0.159
N2V	0.367	0.389	0.531	0.577	0.701	0.575	0.708	0.339	**0.217**
$\mathrm{DivN}^{1}_{\mathrm{B}}$	0.116	0.105	0.355	0.285	0.197	0.044	0.508	0.169	0.028
S2S	0.047	0.030	0.012	0.018	0.021	0.019	0.019	0.020	0.005
ES3C	0.166	0.157	0.157	0.216	0.305	0.212	0.295	0.136	0.076
GPMM	0.105	0.104	0.164	0.155	0.187	0.131	0.158	0.094	0.035

Figure 3.S1: Individual image quality scores for each of the nine test images related to the ranking of Fig. 3.2 C (see Tab. 3.S1 for image names associated to file IDs). Numbers were computed using publicly available source code (Koho et al., 2016b).

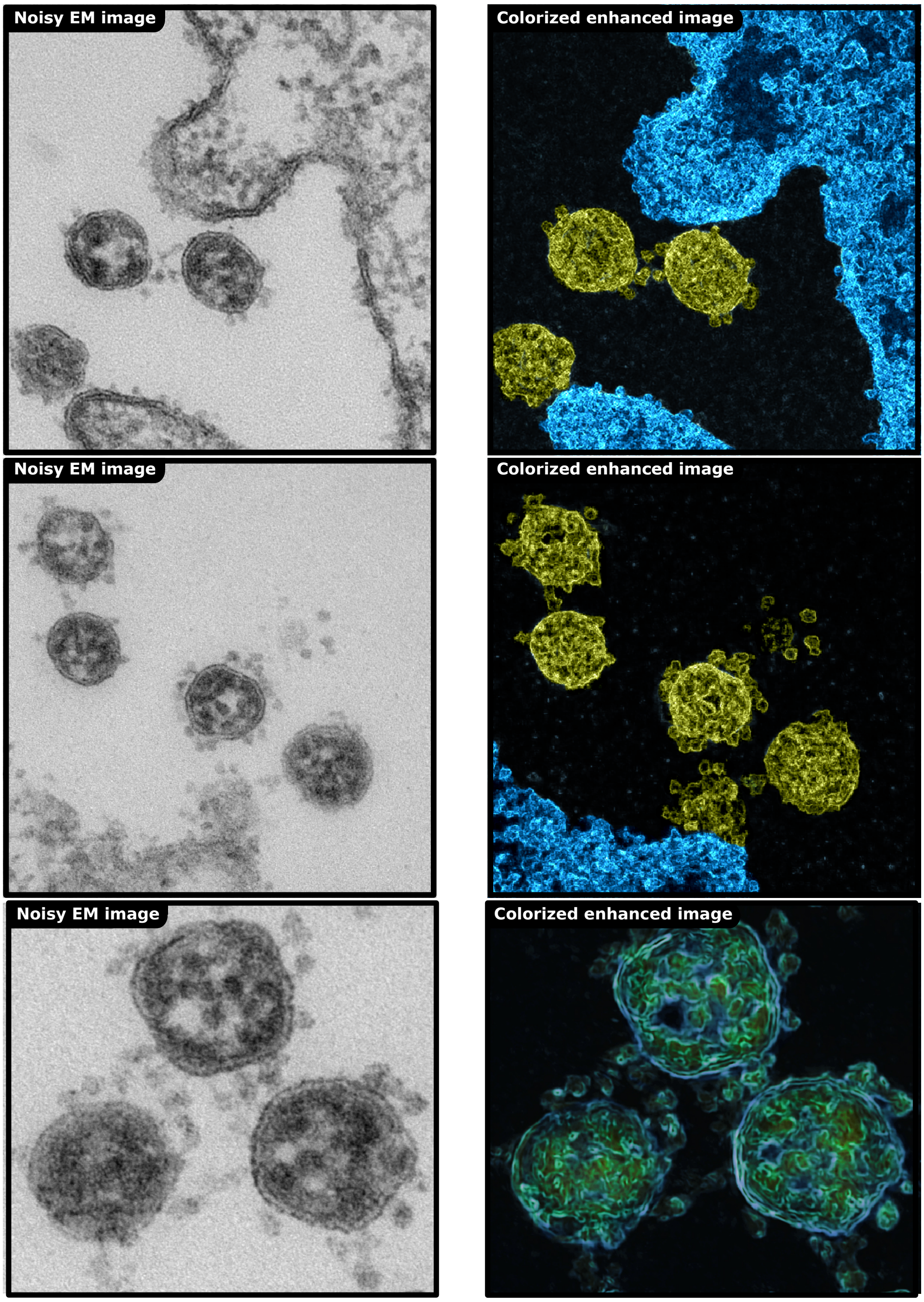


Figure 3.S2: Further examples of colorized visualizations of variances of non-noisy pixel estimations (compare Sec. 3.4). The top two colorized images use the same colorization as in Fig. 3.1 (see respective figure caption for details). The third image, showing three SARS-CoV-2 viruses, was manually colored in green and blue.

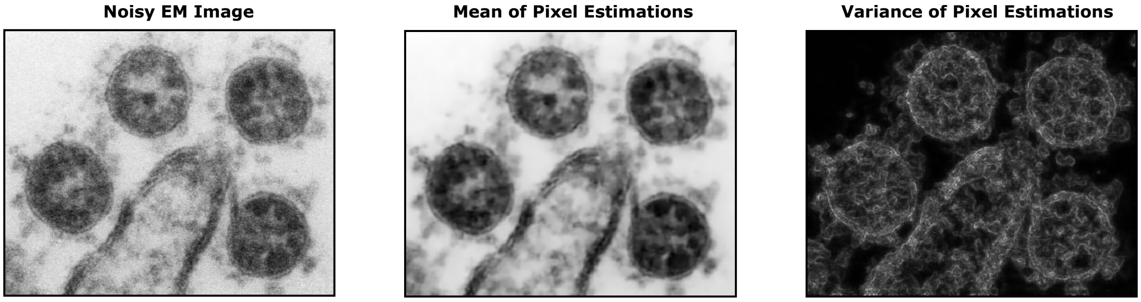


Figure 3.S3: 'Zero-shot' enhancements of electron microscopy images of SARS-CoV-1 viruses using ES3C trained on 9 × 9 TEM image patches (compare Sec. 3.4 and Fig. 3.5). The noisy input was obtained based on a publicly available dataset (Gelderblom et al.). We converted the original image to grayscale, cropped a section showing four viruses and scaled the resulting image to a size of 379 × 305 pixels.

	Convallaria	Mouse nuclei	Mouse actin
$\mathrm{DivN}^1_\mathrm{C}$	38.81 ± 0.06	36.11 ± 0.36	35.06 ± 0.25
$\mathrm{DivN}^1_\mathrm{B}$	38.46 ± 0.23	35.96 ± 0.20	35.12 ± 0.17
$\mathrm{DivN}^\mathrm{all}_\mathrm{C}$	39.65 ± 0.02	37.32 ± 0.12	35.82 ± 0.01
$\mathrm{DivN}^\mathrm{all}_\mathrm{B}$	39.05 ± 0.45	37.09 ± 0.06	35.80 ± 0.01

Table 3.S2: PSNR values (in dB) for denoising fluorescence microscopy images obtained in control experiments with different variants of the DivNoising algorithm (see text for details). Listed are averages and standard deviations over three executions of the algorithm per setting.

respectively. Table 3.S2 lists the PSNRs obtained with the four algorithm configurations for the three single test images considered in Fig. 3.4 and described in Sec. 3.S2.1.

Chapter 4

Truncated Variational Autoencoders

This chapter has appeared as: Jakob Drefs*, Enrico Guiraud*, Filippos Panagiotou, Jörg Lücke. Direct Evolutionary Optimization of Variational Autoencoders with Binary Latents. In Massih-Reza Amini et al. (Eds.): *Machine Learning and Knowledge Discovery in Databases*. ECML PKDD 2022. Lecture Notes in Computer Science, vol. 13715, pp. 357-372, 2023. Springer, Cham. `https://doi.org/10.1007/978-3-031-26409-2_22`. *Joint first authorship. Reproduced with permission from Springer Nature. The Springer Nature content included in this chapter is not covered by the open access license under which this thesis is published. The right to reuse the content requires the explicit permission of Springer Nature.

Abstract. Many types of data are generated at least partly by discrete causes. Deep generative models such as variational autoencoders (VAEs) with binary latents consequently became of interest. Because of discrete latents, standard VAE training is not possible, and the goal of previous approaches has therefore been to amend (i.e, typically anneal) discrete priors to allow for a training analogously to conventional VAEs. Here, we divert more strongly from conventional VAE optimization: We ask if the discrete nature of the latents can be fully maintained by applying a direct, discrete optimization for the encoding model. In doing so, we sidestep standard VAE mechanisms such as sampling approximation, reparameterization and amortization. Direct optimization of VAEs is enabled by a combination of evolutionary algorithms and truncated posteriors as variational distributions. Such a combination has recently been suggested, and we here for the first time investigate how it can be applied to a deep model. Concretely, we (A) tie the variational method into gradient ascent for network weights, and (B) show how the decoder is used for the optimization of variational parameters. Using image data, we observed the approach to result in much sparser codes compared to conventionally trained binary VAEs. Considering the for sparse codes prototypical application to image patches, we observed very competitive performance in tasks such as 'zero-shot' denoising and inpainting. The dense codes emerging from conventional VAE optimization, on the other hand, seem preferable on other data, e.g., collections of images of whole single objects (CIFAR etc), but less preferable for image patches. More generally, the realization of a very different type of optimization for binary VAEs allows for investigating advantages and disadvantages of the training method itself. And we here observed a strong influence of the method on the learned encoding with significant impact on VAE performance for different tasks.

Keywords: Variational Autoencoder, Evolutionary Optimization, Sparse Encoding, Variational Optimization, Binary Latents

Author Contributions. The original idea of exploring evolutionary variational optimization (as presented in Ch. 2) for discrete latent VAE training was developed by Enrico Guiraud (EG) and Jörg Lücke (JL). The initial literature research was carried out by all authors with Jakob Drefs (JD) having researched literature specific to gradient estimation techniques and having contributed the literature review related to image denoising and inpainting. Mathematical derivations (Sec. 4.S1) were carried out by EG and JL. EG designed and implemented the binary latent VAE model, including methods for optimizing model parameters using automatic differentiation tools, for use within the TVO software library (TVO developers, 2022, compare Ch. 2). Filippos Panagiotou (FP) contributed implementations for flexible decoder design and hyperparameter optimization routines. Methods for TVAE-based image denoising and inpainting were implemented by EG based on implementations for probabilistic data estimation and image reconstruction worked out by JD (compare Ch. 2). EG carried out all verification and scalability experiments, except for the reliability study with different evolutionary operator combinations (Fig. 4.S8) which was performed by JD. The results depicted in Fig. 4.1 were produced by EG and JD with contributions from FP (EG carried out the experiments with TVAE, JD performed the controls with EBSC, N2V, S2S, and GSVAE, and FP conducted the controls with VLAE). The results shown in Fig. 4.2 were produced by JD with contributions from FP (JD carried out the experiments with GSVAE, EBSC, TVAE, and ES3C, and FP those with VLAE). The results reported in Fig. 4.3 were produced by EG and JD. JD carried out the runtime analysis experiments (Tab. 4.S3) and all control experiments with CIFAR-10 data (Tab. 4.S4). Earlier versions of the manuscript were primarily written by EG and JL with major contributions from JD and partly appeared in Guiraud (2021) and in the preprint Guiraud et al. (2020). The final version of the manuscript that was accepted for publication at ECML 2022 was worked out by JD with input from JL. Figures and tables were created by EG and JD with input from JL. All authors discussed the results and approved the final version of the manuscript.

Section 4.S2.7 is not part of the published paper. It was worked out and written by JD as an extension of the accepted ECML 2022 manuscript.

Acknowledgments. We acknowledge funding by the German Research Foundation (DFG) under grant SFB 1330/1&2 (B2), ID 352015383 (JD), and the H4a cluster of excellence EXC 2177/1, ID 390895286 (JL), and by the German Federal Ministry of Education and Research (BMBF) through a Wolfgang Gentner scholarship (awarded to EG, ID 13E18CHA). Furthermore, computing infrastructure support is acknowledged through the HPC Cluster CARL at UOL (DFG, grant INST 184/157-1 FUGG) and the HLRN Alliance, grant ID nim00006.

4.1 Introduction and Related Work

Objects or edges in images are either present or absent, which suggests the use of discrete latents for their representation. There are also typically only few objects per image (of all possible objects) or only few edges in any given image patch (of all possible edges), which suggests a sparse code (e.g., Goodfellow et al., 2013; Olshausen and Field, 1996; Sheikh et al., 2014; Titsias and Lázaro-Gredilla, 2011). In order to model such and similar data, we study a novel, direct optimization approach for variational autoencoders (VAEs), which can learn discrete and potentially sparse encodings. VAEs (Kingma and Welling, 2014; Rezende et al., 2014) in their many different variations, have successfully been applied to a large number of tasks including semi-supervised learning (e.g., Maaløe et al., 2016), anomaly detection (e.g., Kiran et al., 2018) or sentence and music interpolation (Bowman et al., 2016; Roberts et al., 2018) to name just a few. The success of VAEs, in these tasks, rests on a series of methods that enable the derivation of scalable training algorithms to optimize VAE parameters. These methods were originally developed for Gaussian priors (Kingma and Welling, 2014; Rezende et al., 2014). To account for VAEs with discrete latents, novel methodology had to be introduced (we elaborate below and later in Sec. 4.S1).

The training objective of VAEs is derived from a likelihood objective, i.e., we seek model parameters Θ of a VAE that maximize the data log-likelihood, $\mathcal{L}(\Theta) = \sum_n \log\big(p_\Theta(\vec{x}^{\,(n)})\big)$, where we denote by $\vec{x}^{(1:N)}$ a set of N observed data points, and where $p_\Theta(\vec{x})$ denotes the modeled data distribution. Like conventional autoencoders (e.g., Bengio et al., 2007), VAEs use a deep neural network (DNN) to generate (or decode) observables $\vec{x} \in \mathbb{R}^D$, from a latent code $\vec{z}$. Unlike conventional autoencoders, however, the generation of data $\vec{x}$ is not deterministic but it takes the form of a probabilistic generative model. For VAEs with binary latents, we here consider a generative model with Bernoulli prior:

$$p_\Theta(\vec{z}) = \prod_h \big(\pi_h^{z_h}(1-\pi_h)^{(1-z_h)}\big), \quad p_\Theta(\vec{x} \mid \vec{z}) = \mathcal{N}\big(\vec{x}; \vec{\mu}(\vec{z}; W), \sigma^2 \mathbb{I}\big), \tag{4.1}$$

with $\vec{z} \in \{0,1\}^H$ being a binary code, $\vec{\pi} \in [0,1]^H$ being parameters of the prior on $\vec{z}$, and the non-linear function $\vec{\mu}(\vec{z}; W)$ being a DNN (that sets the mean of a Gaussian distribution). $p_\Theta(\vec{x} \mid \vec{z})$ is commonly referred to as *decoder*. The set of model parameters is $\Theta = \{\vec{\pi}, W, \sigma^2\}$, where W incorporates DNN weights and biases. Here, we assume homoscedasticity of the Gaussian distribution, but note that there is no obstacle to generalizing the model by inserting a DNN non-linearity that outputs a covariance matrix. Similarly, the algorithm could easily be generalized to different noise distributions should the task at hand call for it. Here, however, we will focus on the elementary VAEs given by Eq. (4.1).

For conventional and discrete VAEs, essentially all optimization approaches seek to approximately maximize the log-likelihood using the following series of methods (we elaborate in Sec. 4.S1):

(A) Instead of the log-likelihood, a variational lower bound (a.k.a. ELBO) is optimized.

(B) VAE posteriors are approximated by an *encoding model*, i.e., by a specific distribution (usually Gaussian) parameterized by one or more DNNs.

(C) The variational parameters of the encoder are optimized using gradient ascent on the lower bound, where the gradient is evaluated based on sampling and the reparameterization trick (which allows for sufficiently low-variance and yet efficiently computable estimates).

(D) Using samples from the encoder, the parameters of the decoder are optimized using gradient ascent on the variational lower bound.

Optimization procedures for VAEs with discrete latents follow the same steps (Points A to D). However, discrete or binary latents pose substantial further obstacles for learning, mainly due to the fact that backpropagation through discrete variables is generally not possible or biased (Bengio et al., 2013; Rolfe, 2017). Widely used stochastic gradient estimators for discrete random variables typically either exploit the REINFORCE (Williams, 1992) estimator in combination with variance control techniques (Dimitriev and Zhou, 2021; Dong et al., 2021; Kool et al., 2020; Liu et al., 2019) or reparameterization of continuous relaxations of discrete distributions (Jang et al., 2017; Maddison et al., 2017); reparameterization is also combined with REINFORCE (Grathwohl et al., 2018) or generalized to non-reparameterizable distributions (Cong et al., 2019). Also a recent approach by Berliner et al. (2022) is related to REINFORCE but uses natural evolution strategies (not to be confused with evolutionary optimization we apply here) to derive low-variance estimates for gradients (also see Sec. 4.1.1 and Sec. 4.S1). While accomplishing, in different senses, the goal of maintaining standard VAE training as developed for continuous latents (i.e., learning procedures and/or learning objectives that allow for gradient-based optimization of the encoder and decoder DNNs), gradient estimation methods usually apply significant amounts of methodology *additional* to the learning methods conventionally applied for VAE optimization (see Fig. 4.S2). These additional methods, their accompanying design decisions and used hyper-parameters increase the complexity of the system. Furthermore, the additional methods usually impact the learned representations. For instance, softening of discrete distributions, e.g., by using ‘Gumbel-softmax’ (Jang et al., 2017) or ‘tanh’ approximations (Fajtl et al., 2020) seems to favor dense codes. While dense codes (as also used by conventional VAEs and generative adversarial networks (Goodfellow et al., 2014)) can result in competitive performance for a subset of the above discussed tasks, other recent contributions point out advantages of sparse codes, e.g., in terms of disentanglement (Tonolini et al., 2020) or robustness (Paiton et al., 2020; Sulam et al., 2020).

In order to avoid adding methods for discrete latents to those already in place for standard VAEs, it may be reasonable to investigate more direct optimization procedures that do not require, e.g., a softening of discrete distributions or other mechanisms. Such a direct approach is challenging, however, because once DNNs are used to define the encoding model (as commonly done), we require methodologies for discrete latents to estimate gradients for the encoder (as done via sampling and reparameterization; see Points C and D). A direct optimization procedure, as we investigate here, consequently has to change VAE training substantially. For the data model of Eq. (4.1), we will maintain the variational setting (Point A) and a decoding model with DNNs as non-linearity. However, we will not use an encoding model parameterized by DNNs (Point B). Instead, the variational bound will be increased w.r.t. an implicitly defined encoding model which allows for an efficient discrete optimization. The procedure does not require gradients to be computed for the encoder such that discrete latents are addressed without the use of reparameterization trick and sampling approximations.

4.1.1 Related Work

In order to maintain the general VAE framework for encoder optimization in the case of discrete latents, different groups have suggested different possible solutions (for discussion of numerical evaluations of related approaches, see Sec. 4.S1.3): Rolfe (2017), for instance, extends VAEs with discrete latents by auxiliary continuous latents such that gradients can still be computed. Work on the concrete distribution (Maddison et al., 2017) or Gumbel-softmax distribution (Jang et al., 2017) proposes newly defined continuous distributions that contain discrete distributions as limit cases. Lorberbom et al. (2019) merge the Gumbel-Max reparameterization with the use of direct loss minimization for gradient estimation, enabling efficient training on structured latent spaces (also compare Paulus et al., 2021; Potapczynski et al., 2020, for further improved Gumbel-softmax versions). Furthermore, work, e.g., by van den Oord et al. (2017) combines VAEs with a vector quantization (VQ) stage in the latent layer. Latents become discrete through quantization but gradients for learning are adapted from latent values before they are processed by the VQ stage. Similarly, Tomczak and Welling (2018) use, what they call, (learnable) pseudo-inputs which determine a mixture distribution as prior, and the ELBO then contains an additional regularization for consistency between prior and average posterior. Tonolini et al. (2020) extend this work and introduce an additional DNN classifier which selects pseudo-inputs and whose weights are learned instead of the pseudo-inputs themselves. Tonolini et al. also argue for the benefits not only of discrete latents but of a sparse encoding in the latent layer in general. Fajtl et al. (2020) base their approach on a deterministic autoencoder and use a tanh-approximation of binary latents and projections to spheres in order to treat binary values. Targeting not only the optimization of discrete latent VAEs but also more general approaches such as probabilistic programming or general stochastic automatic differentiation, Bingham et al. (2019) and van Krieken et al. (2021) apply gradient estimators for discrete random variables which optimize surrogate losses (Schulman et al., 2015) derived based on the score function (Foerster et al., 2018) or other methods (van Krieken et al., 2021).

4.2 Direct Variational Optimization

Let us consider the variational lower bound of the likelihood. If we denote by $q^{(n)}_{\Phi}(\vec{z})$ the variational distributions with parameters $\Phi = (\Phi^{(1)}, \ldots, \Phi^{(N)})$, then the lower bound is given by:

$$\mathcal{F}(\Phi,\Theta) = \sum_{n} \mathbb{E}_{q^{(n)}_{\Phi}}\big[\log\big(p_{\Theta}(\vec{x}^{\,(n)}\,|\,\vec{z}\,)\,p_{\Theta}(\vec{z}\,)\big)\big] - \sum_{n} \mathbb{E}_{q^{(n)}_{\Phi}}\big[\log\big(q^{(n)}_{\Phi}(\vec{z}\,)\big)\big], \tag{4.2}$$

where we sum over all data points $\vec{x}^{(1:N)}$, and where $\mathbb{E}_{q^{(n)}_{\Phi}}\big[h(\vec{z})\big]$ denotes the expectation value of a function $h(\vec{z})$ w.r.t. $q^{(n)}_{\Phi}(\vec{z})$. The general challenge for the maximization of $\mathcal{F}(\Phi,\Theta)$ is the optimization of the encoding model $q^{(n)}_{\Phi}$. VAEs with discrete latents, as an additional challenge, have to address the question how gradients w.r.t. discrete latents can be computed. Seeking to avoid the problem of gradients w.r.t. discrete variables, we do *not* use a DNN for the encoding model. Consequently, we need to define an *alternative* encoding model $q^{(n)}_{\Phi}$, which has to remain sufficiently efficient. Considering prior work on generative models with discrete latents, variational distributions based on truncated posteriors offer themselves as such an alternative. Truncated posteriors have previously been considered to be functionally competitive (e.g., Hughes and Sudderth, 2016; Sheikh et al., 2014; Shelton et al., 2017). Most relevant for our purposes are very efficient and fully variational approaches that allow mixture models (Exarchakis et al., 2022; Hirschberger et al., 2022) and shallow generative approaches (Drefs et al., 2022) to be very efficiently scaled to large model sizes. In all these previous applications, optimization of truncated variational distributions used standard expectation maximization based on closed-form or pseudo-closed form M-steps available for the shallow decoder models considered. In the context of VAEs with discrete latents, the important question arising is if or how efficient optimization with truncated variational distributions can be performed for deep generative models.

4.2.1 Optimization of the Encoding Model

Encoder optimization is usually based on a reformulation of the variational bound of Eq. (4.2) given by:

$$\mathcal{F}(\Phi,\Theta) = \sum_n \mathbb{E}_{q_\Phi^{(n)}}\big[\log\big(p_\Theta(\vec{x}^{\,(n)}\,|\,\vec{z}\,)\big)\big] - \sum_n D_{\mathrm{KL}}\big[q_\Phi^{(n)}(\vec{z}\,); p_\Theta(\vec{z}\,)\big]. \tag{4.3}$$

For discrete latent VAEs, the variational distributions in Eq. (4.3) are commonly replaced by an amortized encoding model $q_\Phi(\vec{z}\,)$ with a DNN-based parameterization. When expectations w.r.t. $q_\Phi(\vec{z}\,)$ are approximated (as usual) via sampling, the encoder optimization requires gradient estimation methods for discrete random variables (cf. Sec. 4.1.1 and Sec. 4.S1). At this point truncated posteriors represent alternative variational distributions which avoid gradients w.r.t. discrete latents. Given a data point $\vec{x}^{\,(n)}$, a truncated posterior is the posterior itself truncated to a subset $\Phi^{(n)}$ of the latent space, i.e., for $\vec{z} \in \Phi^{(n)}$ applies:

$$q_\Phi^{(n)}(\vec{z}\,) := \frac{p_\Theta(\vec{z}\,|\,\vec{x}^{\,(n)})}{\sum\limits_{\vec{z}\,'\in\Phi^{(n)}} p_\Theta(\vec{z}\,'\,|\,\vec{x}^{\,(n)})} = \frac{p_\Theta(\vec{x}^{\,(n)}\,|\,\vec{z}\,)\,p_\Theta(\vec{z}\,)}{\sum\limits_{\vec{z}\,'\in\Phi^{(n)}} p_\Theta(\vec{x}^{\,(n)}\,|\,\vec{z}\,')\,p_\Theta(\vec{z}\,')} \tag{4.4}$$

while $q_\Phi^{(n)}(\vec{z}\,) = 0$ for $\vec{z} \notin \Phi^{(n)}$. The subsets $\Phi = \{\Phi^{(n)}\}_{n=1}^N$ are the variational parameters. Centrally for this work, truncated posteriors allow for a specific alternative reformulation of the bound. The reformulation recombines the entropy term of the original form (Eq. (4.2)) with the first expectation value into a single term, and is given by (see Drefs et al., 2022; Exarchakis et al., 2022; Hirschberger et al., 2022, for details):

$$\mathcal{F}(\Phi,\Theta) = \sum_n \log\Big(\sum_{\vec{z}\in\Phi^{(n)}} p_\Theta(\vec{x}^{\,(n)}\,|\,\vec{z}\,)\,p_\Theta(\vec{z}\,)\Big). \tag{4.5}$$

Thanks to the simplified form of the bound, the variational parameters $\Phi^{(n)}$ of the encoding model can now be sought using direct discrete optimization procedures. More concretely, because of the specific form of Eq. (4.5), pairwise comparisons of joint probabilities are sufficient to maximize the lower bound: if we update the set $\Phi^{(n)}$ for a given $\vec{x}^{\,(n)}$ by replacing a state $\vec{z}^{\,\mathrm{old}} \in \Phi^{(n)}$ with a state $\vec{z}^{\,\mathrm{new}} \notin \Phi^{(n)}$, then $\mathcal{F}(\Phi,\Theta)$ increases if and only if:

$$\log\big(p_\Theta(\vec{x}^{\,(n)},\vec{z}^{\,\mathrm{new}})\big) > \log\big(p_\Theta(\vec{x}^{\,(n)},\vec{z}^{\,\mathrm{old}})\big). \tag{4.6}$$

To obtain intuition for the pairwise comparison, consider the form of $\log(p_\Theta(\vec{x},\vec{z}\,))$ when inserting the binary VAE defined by Eq. (4.1). Eliding terms that do not depend on $\vec{z}$ we obtain:

$$\widetilde{\log}\,p_\Theta(\vec{x},\vec{z}\,) = -\|\vec{x} - \vec{\mu}(\vec{z}, W)\|^2 \;-\; 2\,\sigma^2 \sum_h \tilde{\pi}_h\, z_h, \tag{4.7}$$

where $\tilde{\pi}_h = \log\big((1-\pi_h)/\pi_h\big)$. The expression assumes an even more familiar form if we restrict ourselves for a moment to sparse priors with $\pi_h = \pi < \frac{1}{2}$, i.e., $\tilde{\pi}_h = \tilde{\pi} > 0$. The criterion defined by Eq. (4.6) then becomes:

$$\|\vec{x}^{\,(n)} - \vec{\mu}(\vec{z}^{\,\mathrm{new}}, W)\|^2 \,+\, 2\,\sigma^2\tilde{\pi}\;|\vec{z}^{\,\mathrm{new}}| < \;\; \|\vec{x}^{\,(n)} - \vec{\mu}(\vec{z}^{\,\mathrm{old}}, W)\|^2 \,+\, 2\,\sigma^2\tilde{\pi}\;|\vec{z}^{\,\mathrm{old}}|\,, \qquad (4.8)$$

where $|\vec{z}\,| = \sum_{h=1}^{H} z_h$ and $2\,\sigma^2\tilde{\pi} > 0$. Such functions are routinely encountered in sparse coding or compressive sensing (Eldar and Kutyniok, 2012): for each set $\Phi^{(n)}$, we seek those states $\vec{z}$ that are reconstructing $\vec{x}^{\,(n)}$ well while being sparse ($\vec{z}$ with few non-zero bits). For VAEs, $\vec{\mu}(\vec{z}, W)$ is a DNN and as such much more flexible in matching the distribution of observables $\vec{x}$ than can be expected from linear mappings. Furthermore, criteria like Eq. (4.8) usually emerge for maximum a-posteriori (MAP) training in sparse coding (Olshausen and Field, 1996). In contrast to MAP, however, here we seek a *population* of states $\vec{z}$ in $\Phi^{(n)}$ for each data point. It is a consequence of the reformulated lower bound defined by Eq. (4.5) that it remains optimal to evaluate joint probabilities (as for MAP) although the constructed population of states $\Phi^{(n)}$ can capture (unlike MAP training) rich posterior structures.

4.2.2 Evolutionary Search

But how can new states $\vec{z}^{\,\mathrm{new}}$ that optimize $\Phi^{(n)}$ be found efficiently in high-dimensional latent spaces? While blind random search for states $\vec{z}$ can in principle be used, it is not efficient; and adaptive search space approaches (Exarchakis et al., 2022; Hirschberger et al., 2022) are only defined for mixture models. However, a recently suggested combination of truncated variational optimization with evolutionary optimization (EVO; Drefs et al., 2022) is more generally defined for models with discrete latents, and does only require the efficient computation of joint probabilities $p_\Theta(\vec{x}, \vec{z})$. It can consequently be adapted to the VAEs considered here.

EVO optimization interprets the sets $\Phi^{(n)}$ of Eq. (4.4) as populations of binary genomes $\vec{z}$, and we can here adapt it by using Eq. (4.7) in order to assign to each $\vec{z} \in \Phi^{(n)}$ a fitness for evolutionary optimization. For the concrete updates, we use for each EVO iteration $\Phi^{(n)}$ as initial parent pool. We then apply the following genetic operators in sequence to suggest candidate states $\vec{z}^{\,\mathrm{new}}$ to update the $\Phi^{(n)}$ based on Eq. (4.6) (see Fig. 4.S3 for an illustration and Sec. 4.S1.2 and Drefs et al. (2022) for further details): Firstly, *parent selection* stochastically picks states from the parent pool. Subsequently, each of these states undergoes *mutation* which flips one or more entries of the bit vectors. Offspring diversity can be further increased by crossover operations. Using the *children* generated this way as the new parent pool, the procedure is repeated giving birth to multiple *generations* of candidate states. Finally, we update $\Phi^{(n)}$ by substituting individuals with low fitness with candidates with higher fitness according to Eq. (4.6). The whole procedure can be seen as an evolutionary algorithm (EA) with perfect memory or very strong elitism (individuals with higher fitness never drop out of the gene pool). Note that the improvement of the variational lower bound depends on generating as many as possible *different* children with high fitness over the course of training.

We point out that the EAs optimize each $\Phi^{(n)}$ independently, which allows for distributed execution s.t. the technique can be efficiently applied to large datasets in conjunction with stochastic or batch gradient descent on the model parameters Θ. The approach is, at the same time, memory intensive, i.e., all sets $\Phi^{(n)}$ need to be kept in memory (details in Sec. 4.S1.1). Furthermore, we point out the we here optimize variational parameters $\Phi^{(n)}$ of the encoding model which is fundamentally different from the approach of Hajewski and Oliveira (2020) (who use EAs to optimize DNN architectures of otherwise conventionally optimized VAEs with continuous latents).

4.2.3 Optimization of the Decoding Model

Using the previously described encoding model $q_{\Phi}^{(n)}(\vec{z})$, we can compute the gradient of Eq. (4.2) w.r.t. the decoder weights W which results in (see Sec. 4.S1 for details):

$$\vec{\nabla}_W \mathcal{F}(\Phi, \Theta) = -\frac{1}{2\sigma^2} \sum_n \sum_{\vec{z} \in \Phi^{(n)}} q_{\Phi}^{(n)}(\vec{z})\ \vec{\nabla}_W \|\vec{x}^{\,(n)} - \vec{\mu}(\vec{z}, W)\|^2. \tag{4.9}$$

The right-hand-side has salient similarities to standard gradient ascent for VAE decoders. Especially the familiar gradient of the mean squared error (MSE) shows that, e.g., standard automatic differentiation tools can be applied. However, the decisive difference is represented by the weighting factors $q_{\Phi}^{(n)}(\vec{z})$. Considering Eq. (4.4), we require all $\vec{z} \in \Phi^{(n)}$ to be passed through the decoder DNN in order to compute the $q_{\Phi}^{(n)}(\vec{z})$. As all states of $\Phi^{(n)}$ anyway have to be passed through the decoder for the MSE term of Eq. (4.9), the overall computational complexity is not higher than an estimation of the gradient with samples instead of states in $\Phi^{(n)}$ (but we use many states per $\Phi^{(n)}$, compare Tab. 4.S1).

To complete the decoder optimization, update equations for variance σ^2 and prior parameters $\vec{\pi}$ can be computed in closed-form (compare, e.g., Shelton et al., 2011) and are given by:

$$\begin{aligned} \sigma^2 &= \frac{1}{DN} \sum_n \sum_{\vec{z} \in \Phi^{(n)}} q_{\Phi}^{(n)}(\vec{z})\ \|\vec{x}^{\,(n)} - \vec{\mu}(\vec{z}, W)\|^2, \\ \vec{\pi} &= \frac{1}{N} \sum_n \sum_{\vec{z} \in \Phi^{(n)}} q_{\Phi}^{(n)}(\vec{z})\ \vec{z}. \end{aligned} \tag{4.10}$$

The full training procedure for binary VAEs is summarized in Alg. 2. We refer to the binary VAE trained with this procedure as *Truncated Variational Autoencoder* (TVAE) because of the applied truncated posteriors[6].

4.3 Numerical Experiments

TVAE can flexibly learn prior parameters $\vec{\pi}$, and if low values for the π_h are obtained (which will be the case), the code is sparse. The prototypical application domain to study sparse codes is image patch data (Goodfellow et al., 2013; Olshausen and Field, 1996). We consequently use such data to investigate sparsity, scalability and efficiency on benchmarks. For all numerical experiments, we employ fully connected DNNs $\vec{\mu}(\vec{z}; W)$ for the decoder (compare Fig. 4.S4); the exact network architectures and activations used are listed in Tab. 4.S1. The DNN parameters are optimized based on Eq. (4.9) using mini-batches and the Adam optimizer (details in Sec. 4.S2.1).

---

[6] Source code available at `https://github.com/tvlearn`.

Algorithm 2: Training Truncated Variational Autoencoders (TVAE)

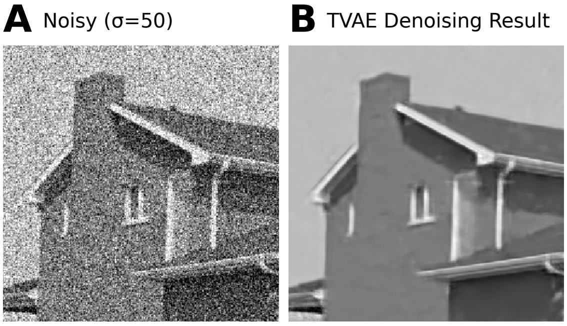
```
Initialize model parameters $\Theta = (\vec{\pi}, W, \sigma^2)$
Initialize each $\Phi^{(n)}$ with $S$ distinct latent states
repeat
  for all batches in dataset do
    for sample $n$ in batch do
      $\Phi^{\mathrm{new}} = \Phi^{(n)}$
      for all generations do
        $\Phi^{\mathrm{new}} = \mathrm{mutation}\left(\mathrm{selection}\left(\Phi^{\mathrm{new}}\right)\right)$
        $\Phi^{(n)} = \Phi^{(n)} \cup \Phi^{\mathrm{new}}$
      end for
      Define new $\Phi^{(n)}$ by selecting the $S$ fittest elements in $\Phi^{(n)}$ using Eq. (4.6)
    end for
    Use Adam to update $W$ using Eq. (4.9)
  end for
  Use Eq. (4.10) to update $\vec{\pi}$, $\sigma^2$
until parameters $\Theta$ have sufficiently converged
```

4.3.1 Verification and Scalability

After first verifying that the procedure can recover generating parameters using ground-truth data (see Sec. 4.S2.2), we trained TVAE on $N = 100,000$ whitened image patches of $D = 16 \times 16$ pixels (van Hateren and van der Schaaf, 1998) using two different decoder architectures, namely a shallow, linear decoder with $H = 300$ binary latents, and second, a deep non-linear decoder with a 300-300-256 architecture (i.e., $H = 300$ binary latents and two hidden layers with 300 and 256 units, respectively; details in Sec. 4.S2.3). For both linear and non-linear TVAE, we observed a sparse encoding with on average $\frac{\sum_h \pi_h}{H} = \frac{20.3}{300}$ and $\frac{\sum_h \pi_h}{H} = \frac{28.5}{300}$ active latents across data points, respectively. We observed sparse codes also when we varied the parameter initialization and further modified the decoder DNN architecture. As long as decoder DNNs were of small to intermediate size, we observed efficient scalability to large latent spaces (we went up to $H = 1,000$). Compared to linear decoders, the main additional computational cost is given by passing the latent states in the $\Phi^{(n)}$ sets through the decoder DNN instead of just through a linear mapping. The sets of states (i.e., the bitvectors in $\Phi^{(n)}$) could be kept small, at size $S = |\Phi^{(n)}| = 64$, such that $N \times (|\Phi^{(n)}| + |\Phi^{(n)}_{\mathrm{new}}|)$ states had to be evaluated per epoch. This compares to $N \times M$ states that would be used for standard VAE training (given M samples are drawn per data point). In contrast to standard VAE training, the sets $\Phi^{(n)}$ have to be remembered across iterations. For very large datasets, the additional $\mathcal{O}(N \times |\Phi^{(n)}| \times H)$ memory demand can be distributed over compute nodes, however.

4.3.2 Denoising - Controlled Conditions

Due to its non-amortized encoding model, the computational load of TVAE increases more strongly with data points compared to amortized training. Consequently, tasks such as disentanglement of features using high-dimensional input data, large DNNs, and small latent spaces are not a regime where the approach can be applied efficiently. With this in mind, we focused on tasks with relatively few data for which an as effective as possible



C Controlled Conditions

	σ=15	σ=25	σ=50
MTMKL	**34.29**	31.88	28.08
GSC	32.68	31.10	28.02
EBSC	32.20 ± .08	31.14 ± .13	28.50 ± .12
TVAE	34.27 ± .02	**32.65** ± .06	**29.61** ± .02

D Different Optimized Hyperparameters

	σ=15	σ=25	σ=50
N2V*	31.86 ± .37	29.34 ± .37	24.24 ± .57
S2S	34.86 ± .13	32.87 ± .08	29.95 ± .26
GSVAE-B	29.39 ± .02	28.32 ± .28	25.73 ± .14
GSVAE-C	30.94 ± .17	29.83 ± .04	26.75 ± .02
VLAE	34.47 ± .05	32.44 ± .05	29.16 ± .07
MTMKL	34.29	31.88	28.08
GSC	33.78	32.01	28.35
S5C	33.50	32.08	28.35
EBSC	33.66 ± .04	32.40 ± .07	28.98 ± .06
ES3C	**34.90** ± .04	**33.15** ± .06	29.83 ± .07
TVAE	34.27 ± .02	32.65 ± .06	**29.98** ± .05
N2V†	33.91	32.10	28.94
KSVD	34.32	32.15	27.95
WNNM	**35.13**	**33.22**	**30.33**
BM3D	34.94	32.86	29.37
EPLL	34.17	32.17	29.12
BDGAN	34.57	33.28	30.61
DPDNN	**35.40**	**33.54**	**31.04**

Figure 4.1: Denoising results for House. **C** compares PSNRs (in dB) obtained with different 'zero-shot' models using a fixed patch size and number of latents (means and standard deviations were computed over three runs with independent noise realizations, see text for details). **D** lists PSNRs for different algorithms with different optimized hyper-parameters. The top category only requires the noisy image ('zero-shot' setting). The middle uses additional information such as noise level (KSVD, WNNM, BM3D) or additional noisy images with matched noise level (N2V†). The bottom three algorithms use large clean datasets. The highest PSNR per category is marked bold, and the overall highest PSNR is bold and underlined. **B** depicts the denoised image obtained with TVAE for $\sigma = 50$ in the best run (PSNR=30.03 dB).

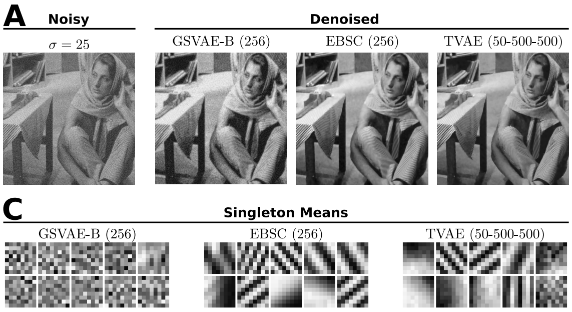
optimization is required, and for which advantages of a direct optimization can be expected. As one such task, we here considered 'zero-shot' image denoising. To apply TVAE in a 'zero-shot' setting (in which no additional information besides the noisy image is available, e.g., Imamura et al., 2019; Shocher et al., 2018), we trained the model on overlapping patches extracted from a given noisy image and subsequently applied the learned encoding to estimate non-noisy image pixels (details in Sec. 4.S2.4). In general, denoising represents a canonical benchmark for evaluating image patch models, and approaches exploiting sparse encodings have shown to be particularly well suited (compare, e.g., Mairal et al., 2008; Sheikh et al., 2014; Zhou et al., 2009). The 'zero-shot' setting has recently become popular also because the application of conventional DNN-based approaches has shown to be challenging (see discussion in Sec. 4.S2.4).

One denoising benchmark, which allows for an extensive comparison to other methods is the House image. Standard benchmark settings for this image make use of additive Gaussian white noise with standard deviations $\sigma \in \{15, 25, 50\}$ (Fig. 4.1 A). First, consider the comparison in Fig. 4.1 C where all models used the same patch size of $D = 8 \times 8$ pixels and $H = 64$ latent variables (details in Sec. 4.S2.4). Fig. 4.1 C lists the different approaches in terms of the standard measure of peak signal-to-noise ratio (PSNR). Values for MTMKL (Titsias and Lázaro-Gredilla, 2011) and GSC (Sheikh et al., 2014) were taken from the respective original publications (which both established new state-of-the-art results when first published); for EBSC (Drefs et al., 2022), we produced PSNRs ourselves by running publicly available source code (cf. Sec. 4.S2.4). As can be observed, TVAE significantly improves performance for high noise levels; the approach is able to learn the best data representation for denoising and establishes new state-of-the-art results in this controlled setting (i.e., fixed D and H). The decoder DNN of TVAE provides the decisive performance advantage: TVAE significantly improves performance compared to EBSC (which can be considered as an approach with a shallow, linear decoding model), confirming that the high lower bounds of TVAE on natural images (compare Fig. 4.S9) translate into improved performance on a concrete benchmark. For $\sigma = 25$ and $\sigma = 50$, TVAE also significantly improves on MTMKL, and GSC, which are both based on a spike-and-slab sparse coding (SSSC) model (also compare Goodfellow et al., 2013). Despite the less flexible Bernoulli prior, the decoder DNN of TVAE provides the highest PSNR values for high noise levels.

4.3.3 Denoising - Uncontrolled Conditions

To extend the comparison, we next evaluated denoising performance without controlling for equal conditions, i.e., we also included approaches in our comparison that use large image datasets and/or different patch sizes for training (including multi-scale and whole image processing). Note that different approaches may employ very different sets of hyper-parameters that can be optimized for denoising performance (e.g., patch and dictionary sizes for sparse coding approaches, or network and training scheme hyper-parameters for DNN approaches). By allowing for comparison in this less controlled setting, we can compare to a number of recent approaches including large DNNs trained on clean data and training schemes specifically targeted to noisy training data. See Fig. 4.1 D for an extensive PSNR overview with results for other algorithms cited from their corresponding original publications if not stated otherwise. PSNRs for S5C originate from Sheikh and Lücke (2016), GSVAE-B, EBSC and ES3C from Drefs et al. (2022), and WNNM and EPLL from Zhang et al. (2017). For Noise2Void (N2V; Krull et al., 2019a), Self2Self (S2S; Quan et al., 2020a), GSVAE-C (Jang et al., 2017), and VLAE (Park et al., 2019a), we produced results ourselves by applying publicly available source code (details in Sec. 4.S2.4). Note that the best performing approaches in Fig. 4.1 D were trained on noiseless data: EPLL (Zoran and Weiss, 2011), BDGAN (Zhu et al., 2019) and DPDNN (Dong et al., 2019) all make use of clean training data (typically hundreds of thousands of data points or more). EPLL, KSVD (Elad and Aharon, 2006), WNNM (Gu et al., 2014) and BM3D (Dabov et al., 2007) leverage a-priori noise level information (these algorithms use the ground-truth noise level of the test image as input parameter). As noisy data is very frequently occurring, lifting the requirement of clean data has been of considerable recent interest with approaches such as Noise2Noise (N2N; Lehtinen et al., 2018), N2V, and S2S having received considerable attention.

Considering Fig. 4.1 D, first note that TVAE consistently improves PSNRs of N2V, also when comparing to a variant trained on external data with matched-noise level (N2V† in Fig. 4.1 D). At high noise level ($\sigma = 50$), PSNRs of TVAE represent state-of-the-art performance in the 'zero-shot' category (Fig. 4.1 D, top); compared to methods which exploit additional a-priori information (Fig. 4.1 D, middle and bottom), the denoising performance of TVAE (at high noise level) is improved only by WNNM, BDGAN and DPDNN. At lower noise levels, TVAE still performs competitively in the 'zero-shot' setting, yet highest PSNRs are obtained by other methods (S2S and ES3C). Figure 4.1 D reveals that TVAE can improve on two competing VAE approaches, namely GSVAE (which uses



B

Model	Sparsity	Lower Bound	PSNR
GSVAE-B (256)	$\frac{128}{256}$	39.02	25.06
GSVAE-B (256-256-512)	$\frac{128}{256}$	38.68	25.06
GSVAE-C ((32×8)-256-512)	n/a	45.66	26.18
EBSC (256)	$\frac{1.5}{256}$	43.85	28.37
TVAE (256-256-512)	$\frac{2.3}{256}$	46.86	29.96
TVAE (50-500-500)	$\frac{2.6}{50}$	**47.98**	**30.03**
VLAE (50-500-500)	n/a	45.93*	28.83
ES3C (256)	$\frac{2.0}{256}$	**46.09**	**29.69**

Figure 4.2: Data encodings and denoising results for Barbara obtained with generative model approaches and different decoding models. In **B**, approaches with binary (top) and continuous (bottom) latents are separated. EBSC and ES3C are considered as using a shallow, linear decoder. Listed are best performances of several runs of each algorithm[7]. *VLAE uses importance sampling-based log-likelihood estimation. **C** compares decoder outputs for singleton (i.e., one-hot) input vectors. See Sec. 4.S2.4 for details.

Gumbel-softmax-based optimization for discrete latents) and VLAE (which uses continuous latents and Gaussian posterior approximations). For more systematic comparison, we applied the VAE approaches using identical decoder architectures and identical patch sizes (details in Sec. 4.S2.4). As striking difference between the approaches, we observed GSVAE to learn a significantly denser encoding compared to TVAE. Furthermore, we observed that the sparse encodings of TVAE resulted in strong performance not only in terms of denoising PSNR but also in terms of lower bounds (see Fig. 4.2).

4.3.4 Inpainting

Finally, we applied TVAE to 'zero-shot' inpainting tasks. For TVAE, the treatment of missing data is directly available given the probabilistic formulation of the model. Concretely, when evaluating log-joint probabilities of a data-point, missing values are treated as unknown observables (details in Sec. 4.S2.5). In contrast, amortized approaches need to specify how the deterministic encoder DNNs should treat missing values. Figure 4.3 evaluates performance of TVAE on two standard inpainting benchmarks with randomly missing pixels. Methods compared to include MTMKL, BPFA (Zhou et al., 2012), ES3C, the method of Papyan et al. (2017), DIP (Ulyanov et al., 2018), PLE (Yu et al., 2012), and IRCNN (Chaudhury and Roy, 2017). PLE uses the noise level as a-priori information, and IRCNN is trained on external clean images. On House, TVAE improves the performance of Papyan et al. and BPFA; highest PSNRs for this benchmark are obtained by DIP (which, in contrast to TVAE, is not permutation invariant and uses large U-Nets) and ES3C (which is based on a SSSC model and EVO-based training). On Castle, PSNRs of TVAE are higher in comparison to SSSC-based BPFA (for 50% missing pixels) and IRCNN.

---

[7] We restricted ourselves here to a comparison of models with at most 256 latents, due to computational limitations at the time of conducting this evaluation. We acknowledge that for a higher number of latents and a larger patch size (i.e., $H = 512$ and $D = 11 \times 11$, respectively), PSNRs of ES3C showed to increase to 30.19 dB on average over multiple runs as illustrated in Fig. 2.6 (also compare Tab. 2.S2), which improves the best performance of TVAE reported here. The numbers are, at the same time, not directly comparable since, in contrast to the setting of ES3C for Fig. 2.6, we here used rescaled pixel amplitudes (compare the details of the Comparison to Deep Generative Models in Sec. 4.S2.4).

	House 50%
Papyan et al.	34.58
BPFA	38.02
DIP	39.16
ES3C	**39.59**
TVAE	38.56

	Castle 50%	Castle 80%
MTMKL	n/a	28.94
BPFA	36.45	29.12
ES3C	**38.23**	**29.66**
TVAE	37.33	28.93
PLE	**38.34**	**30.07**
IRCNN	n/a	28.74

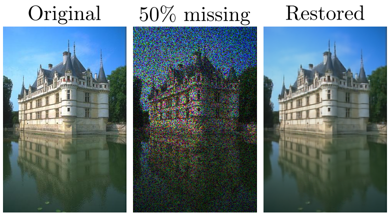


Figure 4.3: Inpainting results for House (50% missing pixels) and Castle (50% and 80% missing; top group lists 'zero-shot' approaches). PSNR for Papyan et al. as reported in Ulyanov et al. (2018). Depicted is TVAE's restoration for 50% missing pixels (sparsity $\frac{\sum_h \pi_h}{H} = \frac{10.56}{512}$).

4.4 Discussion

We investigated a novel approach built upon Evolutionary Variational Optimization (Drefs et al., 2022) to train VAEs with binary latents. Compared to all previous optimizations suggested for VAEs with discrete latents, the approach followed here differs the most substantially from conventional VAE training. While all other VAEs maintain amortization and reparameterization as key elements, the TVAE approach instead uses a direct and non-amortized optimization. Recent work using elementary generative models such as mixtures and shallow models (Drefs et al., 2022; Exarchakis et al., 2022) have made considerations of direct VAE optimization possible for intermediately large scales. A conceptual advantage of the here developed approach is its concise formulation (compare Fig. 4.S2) with fewer algorithmic elements, fewer hyperparameters and fewer model parameters (e.g., no parameters of encoder DNNs). Functional advantages of the approach are its avoidance of an amortization gap (e.g., Cremer et al., 2018; Kim et al., 2018), its ability to learn sparse codes, and its generality (it does not use a specific posterior model, and can be applied to other noise models, for instance). However, non-amortized approaches do in general have the disadvantage of a lower computational efficiency: an optimization of variational parameters for each data point is more costly (Tab. 4.S3). Conventional amortized approaches (for discrete or continuous VAEs) are consequently preferable for large-scale data sets and for the optimization of large, intricate DNNs. There are, however, alternatives such as transformers (which can use >150M parameters) or diffusion nets, which both are considered to perform more strongly than VAEs for large-scale settings and density modeling (Child et al., 2019; Kingma et al., 2021, for recent comparisons).

At the same time, direct discrete optimization can be feasible and can be advantageous. For image patch data, for instance, we showed that TVAEs with intermediately large decoder DNNs perform more strongly than Gumbel-softmax VAEs (GSVAE), and TVAEs are also outperforming a recent continuous VAE baseline (VLAE; Figs. 4.1 and 4.2). The stronger performance of TVAE is presumably, at least in part, due to the approach not being subject to an amortization gap, due to it avoiding factored variational distributions, and, more generally, due to the emerging sparse codes being well suited for modeling image patch data. In comparison, the additional methods to treat discrete latents in GSVAE seem to result in dense codes with significantly lower performance than TVAE. Compared to GSVAE, the VLAE approach, which uses standard non-sparse (i.e. Gaussian) latents, is more competitive on the benchmarks we considered. The reason is presumably that VLAE's continuous latents are able to better capture component intensities in image patches. This advantage does not outweigh the advantages of sparse codes learned by TVAE, however. If sparse codes and continuous latents are combined, the example of ES3C shows that strong performances can be obtained (Figs. 4.1 to 4.3). For the here considered binary latents, however, a linear decoder (compare EBSC) is much inferior to a deep decoder (Figs. 4.1, 4.2 and 4.S9), which suggests future work on VAEs with more complex, sparse priors if the goal is to improve ‘zero-shot’ denoising and inpainting. Dense codes are notably not necessarily disadvantageous for image data. On the contrary, for datasets with many images of single objects like CIFAR, the dense codes of GSVAE and also of VLAE are, in terms of ELBO values, similar or better compared to TVAE (Tab. 4.S4). The suitability of sparse versus dense encoding consequently seems to highly depend on the data, and here we confirm the suitability of sparse codes for image patches. In addition to learning sparse codes, direct optimization can have further advantages compared to conventional training. One such advantage is highlighted by the inpainting task: in contrast to other (continuous or discrete) VAEs, it is not required to additionally specify how missing data shall be treated by an encoder DNN (compare Sec. 4.S2.5).

We conclude that direct discrete optimization can, depending on the data and task, serve as an alternative for training discrete VAEs. In a sense, the approach can be considered more brute-force than conventional amortized training: direct optimization is slower but at scales at which it can be applied, it is more effective. To our knowledge, the approach is also the first training method for discrete VAEs not using gradient optimization of encoder models, and can thus contribute to our understanding of how good representations can be learned by different approaches.

Supplementary Material

4.S1 Details of Encoder and Decoder Optimization

See Fig. 4.S1 for a graphical comparison between the decoding models of a vanilla VAE and the here considered binary VAE defined by Eq. (4.1). Figure 4.S2 graphically illustrates different steps to optimize standard VAEs, and additional steps suggested by different contributions in order to optimize discrete VAEs. For the optimization of the binary VAE (Eq. (4.1)), consider the original form of the lower bound given by Eq. (4.2). When taking derivatives of $\mathcal{F}(\Phi,\Theta)$ w.r.t. Θ we can ignore the entropy term[8]. For the binary VAE model of Eq. (4.1), the gradient of the lower bound w.r.t. W is then given by:

$$\begin{aligned}
\vec{\nabla}_W \mathcal{F}(\Phi,\Theta) &= \sum_n \vec{\nabla}_W \mathbb{E}_{q_\Phi^{(n)}}\big[\log\big(p_\Theta(\vec{x}^{\,(n)}\,|\,\vec{z})\,p_\Theta(\vec{z})\big)\big] \\
&= \sum_n \vec{\nabla}_W \mathbb{E}_{q_\Phi^{(n)}}\big[\log\big(p_\Theta(\vec{x}^{\,(n)}\,|\,\vec{z})\big)\big] \\
&= \sum_n \vec{\nabla}_W \mathbb{E}_{q_\Phi^{(n)}}\big[\log\big(\mathcal{N}(\vec{x}^{\,(n)};\vec{\mu}(\vec{z},W),\sigma^2\mathbb{I})\big] \\
&= -\frac{1}{2\sigma^2}\sum_n \vec{\nabla}_W \sum_{\vec{z}\in\Phi^{(n)}} q_\Phi^{(n)}(\vec{z})\|\vec{x}^{\,(n)}-\vec{\mu}(\vec{z},W)\|^2,
\end{aligned} \tag{4.S1}$$

where rearranging leads to the final expression:

$$\vec{\nabla}_W \mathcal{F}(\Phi,\Theta) = -\frac{1}{2\sigma^2}\sum_n \sum_{\vec{z}\in\Phi^{(n)}} q_\Phi^{(n)}(\vec{z})\vec{\nabla}_W\|\vec{x}^{\,(n)}-\vec{\mu}(\vec{z},W)\|^2. \tag{4.9 revisited}$$

The weighting factors $q_\Phi^{(n)}(\vec{z})$ are, by using Eqs. (4.1) and (4.4), given by:

$$\begin{aligned}
q_\Phi^{(n)}(\vec{z}) &= \frac{p_\Theta(\vec{x}^{\,(n)}\,|\,\vec{z})\,p_\Theta(\vec{z})}{\sum_{\vec{z}\,'\in\Phi^{(n)}} p_\Theta(\vec{x}^{\,(n)}\,|\,\vec{z}\,')\,p_\Theta(\vec{z}\,')} \\
&= \frac{\exp\big(-\frac{1}{2\sigma^2}\,\|\vec{x}^{\,(n)}-\vec{\mu}(\vec{z},W)\|^2 - \tilde{\vec{\pi}}^{T}\vec{z}\big)}{\sum\limits_{\vec{z}\,'\in\Phi^{(n)}} \exp\big(-\frac{1}{2\sigma^2}\,\|\vec{x}^{\,(n)}-\vec{\mu}(\vec{z}\,',W)\|^2 - \tilde{\vec{\pi}}^{T}\vec{z}\,'\big)}
\end{aligned} \tag{4.S2}$$

for all $\vec{z}\in\Phi^{(n)}$, with $\tilde{\pi}_h = \log\big(\frac{1-\pi_h}{\pi_h}\big)$. Note that the $q_\Phi^{(n)}(\vec{z})$ are evaluated at the current values of the parameters Θ, they are therefore treated as constant, e.g., for the gradient w.r.t. W.

---

[8] For our choice of variational distributions, it is not trivial that the entropy term actually can be ignored because the encoding model $q_\Phi(\vec{z};\vec{x})$ in Eq. (4.4) is defined in terms of the decoding model and its parameters. For truncated distributions, however, it can be shown that the entropy term can still be ignored (Lücke, 2019).

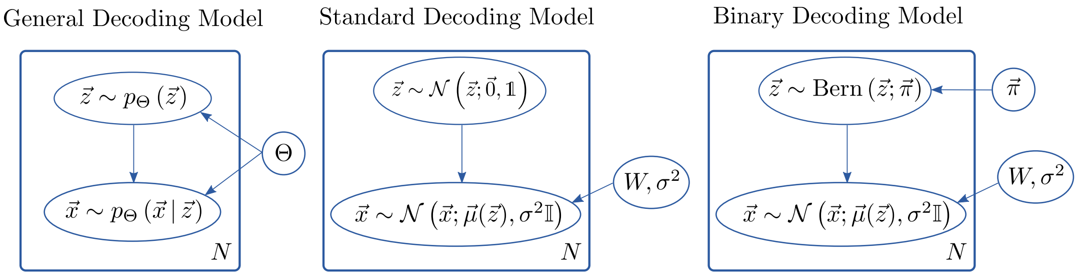

General Decoding Model
$\vec{z} \sim p_\Theta(\vec{z})$
$\vec{x} \sim p_\Theta(\vec{x} \mid \vec{z})$
Θ
N
Standard Decoding Model
$\vec{z} \sim \mathcal{N}(\vec{z}; \vec{0}, \mathbb{1})$
$\vec{x} \sim \mathcal{N}(\vec{x}; \vec{\mu}(\vec{z}), \sigma^2 \mathbb{I})$
W, σ^2
N
Binary Decoding Model
$\vec{z} \sim \mathrm{Bern}(\vec{z}; \vec{\pi})$
$\vec{\pi}$
$\vec{x} \sim \mathcal{N}(\vec{x}; \vec{\mu}(\vec{z}), \sigma^2 \mathbb{I})$
W, σ^2
N

Figure 4.S1: Decoding models. **Left**: Generic and general VAE decoding model. **Center**: VAE with standard continuous latents. **Right**: VAE with binary latents of Eq. (4.1).

It may be interesting to compare the gradient estimate given by Eq. (4.9) to the gradient estimate of conventional VAE training. For this, consider a standard encoder given by an amortized variational distribution which we shall denote by $\tilde{q}_\Phi^{(n)}(\vec{z})$. The distribution $\tilde{q}_\Phi^{(n)}(\vec{z})$ could be a Gaussian whose mean and variance are set by passing data point $\vec{x}^{(n)}$ through encoder DNNs. For discrete VAEs, $\tilde{q}_\Phi^{(n)}(\vec{z})$ can be thought of as an analog discrete distribution. If we now take gradients of Eq. (4.3) w.r.t. W and estimate using samples from $\tilde{q}_\Phi^{(n)}(\vec{z})$, we obtain the familiar form:

$$\begin{aligned}\vec{\nabla}_W \mathcal{F}(\Phi,\Theta) &= \sum_n \vec{\nabla}_W \mathbb{E}_{\tilde{q}_\Phi^{(n)}}\left[\log\left(p_\Theta(\vec{x}^{(n)} \mid \vec{z})\, p_\Theta(\vec{z})\right)\right] \\ &= \sum_n \vec{\nabla}_W \mathbb{E}_{\tilde{q}_\Phi^{(n)}}\left[\log\left(\mathcal{N}(\vec{x}^{(n)}; \vec{\mu}(\vec{z}, W), \sigma^2 \mathbb{I})\right)\right] \\ &\approx -\frac{1}{2\sigma^2} \sum_n \frac{1}{M} \sum_{m=1}^{M} \vec{\nabla}_W \|\vec{x}^{(n)} - \vec{\mu}(\vec{z}^{(m)}, W)\|^2 ,\end{aligned}$$

where $\vec{z}^{(m)} \sim \tilde{q}_\Phi^{(n)}(\vec{z})$. We can slightly rewrite this expression to obtain:

$$\vec{\nabla}_W \mathcal{F}(\Phi,\Theta) \approx -\frac{1}{2\sigma^2} \sum_n \sum_{\vec{z} \sim \tilde{q}_\Phi^{(n)}} \left(\frac{1}{M}\right) \vec{\nabla}_W \|\vec{x}^{(n)} - \vec{\mu}(\vec{z}, W)\|^2 , \tag{4.S3}$$

If we now compare with the gradient using the truncated approximation $q_\Phi^{(n)}(\vec{z})$ given by Eq. (4.9), one can discuss analogous roles played by the sets $\Phi^{(n)}$ (the variational parameters of $q_\Phi^{(n)}(\vec{z})$) and by a standard encoder $\tilde{q}_\Phi^{(n)}$: The states in a set $\Phi^{(n)}$ are used to estimate the gradient similar to the samples from a standard encoder $\tilde{q}_\Phi^{(n)}(\vec{z})$. The size of $\Phi^{(n)}$ can consequently be thought of as analog to the number of samples used in a conventional estimation of the gradient. Standard VAE training estimates the gradient by weighting all samples equally (with $(1/M)$), and the gradient direction is approximated using sufficiently many samples drawn from the current $\tilde{q}_\Phi^{(n)}(\vec{z})$. In contrast, truncated gradient estimation uses the states in $\Phi^{(n)}$, and the gradient is computed using a weighted summation with weights $q_\Phi^{(n)}(\vec{z})$. These weights are computed by passing the states $\vec{z}$ through the *decoder* network. The gradient is then, notably, not a stochastic estimation but exact: gradient ascent is guaranteed (for small steps) to always monotonically increase the variational lower bound.

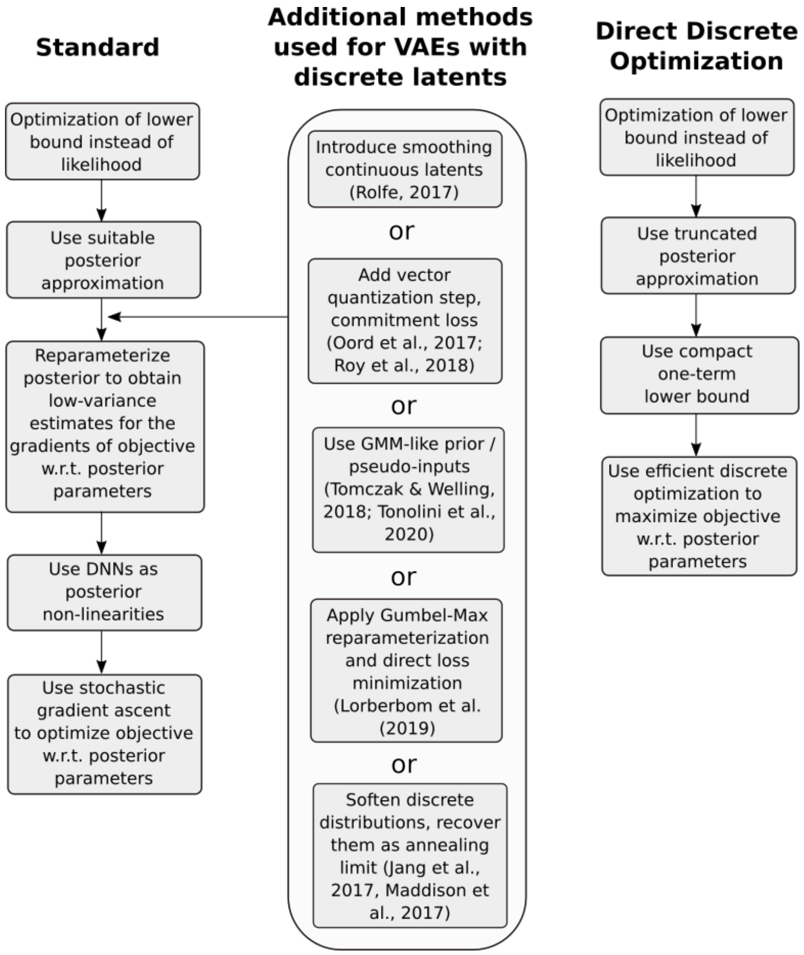

Standard
Optimization of lower bound instead of likelihood
Use suitable posterior approximation
Reparameterize posterior to obtain low-variance estimates for the gradients of objective w.r.t. posterior parameters
Use DNNs as posterior non-linearities
Use stochastic gradient ascent to optimize objective w.r.t. posterior parameters
Additional methods used for VAEs with discrete latents
Introduce smoothing continuous latents (Rolfe, 2017)
or
Add vector quantization step, commitment loss (Oord et al., 2017; Roy et al., 2018)
or
Use GMM-like prior / pseudo-inputs (Tomczak & Welling, 2018; Tonolini et al., 2020)
or
Apply Gumbel-Max reparameterization and direct loss minimization (Lorberbom et al. (2019)
or
Soften discrete distributions, recover them as annealing limit (Jang et al., 2017, Maddison et al., 2017)
Direct Discrete Optimization
Optimization of lower bound instead of likelihood
Use truncated posterior approximation
Use compact one-term lower bound
Use efficient discrete optimization to maximize objective w.r.t. posterior parameters

Figure 4.S2: Typical collection of methods used to optimize the encoding model of VAEs. **Left:** Methods of the standard procedure to optimize VAEs. **Middle:** Examples of additional methods applied to maintain the standard VAE procedure also for VAEs with discrete latent variables. **Right:** Alternative direct discrete optimization of VAE encoding models.

4.S1.1 Computational Complexity

To add to the discussion of computational complexity of TVAE compared to standard VAE training, consider again Eqs. (4.S3) and (4.9). If as many samples M are used, per data point, as there are states in each $\Phi^{(n)}$, then both sums have the same number of summands. The evaluation of the gradients of the mean square error (MSE) is consequently precisely the same for both approaches. The additional weighting factors $q_{\Phi}^{(n)}(\vec{z})$ have to be computed for TVAE. However, the weighting factors just represent a small overhead because the evaluation of the decoder DNN for the states in $\Phi^{(n)}$ is a computation that can be reused from the updates of $\Phi^{(n)}$ (compare Alg. 2).

The main computational differences are in the updates of $\Phi^{(n)}$ compared to the update of encoder DNNs for conventional VAEs. Once the parameters $\Theta = (W, \sigma^2, \vec{\pi})$ are updated using Eqs. (4.9) and (4.10), new states for $\Phi^{(n)}$ have to be sought based on criterion Eq. (4.6).

In practice and for each n, we generate M' new states according to the applied evolutionary procedure. To select the best states we have to pass all these M' new states through the decoder DNN to evaluate Eq. (4.7). Furthermore, we have to pass all M states already in $\Phi^{(n)}$ through the DNN to re-evaluate Eq. (4.7) because the parameters Θ have changed. In summary, we require $\mathcal{O}(N \times (M + M'))$ passes through the decoder DNN. Selecting the M best states from the $(M + M')$ states does not add complexity as this can be done in $\mathcal{O}(M + M')$ for each n (Blum et al., 1973). The EA adds to the computational load but parent selection and mutation only add a constant offset for each of the considered states.

For comparison with standard VAEs, if we use M samples of an encoder $\tilde{q}_{\Phi}^{(n)}(\vec{z})$, we require $\mathcal{O}(M \times N)$ passes through the decoder DNN to update the parameters Θ according to Eq. (4.S3). For the encoder update, one requires $N \times \tilde{M}$ passes through encoder and decoder DNN to estimate the gradient w.r.t. the encoder weights (if we draw $\tilde{M}$ samples for each data point from a conventional encoder distribution $\tilde{q}_{\Phi}^{(n)}(\vec{z})$). The additional overhead to actually draw the samples is usually negligible.

Hence, the computational complexity of TVAE training is comparable if $M \approx M' \approx \tilde{M}$. However, conventional VAE training is amortized, i.e., the update of encoder weights uses information from all data points n. In contrast, TVAE training is not amortized, i.e., the $\Phi^{(n)}$ are updated per data point. The advantage of amortization is that in practice, weights of a conventional encoder can converge faster or (alternatively) less samples $\tilde{M}$ are required. Considering the observed runtimes, more efficient conventional VAE training can, presumably, in large parts be attributed to faster convergence using amortization. Furthermore, the used number of samples M for conventional VAE training is usually smaller than best working sizes of $\Phi^{(n)}$ (we used, e.g., $|\Phi^{(n)}| \in [20, 200]$ for denoising, see Tab. 4.S1); and the required storage of $\Phi^{(n)}$ results in overhead computations. On the other hand, amortization also has disadvantages (e.g., Cremer et al., 2018; Kim et al., 2018). The competitive performance for denoising may consequently be attributed, at least in part, to TVAE not being subject to an amortization gap.

4.S1.2 Evolutionary Variational Optimization

Following Drefs et al. (2022), we base the fitness of a genome $\vec{z}$ for a given data point on the reformulation of the joint $p_{\Theta}(\vec{x}, \vec{z})$ defined by Eq. (4.7) which allows for robust computation, and we use an offset (constant w.r.t. $\vec{z}$) to ensure that fitness values are strictly non-negative. In our large scale numerical experiments, we used fitness-proportional parent selection in combination with uniformly random bitflips (refered to as a *fitparents-randflip* EA, see Drefs et al. (2022) for details), as we found this operator combination to efficiently and effectively suggest new states $\vec{z}^{\,\mathrm{new}}$ for the optimization of the sets $\Phi^{(n)}$ based on criterion Eq. (4.6) (see Fig. 4.S8 for an evaluation of further EA designs that we investigated). EVO hyperparameters include the number of parental states selected per generation (denoted $N_p \leq |\Phi^{(n)}|$), the number of children evolved per parent (N_c), and the number of generations evolved (N_g). When crossover is employed, the total number of new states evolved per data point per epoch is given by $N_p(N_p - 1)N_g$, otherwise it is $N_p N_c N_g$. The concrete hyperparameters used in the numerical experiments are listed in Tab. 4.S1. An illustration of the optimization process of the sets $\Phi^{(n)}$ by EVO is provided in Fig. 4.S3.

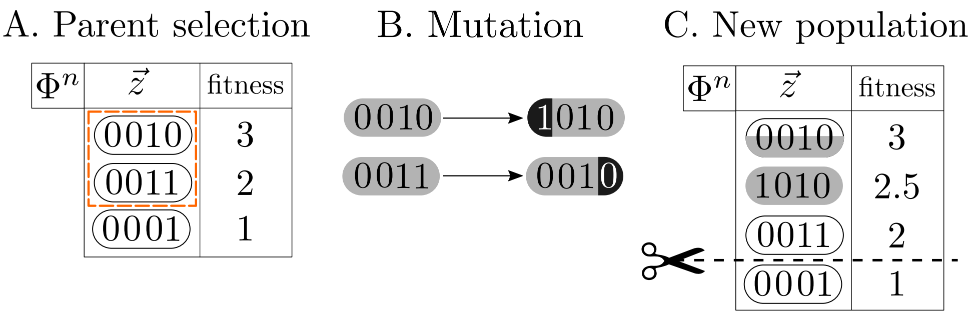


Figure 4.S3: The optimization process of the variational parameters $\Phi^{(n)}$ using evolutionary search. **A.** Some states are selected as parents. **B.** Each child undergoes mutation. **C.** Children are merged with the original population and the least fit are discarded.

4.S1.3 Numerical Evaluation in Related Work

In terms of numerical evaluation, the approaches discussed in Sec. 4.1.1 demonstrate the benefits of the respective new methodology on very different experiments. For instance, van den Oord et al. (2017) focus on image, video and speech generation capabilities of their approach, and report performances, e.g., for phoneme classification. Roy et al. (2018) show improved image generation performance, e.g., on CIFAR and report accuracy scores for machine translation similar to autoregressive baselines. Tomczak and Welling (2018) show competitive likelihood results in the unsupervised permutation invariant setting for many data sets; and Fajtl et al. (2020) evaluate their approach based on accuracy/sensitivity trade-offs among other comparisons. Tonolini et al. (2020) focus on the competitive capabilities of their VAEs to learn meaningful (disentangled) features; and Lorberbom et al. (2019) focus on the benefits of structured priors. Numerical evaluations of their VAEs and different numbers of categoric latents show better performance of their direct loss minimization than Gumbel-softmax VAEs, and the same applies for comparisons in terms of semi-supervised learning. Similarly to Tonolini et al. (2020), disentanglement results for the relatively large CelebA dataset are also shown by Lorberbom et al. (2019). For this and many other datasets, large DNNs are required while the latent layers are often kept relatively small.

4.S2 Details on Numerical Experiments

4.S2.1 Hyperparameters

For all numerical experiments, DNN training using Eq. (4.9) was performed with mini-batches, the Adam optimizer (Kingma and Ba, 2015) and decaying or cyclical learning rate scheduling (Smith, 2017). Xavier/Glorot initialization (Glorot and Bengio, 2010) was used for the DNN weights, while biases were always zero-initialized. Parameters $\vec{\pi}$ and σ^2 were updated via Eq. (4.10). $\vec{\pi}$ was initialized to $\frac{1}{H}$. σ^2 was initialized to 0.01 with the exception of the Barbara, CIFAR-10 datasets and Audio datasets (Figs. 4.2 and 4.S12 and Tab. 4.S4) for which we initialized σ^2 with the data variance. The $\Phi^{(n)}$ were initialized by drawing $\vec{z}$ from a Bernoulli distribution with $p(z_h = 1) = \frac{1}{H}$. Hyperparameter optimization was conducted manually, and for the more complex datasets, it also made use of black box Bayesian optimization based on Gaussian Processes (Nogueira, 2019)

Table 4.S1: Hyperparameters used in the numerical experiments. The architecture of the decoder DNN is denoted H_0-H_1-…-D with H_0 and D indicating the number of Bernoulli latents and Gaussian observables respectively, and H_1-… denoting the number of hidden layer units. By default, we used ReLU activations in the hidden layers and a linear output layer. For Barbara and CIFAR-10, we used LeakyReLU instead of ReLU; for CIFAR-10, we additionally used a Sigmoid in the output layer. Min and max l.r. denote lower and upper learning rate boundaries and are, together with Epochs/Cycle, hyperparameters of the cyclical learning rate scheduler (Smith, 2017). $|\Phi^{(n)}|$ denotes the number of distinct latents per data point (referred to as S in Alg. 2). † and ‡ refer to the parameters used for $\sigma \in \{15, 25\}$ and $\sigma = 50$ in Fig. 4.1 D, respectively.

	Decoding Model					Encoding Model	
	H_0-H_1-…-D	Min l.r.	Max l.r.	Epochs/Cycle	Batch Size	$\lvert\Phi^{(n)}\rvert$	EA
Bars (Fig. 4.S5)	8-8-16	0.0001	0.05	20	32	64	fit-randflip ($N_p = 3$, $N_c = 2$, $N_g = 1$)
Bars (Figs. 4.S6 and 4.S7)	8-8-16	0.0001	0.1	50	32	2^{H_0}	*exact E-step*
Bars (Fig. 4.S8)	6-8-9	0.0001	0.05	20	32	32	*random sampling* ($N_{\text{new}} = 20$, $p(z_h = 1) = \frac{1}{H_0}$)
Bars (Fig. 4.S8)	6-8-9	0.0001	0.05	20	32	32	fit-randflip ($N_p = 5$, $N_c = 4$, $N_g = 1$)
Bars (Fig. 4.S8)	6-8-9	0.0001	0.05	20	32	32	rand-randflip ($N_p = 5$, $N_c = 4$, $N_g = 1$)
Bars (Fig. 4.S8)	6-8-9	0.0001	0.05	20	32	32	fit-cross-randflip ($N_p = 5$, $N_g = 1$)
Bars (Fig. 4.S8)	6-8-9	0.0001	0.05	20	32	32	rand-cross-randflip ($N_p = 5$, $N_g = 1$)
Bars (Fig. 4.S8)	8-8-16	0.0001	0.05	20	32	64	*random sampling* ($N_{\text{new}} = 20$, $p(z_h = 1) = \frac{1}{H_0}$)
Bars (Fig. 4.S8)	8-8-16	0.0001	0.05	20	32	64	fit-randflip ($N_p = 5$, $N_c = 4$, $N_g = 1$)
Bars (Fig. 4.S8)	8-8-16	0.0001	0.05	20	32	64	rand-randflip ($N_p = 5$, $N_c = 4$, $N_g = 1$)
Bars (Fig. 4.S8)	8-8-16	0.0001	0.05	20	32	64	fit-cross-randflip ($N_p = 5$, $N_g = 1$)
Bars (Fig. 4.S8)	8-8-16	0.0001	0.05	20	32	64	rand-cross-randflip ($N_p = 5$, $N_g = 1$)
van Hateren (Fig. 4.S9)	300-300-256	0.0001	0.001	10	32	100	fit-randflip ($N_p = 8$, $N_c = 7$, $N_g = 2$)
House (Fig. 4.1 C,D†)	64-64-64	0.0001	0.01	20	32	200	fit-randflip ($N_p = 10$, $N_c = 9$, $N_g = 4$)
House (Fig. 4.1 D‡)	512-512-144	0.0001	0.05	20	32	64	fit-randflip ($N_p = 5$, $N_c = 4$, $N_g = 1$)
Barbara (Fig. 4.2)	50-500-500-64	0.0001	0.001	160	512	100	fit-randflip ($N_p = 5$, $N_c = 4$, $N_g = 1$)
Barbara (Fig. 4.2)	256-256-512-64	0.0001	0.001	160	512	100	fit-randflip ($N_p = 5$, $N_c = 4$, $N_g = 1$)
House 50 (Fig. 4.3)	512-512-144	0.0001	0.01	20	32	64	fit-randflip ($N_p = 5$, $N_c = 4$, $N_g = 1$)
Castle 50 (Fig. 4.3)	512-512-25	0.0001	0.00125	20	32	32	fit-randflip ($N_p = 5$, $N_c = 4$, $N_g = 1$)
Castle 80 (Fig. 4.3)	512-512-144	0.0001	0.001	20	32	64	fit-randflip ($N_p = 5$, $N_c = 4$, $N_g = 1$)
CIFAR-10 (Tab. 4.S4)	32-512-3072	0.0004	0.009	160	128	20	fit-randflip ($N_p = 15$, $N_c = 10$, $N_g = 1$)
CIFAR-10 (Tab. 4.S4)	50-500-500-3072	0.0004	0.009	160	128	20	fit-randflip ($N_p = 15$, $N_c = 10$, $N_g = 1$)
CIFAR-10 (Tab. 4.S4)	1024-256-512-3072	0.0004	0.009	160	128	20	fit-randflip ($N_p = 15$, $N_c = 10$, $N_g = 1$)
CIFAR-10 (Tab. 4.S4)	1024-512-3072	0.0004	0.009	160	128	20	fit-randflip ($N_p = 15$, $N_c = 10$, $N_g = 1$)
Audio (Fig. 4.S12)	200-512-400	0.0001	0.001	160	512	20	fit-randflip ($N_p = 15$, $N_c = 1$, $N_g = 1$)

and BOHB (Falkner et al., 2018) using the HpBandSter framework (HpBandSterCode developers, 2019). Table 4.S1 provides an overview of the hyperparameters used in each of the reported experiments. A graphical representation of the decoder architecture of TVAE used in the experiments is provided in Figure 4.S4.

4.S2.2 Verification Experiments

We first evaluated TVAE training on artificial datasets with known ground-truth parameters and log-likelihood, in order to verify the correct functioning of the algorithm and to investigate possible local optima effects. The dataset consisted of 500 4x4 images generated by linear superposition of vertical and horizontal bars (compare, e.g., Földiák, 1990; Guiraud et al., 2018; Hoyer, 2003), with a small amount of Gaussian noise. The DNN's input and middle layers had 8 units each. The $\Phi^{(n)}$ variational sets consisted of 64 hidden states each. Figure 4.S5 shows the evolution of the run that achieved the highest ELBO value out of ten. All parameters were correctly recovered, and the ELBO value was consistent with actual ground-truth log-likelihood. Such an elementary test, however, can also be solved by linear models. In order to demonstrate that TVAEs can solve non-linear problems,

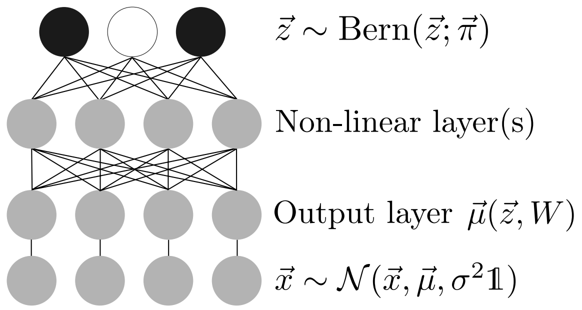

$\vec{z} \sim \mathrm{Bern}(\vec{z}; \vec{\pi})$
Non-linear layer(s)
Output layer $\vec{\mu}(\vec{z}, W)$
$\vec{x} \sim \mathcal{N}(\vec{x}, \vec{\mu}, \sigma^2 \mathbb{1})$

Figure 4.S4: Graphical representation of the model architecture used in numerical experiments.

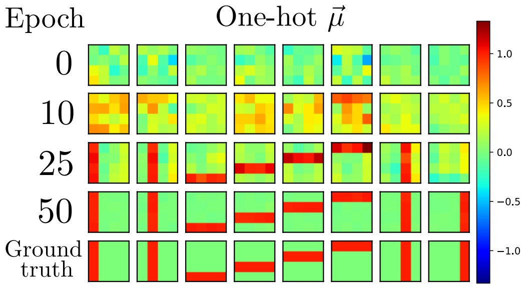


Figure 4.S5: TVAE Training on Simple Bars Data: Noiseless output of the TVAE's DNN for the eight possible one-hot input vectors over several training epochs. Generating parameters are in the last row.

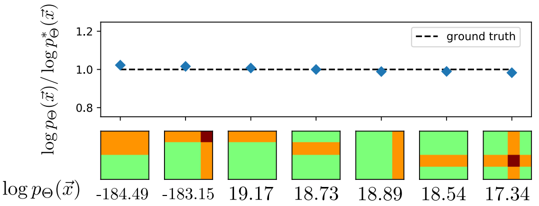


Figure 4.S6: Correlated Bars Test. The plot shows the ratio between inferred and ground-truth log-likelihoods $\log p_\Theta(\vec{x})$ of data points with interesting bar combinations. The inferred values are reported below the data points themselves.

taking advantage of the neural network non-linearity embedded in the generative model, we introduced correlations between pairs of bars: the bar combinations shown in the first two data points from the left in Fig. 4.S6 were discouraged from appearing together. Again, we selected the run with highest peak ELBO value out of ten. The model correctly learned that certain combinations of bars are much more unlikely than others, and correctly estimated their likelihood.

Figure 4.S7 offers more insight into the correlated bars test experiment. The left section of the figure depicts the parameters used to generate the dataset: W_0 is the 8x8 weight matrix of the top-to-middle layer, which, in this case, caused activation of the first latent variable to inhibit activation of the second, and activation of the last latent variable to inhibit activation of the first. Concretely, this results in a dataset where these specific bars combinations are discouraged from appearing. The weights W_1, visualized as 8 4x4 matrices, generate the actual bars. σ^2 was set to 0.01 and the dataset contained an average of two superimposing bars per data point ($\pi_h = 2/8$ for each h). The middle section of the figure shows the ELBO values (averages over all batches for each epoch) as training progresses. The cyclic learning rate schedule is responsible for the oscillatory behavior. The right section shows some example data points together with samples from the trained TVAE model that reached the highest ELBO value out of the ten runs.

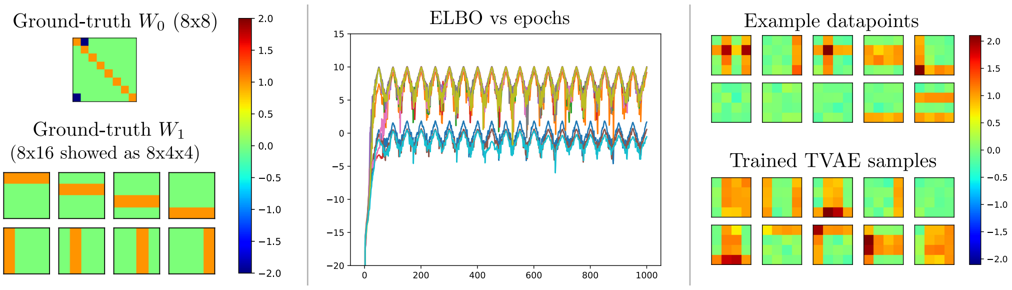


Figure 4.S7: Generative Parameters for the Correlated Bars Test (left); ELBO Values over Epochs for 10 runs (center); Example Datapoints and Samples from the Generative Model (right).

Using bars test data generated with a linear model (no correlations between pairs of bars) as described above, Fig. 4.S8 depicts reliabilities of different evolutionary operator designs (also compare Sec. 4.S1.2).

4.S2.3 Scalability Experiments

We extracted $N = 100,000$ image patches of size $D = 16 \times 16$ from a standard image database (van Hateren and van der Schaaf, 1998) and applied pre-processing as in Guiraud et al. (2018). The most elementary VAEs use a linear mapping for the decoder $\vec{\mu}(\vec{z}, W)$. While being a natural baseline, such VAEs have also played a role in understanding important properties of VAE learning (e.g., Dai et al., 2018; Lucas et al., 2019). For standard Gaussian latents, a linear VAE can be shown to recover probabilistic PCA solutions. For Bernoulli latents, we can recover binary sparse coding (Haft et al., 2004; Shelton et al., 2011) in case of linear decoders. We therefore began our analysis with a shallow linear VAE using $H = 300$ latents. After 100 epochs, the weights of the linear mapping were used to initialize the bottom layer of a deeper, non-linear decoder network with three layers of 300, 300 and $16 \times 16 = 256$ units, respectively (using ReLU activations, cf. Sec. 4.S2.1). The weights of the deeper layers were simply initialized to the identity matrix. Besides DNN weights, prior and variance parameters were continuously optimized. This setup guaranteed a common starting point for linear and non-linear VAEs such that the difference provided by deeper decoder DNNs can be highlighted. Figure 4.S9 C shows the variational bounds during learning of the linear VAE compared to the non-linear one for a typical run. The non-linear VAE can be observed to quickly and significantly optimize the lower bound beyond a linear VAE. In proceeding experiments (where we were not concerned with comparisons to linear VAEs), we simply optimized the weights of the non-linear TVAE directly as we did not observe an advantage in first optimizing a linear model.

Regarding scalability, we observed a similar efficiency of non-linear TVAE compared to linear models (compare main text). To further investigate scalability, we trained a TVAE model with up to H=1000 latent variables (while using 100 units in the DNN middle layer). Training time remained in line with the theoretical linear scaling with H while the variational bound further increased.

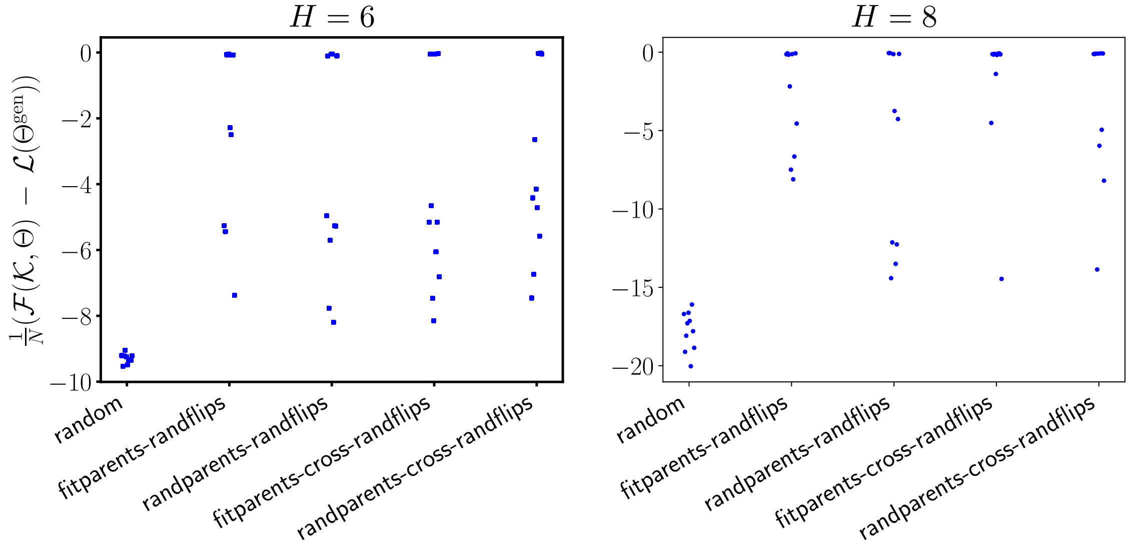


Figure 4.S8: Reliability of different evolutionary operator combinations on bars test (cf. Secs. 4.S1.2 and 4.S2.2). Reliability is quantified in terms of the normalized difference between the lower bound at the last training epoch and the log-likelihood computed using the ground-truth generating parameters. For the strategy labeled as 'random', we randomly sampled states from a Bernoulli with $p(z_h = 1) = \frac{1}{H}$.

4.S2.4 Image Denoising

'Zero-Shot' Setting. If no clean data is available, variations of feed-forward DNNs have been suggested whose training objectives have been altered to enable training on single (or few) noisy images (e.g., Krull et al., 2019a; Lehtinen et al., 2018). However, deep *generative* models are (we would argue) more natural candidates to train on noisy data as their learning objective can be used directly. If few data is available for 'zero-shot' learning, large and very deep DNNs cannot be used; Shocher et al. (2018), for instance, argued that smaller DNNs are sufficient for the 'zero-shot' setting. 'Zero-shot' denoising is consequently natural and well suited to evaluate direct optimization, and it has the very significant additional benefit of allowing for a comparison to a large range of other approaches that have recently been suggested. Most notably, the benchmark allows for comparison to other VAEs (Jang et al., 2017; Park et al., 2019a), to sparse coding, to large feed-forward DNNs (Dong et al., 2019; Zhu et al., 2019), and to DNNs dedicated to learning from noisy data (Krull et al., 2019a; Lehtinen et al., 2018) (Figs. 4.1 and 4.2). The possibility of broad comparability is also provided for the 'zero-shot' benchmarks of Fig. 4.3 (with inpainting highlighting another advantage of the direct optimization, namely that, as no encoder DNNs are used, missing data can naturally and directly be treated probabilistically).

Data Estimator. We followed Drefs et al. (2022), and, given a trained TVAE with parameters Θ, estimated the value of a pixel in a single patch as:

$$x_d^{\mathrm{est}} = \mathbb{E}_{p_\Theta(x_d \mid \vec{x})}\big[x_d\big], \tag{4.S4}$$

which, by using $p_\Theta(x_d \mid \vec{x}) = \sum_{\{\vec{z}\}} p_\Theta(x_d \mid \vec{z}) p_\Theta(\vec{z} \mid \vec{x})$, can be reformulated as:

$$x_d^{\mathrm{est}} = \mathbb{E}_{p_\Theta(\vec{z}\mid\vec{x})}\left[\mathbb{E}_{p_\Theta(x_d\mid\vec{z})}\big[x_d\big]\right] = \mathbb{E}_{p_\Theta(\vec{z}\mid\vec{x})}\big[\mu_d(\vec{z})\big]\,. \tag{4.S5}$$

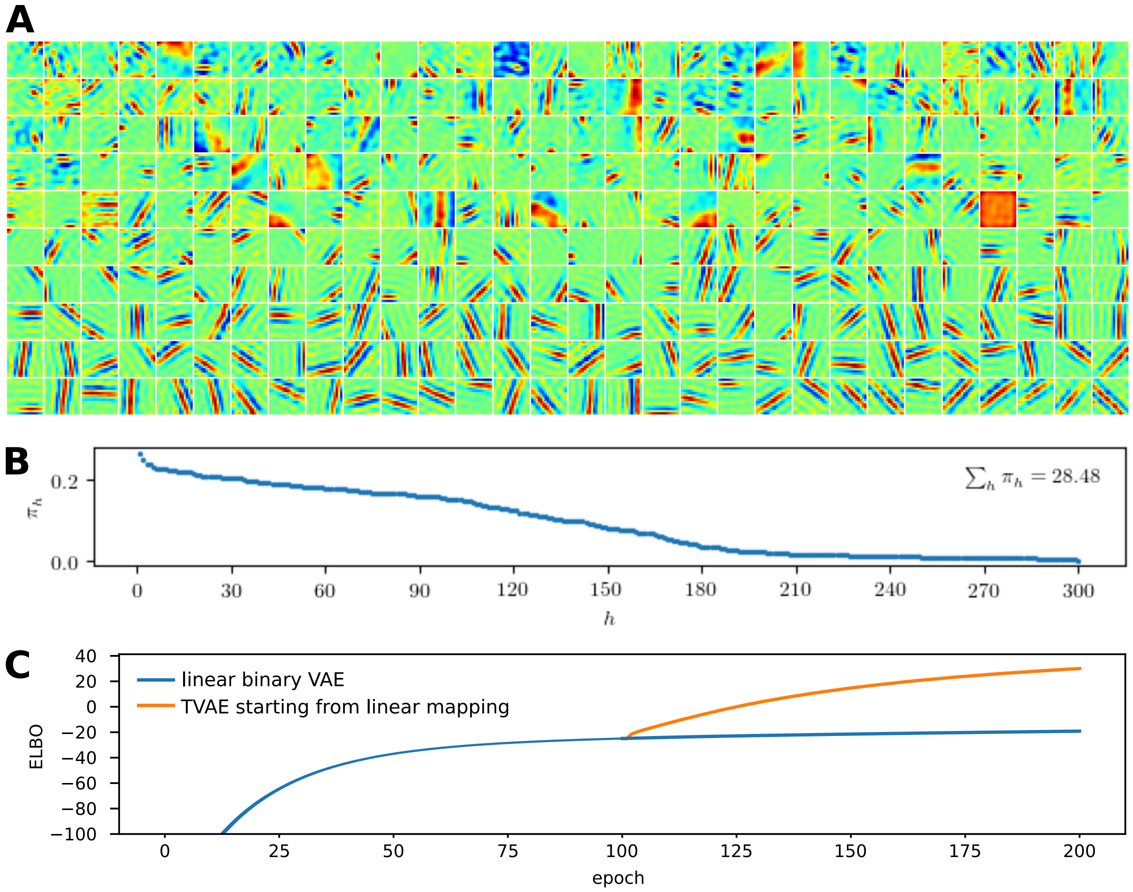


Figure 4.S9: Training TVAE on natural image patches. **A** Decoder outputs for singleton inputs after training, sorted according to priors as depicted in **B**. GFs are locally scaled. Fields with high prior probabilities often have lower amplitudes than other GFs. Especially amplitudes of the first fields (especially top rows) are relatively close to zero. **C** ELBO gain compared to linear VAE with binary latents (i.e., EBSC; see text for details).

The expectation value on the right-hand-side of Eq. (4.S5) is approximated based on the encoding parameters $\Phi^{(n)}$ using truncated posteriors (Eq. (4.4)), i.e.:

$$\mathbb{E}_{p_\Theta(\vec{z}\,|\,\vec{x}^{\,(n)})}\big[g(\vec{z})\big] \approx \mathbb{E}_{q_\Phi^{(n)}}\big[g(\vec{z})\big] = \frac{\sum\limits_{\vec{z}\in\Phi^{(n)}} g(\vec{z})\,p_\Theta(\vec{x}^{\,(n)},\vec{z})}{\sum\limits_{\vec{z}\,'\in\Phi^{(n)}} p_\Theta(\vec{x}^{\,(n)},\vec{z}\,')} \tag{4.S6}$$

Finally, the pixel value of in the denoised image is generated by averaging the pixel estimates obtained from different patches (see Drefs et al., 2022, for details).

Comparison to Probabilistic Sparse Coding. To evaluate the performance on standard denoising benchmarks, we started by comparing TVAE to related probabilistic sparse coding approaches (Fig. 4.1 C). MTMKL and GSC both use spike-and-slab sparse coding (SSSC) data models while for training, the approaches use mean field and truncated posterior approximations with pre-selection, respectively. Compared to MTMKL and GSC, EBSC uses a less complex data model and EVO-based training (Drefs et al., 2022). We computed the PSNRs for EBSC in Fig. 4.1 C by running publicly available source code (EVO developers, 2022) using the 'fitparents-randflip' EA with hyperparameters $S = 200$, $N_p = 10$, $N_c = 9$ and $N_g = 4$. The performances listed in Fig. 4.1 C reveal that, for high noise level ($\sigma = 50$), EBSC achieves higher PSNR values than MTMKL and GSC although the method uses a simpler data model, which demonstrates EVO's effectiveness. However, PSNR values for TVAE are significantly higher due to the higher flexibility in modeling the data distribution provided by the used DNN.

Comparison to Noise2Noise-related Approaches. Figure 4.1 D distinguishes algorithms by the amount of employed training data and by the requirement for clean data. Approaches such as N2N and N2V occupy a middle ground: While they *can* be trained on noisy data, they typically require much larger amounts of data than, e.g., TVAE or MTMKL, to achieve best performance. In the original N2V publication (Krull et al., 2019a), the generation of training datasets leveraged, for instance, 400 (noisy) 180 × 180 BSD (Martin et al., 2001) images and data augmentation procedures. For our comparison in Fig. 4.1 D, we used the standard, publicly available code for N2V (Krull et al., 2019b) together with the default training set ($\sigma = 25$) employed in the original N2V publication (specifically, we followed the 'denoising2D_BSD68' example using its default hyperparameters from the corresponding GitHub repository). We then applied the trained N2V network to denoise the House image with $\sigma = 25$. The resulting PSNR value was 32.10 dB which is 0.76 dB lower compared to BM3D (32.86 dB). The difference is consistent with an on average 0.88 dB lower performance of N2V compared to BM3D on BSD68 images reported in Krull et al. (2019a). The same network can also be used to denoise an image with lower or higher noise level. The N2V network trained on $\sigma = 25$, for instance, results in PSNR values of 32.93 dB for the House image with $\sigma = 15$ and in 20.96 dB for House with $\sigma = 50$ (N2V† in Tab. 4.S2).

Especially for high noise levels, performance can be much improved, however, if N2V is trained using images with matched noise level (i.e., same as in the test image). To consider such setting, we followed the original publication and trained N2V models while adapting the noise level of $\sigma = 15$ in one case and $\sigma = 50$ in the other case (as above, we used the 'denoising2D_BSD68' example as reference implementation). We then applied the trained models to denoise the House image with $\sigma = 15$ in the one, and $\sigma = 50$ in the other case (results listed as N2V‡ in Tab. 4.S2). The PSNR values we obtained are much higher compared to the scenario with unmatched noise level (for $\sigma = 50$, for instance, we observed a PSNR improvement of approximately 8 dB). The much lower performance of N2V for unmatched noise level is, in this respect, consistent with observations for standard DNN denoising for which training with matched noise level has been pointed out crucial to achieve strong performance (e.g., Chaudhury and Roy, 2017; Zhang et al., 2018).

The N2V approach can avoid having to know the exact noise level, e.g., if it is trained on just the single noisy image. In a last experiment, we hence investigated N2V in such 'zero-shot' setting and applied the algorithm to denoise the House image while using the same noisy image for training that we seeked to denoise (our implementation was based on the 'denoising2D_BSD68' example as above). The obtained PSNR values are listed as N2V* in Tab. 4.S2 (they correspond to the best out of three runs for which Fig. 4.1 D reports the respective averages). From Tab. 4.S2 it can be observed that for all considered training settings of N2V and all noise levels, PSNR values of TVAE are consistently higher than

Table 4.S2: Denoising performance of N2V in PSNR (dB) for House. For comparison, we additionally list the performance of TVAE (numbers copied from Fig. 4.1 D). PSNR values for N2V* are obtained by training only on the noisy image (i.e., in the same setting as used for TVAE). More training data improves performance for N2V. PSNR values for N2V† show performance if additional training data in the form of noisy images with AWG noise $\sigma = 25$ is used. Further improvements (especially for high noise) are obtained if the N2V network is trained on training data with a noise level that matches the noise of the test set (see N2V‡). For instance, we used for N2V‡ training data with $\sigma = 50$ to denoise the House image with $\sigma = 50$. PSNR values were computed using the model in its state at the training epoch with smallest validation loss; for N2V*, we performed three independent runs of the algorithm and here report the results of the best run (in terms of validation loss). See text for further details.

	σ=15	σ=25	σ=50
N2V*	32.22	29.69	24.89
N2V†	32.93	32.10	20.96
N2V‡	33.91	32.10	28.94
TVAE	**34.27 ± .02**	**32.65 ± .06**	**29.98 ± .05**

those of N2V even if N2V is trained on external data with matched-noise level. Also note that strong performance of TVAE is notably achieved using basic DNNs and comparably small patch sizes ($D = 8 \times 8$ for $\sigma = 15$ and $\sigma = 25$, and $D = 12 \times 12$ for $\sigma = 50$, cf. Tab. 4.S1); all feed-forward DNNs for denoising, for instance, use much larger patches (e.g., N2V uses 64×64). Additional parameter tuning may improve performance of N2V* to a certain extent but PSNRs are in general much lower than N2V‡. While we followed for N2V‡ the standard hyperparameter setting of the original paper/code publication of N2V (Krull et al., 2019a,b), we cannot exclude further improvements with parameter fine tuning for the House benchmark. However, we remark that the difference of N2V‡ and BM3D for the House benchmark is on the very same range as the differences between N2V and BM3D as reported for BSD images in the original N2V publication. The stronger performing BM3D is, according to denoising performance, the preferable comparison and as such included in Fig. 4.1 D. In terms of efficiency, the N2V approach is in general (once trained) faster than BM3D as well as TVAE, however (compare Krull et al., 2019a).

PSNR values of N2N are usually very closely aligned with performances achievable by feed-forward DNNs. More concretely, N2N uses, for instance, a RED30 network (Mao et al., 2016) which achieves 31.07 dB PSNR on the BSD300 data set if trained on clean data. If directly trained on noisy data, RED30 achieves 31.06 dB (Lehtinen et al., 2018). N2N is thus strongly performing in terms of PSNR. Compared to N2V, however, the caveat of N2N is that its training uses rather articifial data. The pairs of images which N2N is trained on consist of two different noisy realizations of the same underlying clean image. Building such datasets may, under realistic conditions, be very difficult, which motivated follow-up approaches such as N2V or S2S. For S2S, which is a dedicated 'zero-shot' denoising approach, we observed generally very strong performances for the considered 'zero-shot' benchmarks (the PSNRs in Fig. 4.1 D were produced by running publicly available source code (Quan et al., 2020b), specifically using the 'demo_denoising' example of the respective GitHub repository).

Comparison to Deep Generative Models. While deep generative models are in general well suited for denoising applications, performance on common image denoising benchmarks is not as commonly investigated as for standard DNNs (which may also be related to efficiency aspects). Exceptions include the recent BDGAN (Zhu et al., 2019) and the very recent DivNoising (Prakash et al., 2021b) and NN+X (Zheng et al., 2021) approaches. Regarding VAEs, we here compared TVAE to GSVAE (Jang et al., 2017) and VLAE (Park et al., 2019a) using publicly available source code together with default hyperparameters (Jang, 2016; Park et al., 2019b). We observed the implementations of both GSVAE and VLAE to yield significantly better results when rescaling pixel amplitudes to the interval [0, 1]. To maintain comparability, we applied the same procedure for TVAE, EBSC and ES3C when we produced the results of Fig. 4.2 and Tab. 4.S4. To apply GSVAE and VLAE for image denoising, we used the same patch-based processing as employed for TVAE. The results for VLAE in Fig. 4.1 D were produced using a patch size of 12×12. For GSVAE, we used Gaussian rather than Bernoulli observables (as implemented in the publicly available source code). Following Drefs et al. (2022), we applied GSVAE using both binary and categorical latents (referred to as GSVAE-B and GSVAE-C in the main text). The PSNRs for GSVAE-B in Fig. 4.1 D are cited from Drefs et al. (2022) (where the respective model is referred to as GSVAE$^{\text{lin}}$, see Tab. 3 B in that paper). For GSVAE-C in Fig. 4.1 D, we used a (20x10)-256-512-(12x12) decoder architecture (i.e., 20 latents, each with 10 categories, two hidden layers with 256 and 512 units, and $12 \times 12 = 144$ output units). In Fig. 4.2, all models used the same number of decoder output units ($D = 8 \times 8 = 64$); there, we investigated GSVAE-B with shallow and deep decoders (GSVAE-B (256) and GSVAE-B (256-256-512) both used 256 binary latents, and the deep decoder consisted of two hidden layers with 256 and 512 units) as well as GSVAE-C with a deep decoder (32 latents, each with 8 categories and two hidden layers with 256 and 512 units).

Computational Demand. An important limitation of TVAE is its computational demand. For our experiments on the House image with noise level $\sigma = 50$ in Fig. 4.1 D, we used $N = 60025$ patches of $D = 12 \times 12$ pixels, which amounts to all possible overlapping square patches of that size that can be extracted from the image. For training and denoising we used a TVAE with $H = 512$ latent variables, sizes of $|\Phi^{(n)}| = 64$, and 512 units in the DNN middle layer of the decoder (cf. Tab. 4.S1). TVAE training required 49 seconds per training epoch when executing on a single NVIDIA Titan Xp GPU and 2.5 GB of GPU memory. We ran for 500 epochs which required between seven and eight hours on the single GPU. We did not observe significant changes in variational bound values or in denoising performance after 500 epochs in any of the experiments we conducted for Fig. 4.1. Runtime complexity increased linear with the number of data points N, with the dimensionality of the data D, with the number of the latents H, and with the size of the DNN used. Runtimes also increased approximately proportional w.r.t. the size of $\Phi^{(n)}$. Empirically, we observed a sub-linear scaling with $|\Phi^{(n)}|$, presumably because of significant overhead computations. For example, increasing from $|\Phi^{(n)}| = 64$ to $|\Phi^{(n)}| = 128$ (while keeping all other parameters as above) computational time increases from 49 seconds per training epoch to 75 seconds. For noise levels $\sigma = 15$ and $\sigma = 25$ in Fig. 4.1 D, we used smaller patch sizes ($D = 8 \times 8$) and fewer stochastic latents ($H = 64$) but larger $\Phi^{(n)}$ (i.e., $|\Phi^{(n)}| = 200$).

Table 4.S3: Runtimes required by GSVAE, VLAE and TVAE for denoising the Barbara image with $\sigma = 25$ (compare Fig. 4.2). † Total runtime of VLAE consists of 5.1 hrs for parameter optimization and 1.8hrs for importance sampling. The PyTorch implementations of the models were executed on a single NVIDIA Tesla V100 NV-Link 32GB HBM2 on a server with Intel Xeon 4214 12-core 2.20 GHz CPUs.

GSVAE-B				VLAE		TVAE			
50-500-500		256-256-512		50-500-500		50-500-500		256-256-512	
epoch	total	epoch	total	epoch	total	epoch	total	epoch	total
10s	0.01hrs	11s	0.03hrs	46s	$6.9^{\dagger}$hrs	213s	9.5hrs	234s	10.4hrs

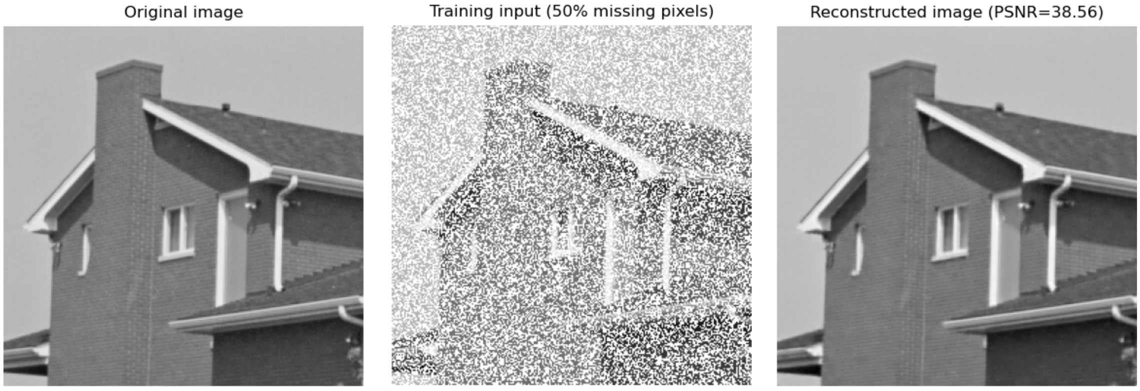

Original image
Training input (50% missing pixels)
Reconstructed image (PSNR=38.56)

Figure 4.S10: TVAE inpainting results for House.

In general, if the patch size D is increased, more structure has to be captured. This can be done either by increasing the size of the stochastic latents H or by using larger DNNs. Both, in turn, requires more training data in order to estimate the increased number of parameters. In the current setup, the sizes of D which are currently feasible are comparably small. The denoising performance based on small patches is, however, notably very high.

For comparison, N2V uses up to $D = 64 \times 64$, and also other feed-forward DNN approaches use significantly larger patch sizes than TVAE (and other ‘zero-shot’ approaches). Still, N2V can be trained efficiently on large patches requiring approximately 19 hours on a NVIDIA Tesla K80 GPU for training on approximately 3K noisy images of shape 180×180 and seconds for the denoising of one 256×256 image. The higher computational demand of TVAE is also the reason why averaging across databases with many images (such as BSD68) or applications to large single images quickly becomes infeasible. As a novel approach, TVAE is, however, far from being fully optimized algorithmically compared to large feed-forward approaches, and there is certainly further potential to improve training efficiency. Tab. 4.S3 systematically compares runtimes of GSVAE, VLAE and TVAE for denoising the Barbara image (cf. Fig. 4.2).

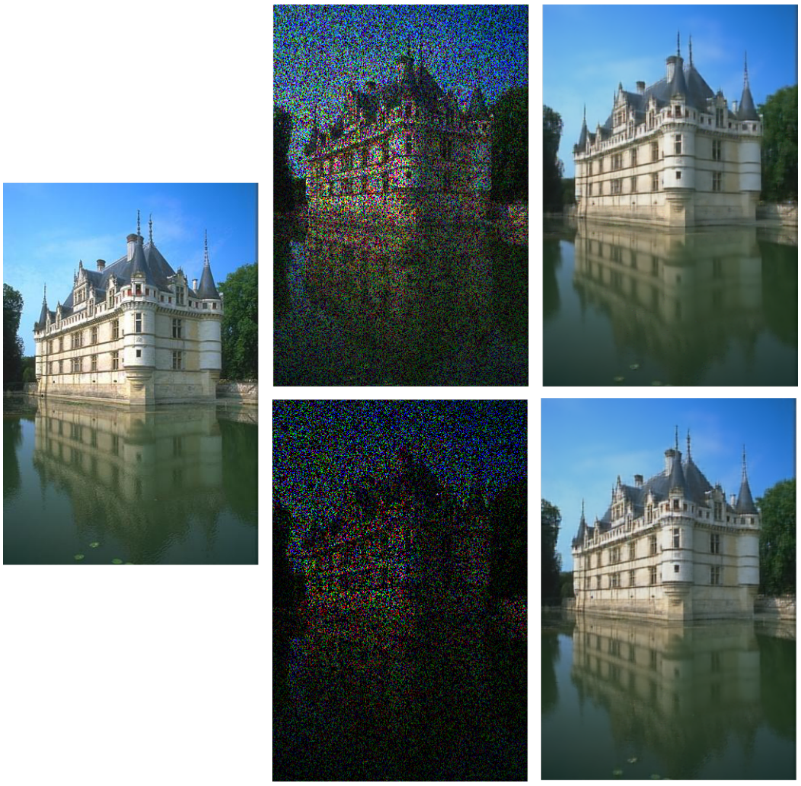


Figure 4.S11: Inpainting of the Castle Image with TVAE. **Left** Original image. **Center** Training image (top, 50% missing pixels, bottom, 80% missing pixels). **Right** Inpainting result with TVAE (top, 50% missing pixels, bottom, 80% missing pixels).

4.S2.5 Image Inpainting

Similarly to the use of TVAE for denoising (see Sec. 4.S2.4), a single image with missing pixels (Fig. 4.S10, center) is divided into square patches to form the training set; during training, missing pixels can be treated as observables with unknown values when evaluating log-joint probabilities of a data-point, which provides a grounded way to treat missing data. To estimate likely values of missing pixels, we use the estimator of Eq. (4.S5) that we applied for denoising before and now only consider the observed entries of a data point when evaluating posterior probabilities:

$$x_d^{\mathrm{est}} = \mathbb{E}_{p_\Theta(\vec{z}|\vec{x}^{\,\mathrm{obs}})}\big[\mu_d(\vec{z})\big]\,. \tag{4.S7}$$

As for denoising, the expectation value in Eq. (4.S5) can be approximated using truncated posteriors with encoding parameters $\Phi^{(n)}$. Complete data points with 'inpainted' pixels can then be used for DNN backpropagation to optimize the decoder. The inpainting procedure consequently directly derives from the standard probabilistic treatment of missing values. In contrast, amortized approaches need to specify how an encoder network should treat missing values because encoder DNNs expect real values for all pixels as input.

When evaluating TVAE on standard inpainting benchmarks, we observed competitive performance compared to other approaches (Fig. 4.3). TVAE is permutation invariant, i.e., the model does not leverage information about the 2D nature of images. In contrast, DIP

Model	Lower Bound	Sparsity
GSVAE-B (1024-256-512)	2380	$\frac{512}{1024}$
GSVAE-C ((1024×2)-256-512)	2559	n/a
EBSC (1024)	2028	$\frac{19}{1024}$
TVAE (32-512)	1649	$\frac{13}{32}$
TVAE (50-500-500)	1831	$\frac{18}{50}$
TVAE (1024-256-512)	2044	$\frac{179}{1024}$
TVAE (1024-512)	2893	$\frac{135}{1024}$
VLAE (32-512)	2392*	n/a
VLAE (50-500-500)	2687*	n/a

Table 4.S4: Log-likelihood estimates and sparsity for CIFAR-10. For comparison, we considered GSVAE (numbers cited from Drefs et al. (2022)) and VLAE (*we cite the importance sampling-based log-likelihood estimates reported in Park et al. (2019a)). Sparsity is measured in terms of the average number of active latents per data point ($\frac{\sum_h \pi_h}{H}$, compare main text). All models used the same number of decoder output units corresponding to the image size of $D = 32 \times 32 \times 3$ pixels. We investigated different decoder architectures (1024-256-512, for instance, denotes $H = 1024$ latents in the decoder input layer and two hidden layers with 256 and 512 hidden units, respectively).

results rely on a large dedicated DNN with LeakyReLU as activation functions, a U-Net / hourglass architecture with skip connections, and convolutional units with reflection padding (see supplement of Ulyanov et al. (2018)). The convolutional stages explicitly assume the 2D image structure. We also remark that DIP uses in total 2 million parameters compared to about 0.5 million parameters of the standard multi-layer perceptron used in TVAE. Furthermore, and as a consequence of its more intricate architecture, DIP uses many more tunable hyper-parameters. In the case of the House image, TVAE training lasted 500 epochs taking around 60 seconds/epoch on a single NVIDIA Titan Xp GPU (see Tab. 4.S1 for hyper-parameters used in the experiment).

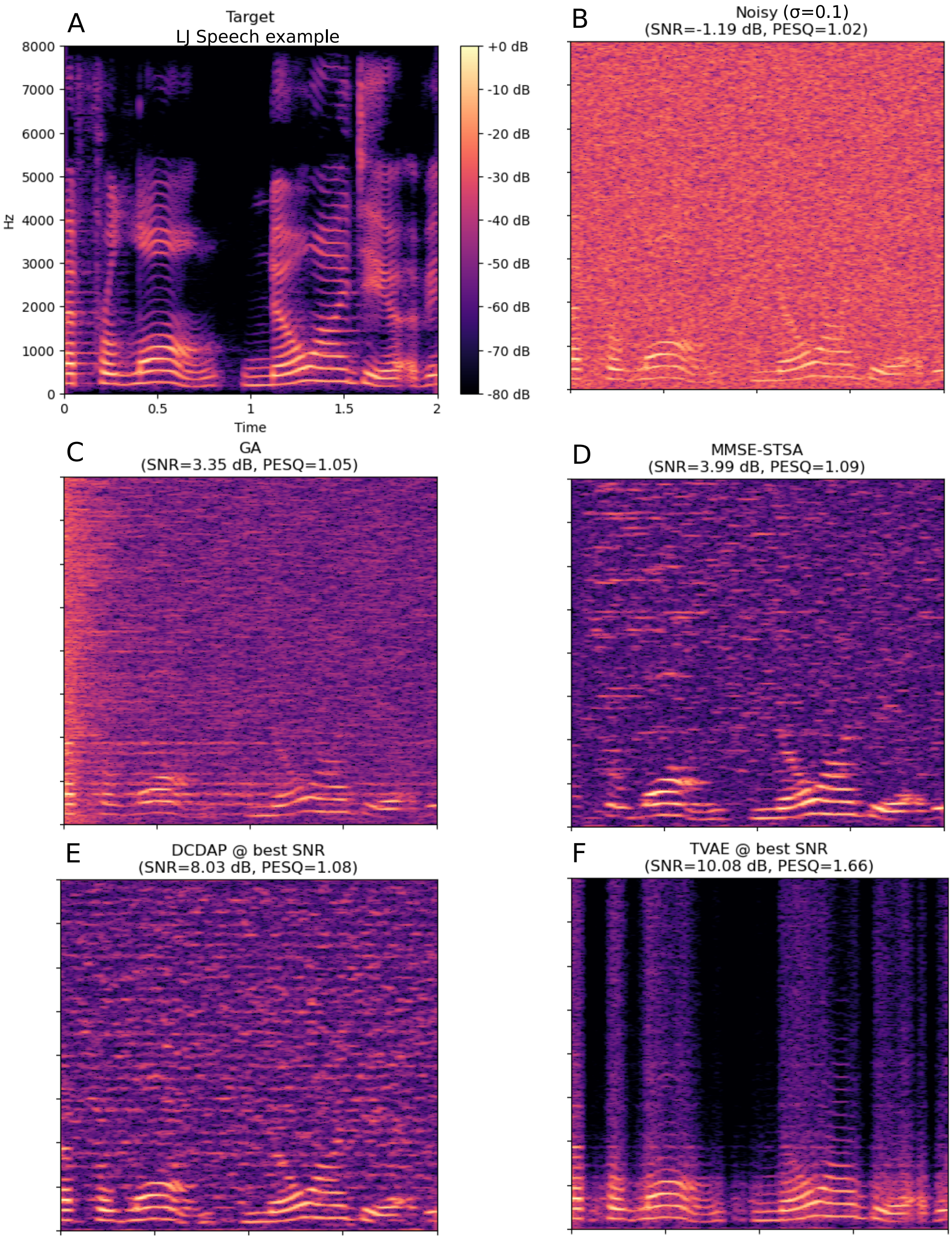
4.S2.6 Single-Object Image Data

Considering further commonly used image data, we applied TVAE to CIFAR-10 images. In contrast to the patches extracted from the Barbara image (Fig. 4.2), which captured a small part of a whole image and as such only little elementary image structures, CIFAR-10 images depict entire objects (cars, animals, etc.), and consequently they contain significantly more complex structure. After applying TVAE to the CIFAR-10 training data set, we observed a sparse encoding (similarly to the case of Barbara image patches). Considering lower bounds on the CIFAR-10 test set, the encoding learned by TVAE showed to result in a less competitive performance in comparison to GSVAE and VLAE when controlling for comparable decoder network architectures (Tab. 4.S4)

4.S2.7 Zero-Shot Audio Denoising

Learning-based 'zero-shot' data enhancement is subject of current research not only in the context of image but also audio processing (e.g., Michelashvili and Wolf, 2020; Narayanaswamy et al., 2021b; Zhang et al., 2020). Narayanaswamy et al. (2021b), for instance, discuss applications of U-Net-based networks with dilated convolutions to the task of unsupervised Gaussian audio denoising. To address the question of the extent to which the strong 'zero-shot' denoising performance of TVAE on visual data (Figs. 4.1 and 4.2) carries over to audio data, we applied TVAE to the benchmark considered in Sec. 4.1 and Tab. 1 in Narayanaswamy et al. (2021b). Concretely, we followed the experimental design described in Narayanaswamy et al. (2021b) and used the following datasets: LJ-Speech (Ito and Johnson, 2021), SC09 Spoken Numbers (Donahue et al., 2019; Warden, 2018; we used the version made available by Bhalley, 2018), and Bach Performances (Donahue et al., 2019). From each of these three datasets, we randomly selected five examples after resampling the audio files to 16 kHz. Regarding the number of examples per dataset, our procedure slightly diverts from the one used by Narayanaswamy et al. (2021b) who collected 50 examples per dataset. The reason for limiting our evaluation to only five samples lies in avoiding long runtimes of TVAE (compare Secs. 4.S1 and 4.S2.4). For the LJ-Speech and Bach Performances datasets, we randomly selected excerpts of two seconds duration; for SC09, each example had a duration of one second. To each of the in total 15 collected snippets, we added Gaussian noise using $\sigma = 0.1$. We then normalized the noisy waveforms to fill the range [-1, 1]. Narayanaswamy et al. (2021b) do not report to have used such normalization; for TVAE, we observed data normalization to be beneficial as it helped to avoid numerical instabilities related to very small variances learned due to very small waveform amplitudes. Finally, we applied TVAE to the noisy waveform after extracting overlapping chunks of $D = 400$ samples (similarly to the overlapping patches extracted from noisy images, cf. Sec. 4.S2.4).

For the evaluation, we used two standard baselines, namely a Geometric Approach (GA; Lu and Loizou, 2008) and the Minimum-Mean Square Error Short-Time Spectral Amplitude estimator (MMSE-STSA; Ephraim and Malah, 1984) together with the noise power spectral density estimator of Gerkmann and Hendriks (2012). Furthermore, we compared to the approach of Narayanaswamy et al. (2021b) which we refer to as DCDAP (as an abbreviation for 'Deep Audio Prior with dilated convolutions'). We applied the publicly available source code of DCDAP (Narayanaswamy et al., 2021a) together with the default parameter settings; as 'dilation type', we chose the option 'exponential'. For the LJ-Speech and the SC09 datasets, we quantified performance in terms of Signal-to-Noise Ratio (SNR) and Perceptual Evaluation of Speech Quality (PESQ); for the non-speech dataset (Bach Performances), we measured only SNRs. For all three benchmark datasets considered, TVAE can improve the performance of all compared methods in terms of both SNR and PESQ improvement (Fig. 4.S12).



G

	LJ Speech		SC09 Numbers		Bach Performances	∅	
	ΔSNR	ΔPESQ	ΔSNR	ΔPESQ	ΔSNR	ΔSNR	ΔPESQ
GA	6.17	0.02	8.14	0.05	10.50	8.27	0.04
MMSE-STSA	9.04	0.07	8.84	0.18	13.30	10.40	0.13
DCDAP	8.51	0.20	11.02	0.06	10.40	9.98	0.13
TVAE	12.05	0.48	15.11	0.34	15.70	14.29	0.41

Figure 4.S12: Results for 'Zero-Shot' Audio Denoising. Panel A depicts the dB-scaled amplitude spectrogram of an example from the LJ-Speech dataset. Panel B was obtained after adding Gaussian noise to the waveform of the clean speech ($\sigma = 0.1$). Panels C-F show the denoising results of GA (Lu and Loizou, 2008), MMSE-STSA (Ephraim and Malah, 1984), DCDAP (Narayanaswamy et al., 2021b) and TVAE. Panel G reports SNR and PESQ improvements averaged over five examples per dataset (the delta denotes the SNR and PESQ improvement of the denoised w.r.t. the noisy waveform; see text for details).

Chapter 5

Double-Dictionary Maximal Causes Analysis

Parts of this chapter have appeared as: Hamid Mousavi*, Jakob Drefs*, Jörg Lücke. A Double-Dictionary Approach Learns Component Means and Variances for V1 Encoding. In Giuseppe Nicosia et al. (Eds.): *Machine Learning, Optimization, and Data Science.* LOD 2020. Lecture Notes in Computer Science, vol. 12566, pp. 240–244, 2020. Springer, Cham. `https://doi.org/10.1007/978-3-030-64580-9_20`. *Joint first authorship. Reproduced with permission from Springer Nature. The Springer Nature content included in this chapter is not covered by the open access license under which this thesis is published. The right to reuse the content requires the explicit permission of Springer Nature.

Abstract. Sparse coding (SC) is a standard approach to relate neural response properties of primary visual cortex (V1) to the statistical properties of images. SC models the dependency between latent and observed variables using one weight matrix that contains the latents' generative fields (GFs). Here, we present a novel SC model that couples latent and observed variables using two matrices: one matrix for component means and another for component variances. When training on natural image patches, we observe Gabor-like and globular GFs. Additionally, we obtain a second dictionary for each component's variances. The double-dictionary model is thus the first to capture first- and second-order statistics of natural image patches using a multiple-causes latent variable model. If response probabilities of V1 simple cells are not restricted to first order statistics, the investigated model is likely to be more closely aligned with neural responses than standard SCs or independent component analysis (ICA) models.

Keywords: Unsupervised Learning, Maximal Causes Analysis, Non-Linear Sparse Coding, Double-Dictionary Learning, Variational EM

Author Contributions. Literature research was conducted by Jörg Lücke (JL) and Jakob Drefs (JD) with JD having contributed most of the literature research specific to image processing applications. The derivation of the learning rule for the variance dictionary was worked out in parallel by JD and Hamid Mousavi (HM) based on initial input from JL; the derivation presented in Sec. 5.2.2 corresponds to the respective solution worked out by JD. The numerical experiments were implemented and conducted by JD. The manuscript in its published form was written by all authors; the additional, non-published parts presented in this chapter were worked out and written by JD. The illustration of the model architecture (Fig. 5.1 A) was created by HM, remaining visualizations were created by JD with input from JL. All authors discussed the results and approved the manuscript before publication.

Acknowledgments. We acknowledge funding by the German Research Foundation (DFG) in project 390895286 (cluster of excellence H4a 2.0, EXC 2177/1) and the German Ministry of Research and Education (BMBF) project 05M2020 (SPAplus, TP 3).

5.1 Introduction

The two well-known standard models of Sparse Coding (SC; Olshausen and Field, 1997) and Independent Component Analysis (ICA; e.g., Hyvärinen and Oja, 1997) have been successfully used to link the response properties of simple cells in primary visual cortex (V1) to the view of sensory systems as optimal information encoders. Since they have been introduced, it has repeatedly been pointed out that generalizations of the original model assumptions are required, for example extensions of ICA which include an encoding of dependencies between latent activities, addition of intensity variables and latents, hierarchical features, and many more (e.g., Karklin and Lewicki, 2003, 2009; Wainwright and Simoncelli, 2000). Also the encoding of variances of image components has been observed to be important for image processing but has, so far, only been realized for mixture models (e.g., Dai and Lücke, 2014; Zoran and Weiss, 2011). Previous work (Bornschein et al., 2013) has studied the impact of occlusion-like non-linearities on predicted simple cell responses by exploiting a maximum superposition (in contrast to the linear superposition in the standard SC model). Here, we step forward and ask if such established non-linear models can be generalized by coupling latents to observables using two matrices: one to model the means of the observable distribution and another one to model the variances. Such generalization imposes additional challenges on the model and its optimization, including novel derivations of parameter update rules. We will show how an efficient learning algorithm can be realized and how it can be scaled to learn dictionaries with hundreds of components capturing first- and second-order statistics of natural image patch data.

5.2 Methods

5.2.1 A Non-Linear Sparse Coding Model with Mean and Variance Dictionaries

We build upon maximal causes analysis (MCA) models (Bornschein et al., 2013; Lücke and Sahani, 2008; Puertas et al., 2010) with binary latents and Gaussian noise. Binary latents suggest themselves for probabilistic neural encoding (Shivkumar et al., 2018), and the maximum non-linearity of MCA can be motivated by statistical properties of stimuli processed by primary visual cortex (Bornschein et al., 2013; Puertas et al., 2010) and auditory cortex (Sheikh et al., 2019). Concretely, we propose the following generative model:

$$p(\vec{s} \mid \Theta) = \prod_{h=1}^{H} \text{Bernoulli}(s_h; \pi_h), \qquad p(\vec{y} \mid \vec{s}, \Theta) = \prod_{d=1}^{D} \mathcal{N}(y_d; \bar{W}_d(\vec{s}, \Theta), \bar{\Sigma}_d(\vec{s}, \Theta)). \tag{5.1}$$

The binary latents $s_h \in \{0, 1\}$ are distributed according to $\text{Bernoulli}(s_h; \pi_h) = \pi_h^{s_h}(1 - \pi_h)^{(1-s_h)}$ with prior parameters $\pi_h \in [0, 1]$; given the latents, observables $y_d \in \mathbb{R}$ are distributed according to a Gaussian with mean and variance parameters given by:

$$\left.\begin{aligned} \bar{W}_d(\vec{s}, \Theta) &= W_{dh(d,\vec{s},\Theta)} \\ \bar{\Sigma}_d(\vec{s}, \Theta) &= \Sigma_{dh(d,\vec{s},\Theta)} \end{aligned}\right\} \; h(d, \vec{s}, \Theta) = \underset{h}{\text{argmax}} \; | \, s_h W_{dh} \, |, \quad h = 1, \ldots, H. \tag{5.2}$$

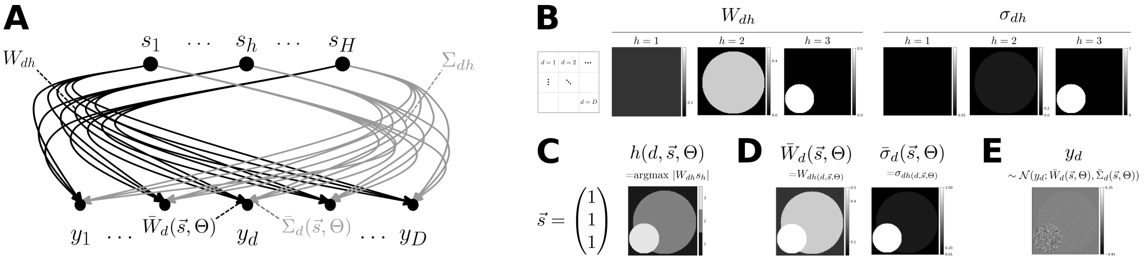

A
W_{dh}
s_1 ... s_h ... s_H
Σ_{dh}
y_1 ... $\bar{W}_d(\vec{s},\Theta)$ y_d $\bar{\Sigma}_d(\vec{s},\Theta)$ ... y_D
B
W_{dh}
σ_{dh}
$h=1$
$h=2$
$h=3$
$d=1$ $d=2$ ...
$d=D$
C
$h(d,\vec{s},\Theta)$
$=\mathrm{argmax}\,|W_{dh}s_h|$
$\vec{s}=\begin{pmatrix}1\\1\\1\end{pmatrix}$
D
$\bar{W}_d(\vec{s},\Theta)$
$=W_{dh(d,\vec{s},\Theta)}$
$\bar{\sigma}_d(\vec{s},\Theta)$
$=\sigma_{dh(d,\vec{s},\Theta)}$
E
y_d
$\sim\mathcal{N}(y_d;\bar{W}_d(\vec{s},\Theta),\bar{\Sigma}_d(\vec{s},\Theta))$

Figure 5.1: Illustration of the proposed DD-MCA model. **A**: Latents s_h are coupled to observables y_d using generative fields (GFs) W_{dh} and Σ_{dh} for component means and component variances, respectively. GFs are combined non-linearly, resulting in $\bar{W}_d(\vec{s},\Theta)$ and $\bar{\Sigma}_d(\vec{s},\Theta)$. **B**: H=3 exemplary circle-like GFs for means and variances. **C**: Non-linearity from Eq. (5.2) resulting from the GFs from B and a given latent vector $\vec{s}$. **D**: Component means and variances combined non-linearly according to C. In B and D, we display standard deviations rather than variances to improve visibility ($\Sigma_{dh}=\sigma_{dh}^2$). In B and D, component means are displayed using the same color scale, and the same applies for component variances. **E**: Exemplary data point drawn from a Gaussian with mean and variance parameter from D (see text for details).

We refer to $W=\{\vec{W}_h\}_{h=1}^{H}$ as a dictionary of generative fields (GFs) for component means $\vec{W}_h=\{W_{dh}\}_{d=1}^{D}$ with $W_{dh}\in\mathbb{R}$, and, analogously, to $\Sigma=\{\vec{\Sigma}_h\}_{h=1}^{H}$ as a dictionary of GFs for component variances $\vec{\Sigma}_h=\{\Sigma_{dh}\}_{d=1}^{D}$ with $\Sigma_{dh}\in\mathbb{R}^{+}$. The parameters of the model are $\Theta=(\vec{\pi},W,\Sigma)$, where $\vec{\pi}=\{\pi_h\}_{h=1}^{H}$. Component means and component variances combine non-linearly: the mean $\bar{W}_d(\vec{s},\Theta)$ and variance $\bar{\Sigma}_d(\vec{s},\Theta)$ of the Gaussian in Eq. (5.1) are determined by the component h yielding the maximal magnitude of s_hW_{dh} (Eq. (5.2)). To ensure that $\bar{\Sigma}_d(\vec{s},\Theta)$ always has a positive value, we force one hidden unit to be constantly active (specifically, we set $s_1=\mathrm{const}=1$, where the choice $h=1$ is arbitrary). Since the respective component means and variances determine the minimal amplitudes of $\bar{W}_d(\vec{s},\Theta)$ and $\bar{\Sigma}_d(\vec{s},\Theta)$, we refer to $\vec{W}_1$ and $\vec{\Sigma}_1$ as background fields.

Non-linear latent interactions based on the maximum have been investigated before: Lücke and Sahani (2008) trained MCA models with Poisson distributed observables, in which the Poisson mean is determined similarly to the definition of Eq. (5.2); as a distinction, Lücke and Sahani used a maximum, rather than a maximum magnitude non-linearity as considered here. Similar to the approach proposed here, Bornschein et al. (2013); Puertas et al. (2010) used Gaussian observables and applied the non-linearity $h(d,\vec{s},\Theta)$ form Eq. (5.2) to set the Gaussian mean parameter (Puertas et al. used, similar to Lücke and Sahani, a maximum rather than a maximum magnitude non-linearity). The MCA model considered here (Eqs. (5.1) and (5.2)) extends previous Gaussian MCA variants by capturing individual variances Σ_{dh} for each latent dimension h and each observed dimension d (previous variants used a global, scalar variance parameter $\Sigma_{dh}=\sigma^2\,\forall\,d=1,\ldots,D,\,\forall\,h=1,\ldots,H$). Fig. 5.1 illustrates the proposed generative model; we will refer to this MCA variant as *Double-Dictionary Maximum Causes Analysis* (DD-MCA) in the remainder of this paper.

5.2.2 Parameter Optimization

To fit the DD-MCA model to a set of data points $\{\vec{y}^{(n)}\}_{n=1,\ldots,N}$, we seek parameters Θ^* that maximize the data log-likelihood $\mathcal{L}(\Theta)=\sum_{n=1}^{N}\log(p(\vec{y}^{(n)}\mid\Theta))$, with $p(\vec{y}\mid\Theta)=\sum_{\vec{s}}p(\vec{y}\mid$

$\vec{s}, \Theta)\, p(\vec{s} \mid \Theta)$ given by Eq. (5.1), and $\sum_{\vec{s}}$ denoting summation over all possible binary latent states. For our purposes, we apply a variational expectation maximization approach (EM; e.g., Neal and Hinton, 1998; Saul and Jordan, 1995) and optimize a log-likelihood lower bound, also referred to as free energy:

$$\mathcal{F}(q, \Theta) = \sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\big[\log\big(p(\vec{y}^{\,(n)} \mid \vec{s}, \Theta)\; p(\vec{s} \mid \Theta)\big)\big] + \sum_{n=1}^{N} \mathcal{H}(q^{(n)}) \leq \mathcal{L}(\Theta), \tag{5.3}$$

with $\mathbb{E}_{q^{(n)}}\big[g(\vec{s})\big] = \sum_{\vec{s}} q^{(n)}(\vec{s}) g(\vec{s})$ and $\mathcal{H}(q^{(n)}) = \mathbb{E}_{q^{(n)}}\big[-\log(q^{(n)}(\vec{s}))\big]$ denoting expectation and Shannon entropy, respectively. $q^{(n)}(\vec{s})$ are variational distributions that approximate the exact posterior $p(\vec{s} \mid \vec{y}^{\,(n)}, \Theta)$. The free energy objective in Eq. (5.3) is optimized alternately with respect to q holding Θ fixed, referred to as the E-step, and with respect to Θ holding q fixed, referred to as the M-step. Here, we use truncated posteriors (e.g., Hirschberger et al., 2022; Lücke and Forster, 2019; Sheikh et al., 2014) as family of variational distributions, and apply Evolutionary Variational Optimization (EVO) to realize highly efficient E-steps (for details, see Drefs et al., 2022). For the M-step, parameter update rules can be obtained by solving $\frac{\partial}{\partial \Theta}\mathcal{F}(q, \Theta) \overset{!}{=} 0$ for Θ. For the parameters W and $\vec{\pi}$, update equations have been derived before (Bornschein et al., 2013; Lücke and Sahani, 2008) and are given by:

$$W_{dh}^{\mathrm{new}} = \frac{\sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\big[\mathcal{A}_{dh}(\vec{s}, \Theta)\big] y_d^{(n)}}{\sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\big[\mathcal{A}_{dh}(\vec{s}, \Theta)\big]}, \qquad \pi_h^{\mathrm{new}} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\big[s_h\big], \tag{5.4}$$

where $\mathcal{A}_{dh}(\vec{s}, \Theta) = 1$ if $h = h(d, \vec{s}, \Theta)$ and 0 otherwise. Here, we provide a derivation of the M-step update equation for Σ_{dh}. To start, we require that:

$$0 \overset{!}{=} \frac{\partial}{\partial \Sigma_{dh}} \mathcal{F}(q, \Theta) \tag{5.5}$$

$$= \frac{\partial}{\partial \Sigma_{dh}} \sum_{n=1}^{N} \sum_{\vec{s}} q^{(n)}(\vec{s}) \log\Big(p(\vec{y}^{\,(n)} \mid \vec{s}, \Theta)\Big) \tag{5.6}$$

$$= \sum_{n=1}^{N} \sum_{\vec{s}} q^{(n)}(\vec{s}) \sum_{d'=1}^{D} \left(\frac{\partial}{\partial \Sigma_{dh}} \log\Big(p\big(y_{d'}^{(n)} \mid \bar{W}_{d'}(\vec{s}, W), \bar{\Sigma}_{d'}(\vec{s}, W, \Sigma)\big)\Big)\right) \tag{5.7}$$

$$= \sum_{n=1}^{N} \sum_{\vec{s}} q^{(n)}(\vec{s}) \left(\frac{\partial}{\partial \Sigma_{dh}} \bar{\Sigma}_d(\vec{s}, W, \Sigma)\right) f\big(y_d^{(n)}, \bar{W}_d(\vec{s}, W), \bar{\Sigma}_d(\vec{s}, W, \Sigma)\big), \tag{5.8}$$

$$\text{where} \quad f(y, w, \sigma^2) = \frac{\partial}{\partial \sigma^2} \log(p(y \mid w, \sigma^2)).$$

Given that

$$\frac{\partial}{\partial \Sigma_{dh}} \bar{\Sigma}_d(\vec{s}, W, \Sigma) = \frac{\partial}{\partial \Sigma_{dh}} \Sigma_{dh'(d,\vec{s},W)} = \delta_{hh'(d,\vec{s},W)}, \tag{5.9}$$

we can rewrite Eq. (5.8) as

$$0 \overset{!}{=} \sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\big[\delta_{hh'(d,\vec{s},W)}\big] f(y_d^{(n)}, W_{dh}, \Sigma_{dh}), \tag{5.10}$$

and since $p(y \mid w, \sigma^2) = \mathcal{N}(y \mid w, \sigma^2)$ such that $f(y, w, \sigma^2) = \frac{1}{2\sigma^2}(\frac{1}{\sigma^2}(y-w)^2 - 1)$, we obtain

$$\sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\left[\delta_{hh'(d,\vec{s},W)}\right] = \sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\left[\delta_{hh'(d,\vec{s},W)}\right] \frac{1}{\Sigma_{dh}} (y_d^{(n)} - W_{dh})^2, \tag{5.11}$$

which can be rearranged yielding the following update equation:

$$\Sigma_{dh}^{\text{new}} = \frac{\sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\left[\mathcal{A}_{dh}(\vec{s}, \Theta)\right] \left(y_d^{(n)} - W_{dh}\right)^2}{\sum_{n=1}^{N} \mathbb{E}_{q^{(n)}}\left[\mathcal{A}_{dh}(\vec{s}, \Theta)\right]}, \tag{5.12}$$

where we use the notation $\mathcal{A}_{dh}(\vec{s}, \Theta)$ instead of $\delta_{hh'(d,\vec{s},W)}$ for consistency with Eq. (5.4).

5.3 Numerical Experiments

5.3.1 Bars Test

We verified the optimization of the DD-MCA model, in particular Eq. (5.12), using a standard component extraction task, namely a bars test (Földiák, 1990; Hoyer, 2003; Lücke and Sahani, 2008). To generate training data, we created mean and variance dictionaries containing GFs in the form of horizontal and vertical bars. We then stacked these GFs together with one additional field for background intensity. Our dictionary contained $H = 11$ GFs, each reshaped as a $D = 5 \times 5$ image. For the background fields, we used constant amplitudes of $\vec{W}_1^{\text{gen}} = 1.0$ and $\vec{\sigma}_1^{\text{gen}} = 0.5$. The remaining GFs contained the same background value, but were additionally overlaid by different bar patterns: for $\vec{W}_h^{\text{gen}}$ with $h = 2, \ldots, H$, bar amplitudes were varied in equidistant steps between $[10.0, 10.9]$, while for the respective fields $\vec{\sigma}_h^{\text{gen}}$, we manually selected bar amplitudes in the range $[1, 5]$. Using priors $\pi_1^{\text{gen}} = 1.0$ and $\pi_h^{\text{gen}} = 0.2$ for $h = 2, \ldots, H$, we sampled $N = 1000$ data points according to Eqs. (5.1) and (5.2). After initialization[9], we used M-steps defined by Eqs. (5.4) and (5.12) and EVO for variational E-steps (we applied the fitparents-cross-sparseflip variant of the algorithm with hyperparameters $S = 30$ and $N_p = 20$, see Drefs et al., 2022 for details) to train a DD-MCA model for 50 epochs on the generated data set. After training, we observed that generating parameters were accurately recovered (Fig. 5.2).

5.3.2 Encoding of Natural Image Patches

Having verified the proposed approach using bars tests, we turned to realistic data and fitted a DD-MCA model to natural image patches. Our training dataset contained patches of size $D = 12 \times 12$ pixels, randomly sampled from a publicly available data set of natural images (van Hateren and van der Schaaf, 1998). We clamped the highest 2 % of the pixel amplitudes to attenuate the influence of very high light intensities and subsequently applied Zero-Phase Component Analysis (Bell and Sejnowski, 1997) to whiten the data, using the set of principal components corresponding to 95 % of the data variance (compare Exarchakis and Lücke, 2017). After whitening, we collected $N = 100000$ training data points, omitting

---

[9] π_h^{init} were randomly uniformly drawn from the interval $[0.1, 0.5]$ for $h = 1, \ldots, H$; $W_{dh}^{\text{init}} = \bar{y}_d + \varepsilon$ and $\sigma_{dh}^{\text{init}} = \sigma_{\mathcal{Y}}$ for $d = 1, \ldots, D$ and $h = 1, \ldots, H$ with $\bar{y}_d = \frac{1}{N}\sum_{n=1}^{N} y_d^{(n)}$, $\sigma_{\mathcal{Y}}^2 = \frac{1}{DN}\sum_{d=1}^{D}\sum_{n=1}^{N}(y_d^{(n)} - \bar{y}_d)^2$, and $\epsilon \sim \mathcal{N}(0, \frac{\sigma_{\mathcal{Y}}}{4})$.

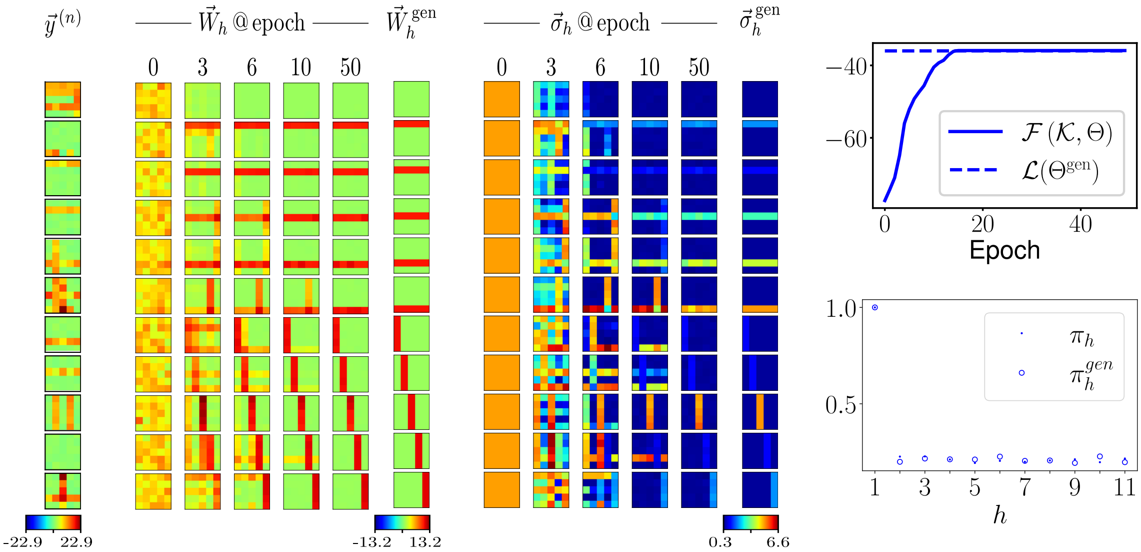


Figure 5.2: Results of the bars test for the DD-MCA model (see text for details). For ease of comparison, we display $\vec{W}_h$ and $\vec{\sigma}_h$ in the same order as the respective GFs $\vec{W}_h^{\text{gen}}$ and $\vec{\sigma}_h^{\text{gen}}$; note that the learned order corresponded to a permutation of the respective ground-truth one.

patches with lowest variance to avoid capturing image sections without significant structure (e.g., flat areas). We then initialized[10] a DD-MCA model with $H = 1000$ components and applied the optimization described above (here using EVO with $S = N_p = 60$ and the fitparents-randflip variant of the algorithm to increase computational efficiency, compare Drefs et al., 2022) to fit the model to the image patches data set. First, we performed 250 training epochs optimizing only W and $\vec{\pi}$ and keeping Σ at its initial value before updating all parameters jointly for 50 epochs (we found this procedure to lead to a numerically more stable behavior of the algorithm compared to updating all parameters jointly from the beginning). Figure 5.3 depicts the result of the experiment. After training, DD-MCA had learned a sparse encoding of image patches with on average approximately 6 out of 1000 active latents per data point. Inspecting the learned dictionaries, we observed a large variety of GFs for the component means (including the familiar Gabor-like and globular fields), as well as a large variety of GFs for the component variances.

If we identify image patches that maximally activate a given latent, we can systematically modify these patches while measuring changes in the latent's responses. We observed that specific modifications, such as addition of certain types of noise, can lead to differences in the response properties of the latent depending on how it encodes component variances. For instance, noise added in proximity of a Gabor component or distant from it has a different effect for latents with individual variances compared to latents with the same variance for all pixels and all components (as used for standard SC). Such differences in responses could be used to design stimuli that could enable the detection of a potential variance encoding in V1 and other sensory areas.

---

[10] $\vec{\pi}^{\text{init}}$ and Σ^{init} were set similarly to the bars test. $\vec{W}_h^{\text{init}}$, $h = 1, \ldots, H$, were set equal to H cluster centers obtained by running var-GMM-S clustering (Forster and Lücke, 2018; Hirschberger et al., 2022) on our data set using publicly available source code together with default hyperparameters from example code (Hirschberger, 2019).

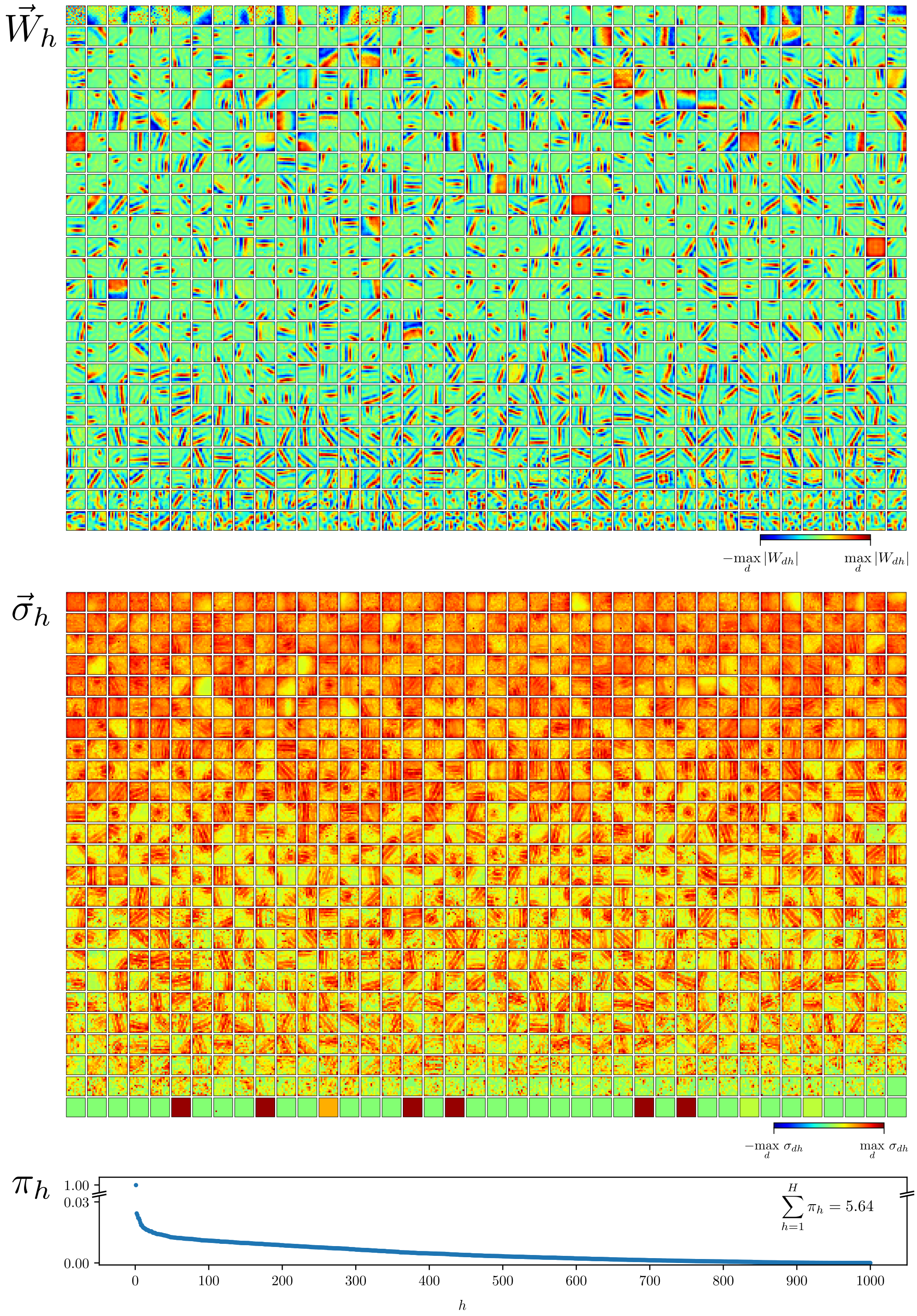


Figure 5.3: DD-MCA encoding learned from natural image patches. The GFs are displayed using a local and symmetric color scale and are sorted by prior activation, from left to right and top to bottom, from most active to least active. See text for details.

5.4 Conclusion

We presented DD-MCA, a non-linear SC model which is capable of learning component variances alongside component means. The model represents a novel variant of maximum causes analysis (Bornschein et al., 2013; Lücke and Sahani, 2008; Puertas et al., 2010) with individual variance parameters per component and per observable, and, as such, provides a generalization of the global isotropic variance in standard SC models (Olshausen and Field, 1997). Our numerical results demonstrate that DD-MCA can reliably and efficiently be trained using evolutionary variational optimization (Drefs et al., 2022) and a newly derived learning rule for the variance dictionary. When training on natural image patch data, the learned GFs for component means capture a variety of structures, many of which corresponding to Gabor-like and globular fields, confirming observations reported for previous MCA models (e.g., Bornschein et al., 2013). The similarly diverse GFs for component variances that we observed for DD-MCA (Fig. 5.3) most clearly illustrate the contrast to the assumption of a global uniform variance of standard SC approaches. That non-elementary variance encodings of images or image components can be beneficial for image processing has been demonstrated for a range of applications including, e.g., image restoration (e.g., Dai and Lücke, 2014; Zoran and Weiss, 2011) or image classification (e.g., Deng et al., 2012; Liu et al., 2018). From a biological perspective, it may be argued that a single variance parameter is statistically sub-optimal and that primary visual cortex may, likewise, represent component variances as well as component means. Based on our numerical results, we have pointed out how variance encoding could in principle be detected in novel *in vivo* experiments.

Chapter 6

Discussion and Conclusion

Machine learning algorithms have become integral part of numerous technologies which are influencing the everyday lives of millions of people, e.g., through their use by content recommendation systems applied by social networks or streaming platforms (e.g., Zhang et al., 2019), online text translation services (e.g., Otter et al., 2021), or autonomous vehicles (e.g., Muhammad et al., 2021), to name just a few examples. A decisive contribution to this development has been made by the increasing availability of data and computational resources over the last decades. Deep Learning (DL) models, i.e., models incorporating deep neural networks (DNNs) such as multi-layer perceptrons (Ivakhnenko and Lapa, 1965; Rosenblatt, 1958), convolutional networks (e.g., LeCun et al., 1989), recurrent networks (e.g., Hochreiter and Schmidhuber, 1997), or other types of networks are currently exceptionally widely used. DNNs offer a high degree of flexibility regarding their parameterization and provide a generic approach for modeling mappings between sets of data (compare, e.g., Hornik et al., 1989). In recent years, a widely adopted strategy in DL approaches for optimizing performance has been to develop DNNs with increasingly large architectures and increasingly many parameters. For instance, recent language models use on the order of up to hundreds of billions of parameters (Brown et al., 2020). DL methodologies powered by increasingly large DNNs have shown impressive performances on computer vision, natural language processing, and many other tasks. Such strong performances, at the same time, rely on considerable amounts of data used for training (i.e., for the optimization of the parameters of the model). For supervised approaches, data sets thereby need to contain large numbers of training pairs consisting of input and target values. Training increasingly large models on increasingly large data sets is accompanied by an increasing demand for computing resources. The computational cost of training DL models has in fact rapidly increased in recent years with considerable environmental implications due to the associated energy consumption and carbon footprint, as discussed, e.g., by Schwartz et al. (2020). Moreover, Floridi and Chiriatti argue that the strong performance of models that derive much of their power from massive amounts of data may be strongly constrained to the tasks for which they were optimized, with an additional issue being that models pick up biases in the data (Floridi and Chiriatti, 2020).

Variational autoencoders (VAEs; Kingma and Welling, 2014; Rezende et al., 2014) represent a class of ML models that use DNNs as part of a probabilistic generative model, which gives them several attractive features: VAEs incorporate an explicitly formulated data density model that allows assumptions about the data generation process to be incorporated into the model. The density model thereby uses a flexible parameterization described by DNNs. Moreover, VAEs allow for unsupervised training. Like other probabilistic generative models, VAEs face the challenge of intractable posterior distributions and must resort to approximate inference methods. To this end, VAEs use encoding models that are parameterized by DNNs and whose parameters are optimized using the backpropagation algorithm (Rumelhart et al., 1986), with the latter point providing an additional link between VAEs and discriminative DL models. To maintain backpropagation in a variational setting (in which a variational lower bound of the log-likelihood serves as optimization objective), VAE approaches, as discussed in Ch. 4, typically incorporate a series of auxiliary methods that may be considered sub-optimal from a probabilistic modeling perspective: (i) While amortized inference provides benefits in terms of computational efficiency, it suffers from an inherent inaccuracy due to the amortization gap. (ii) Assuming a specific form, e.g., a factored Gaussian form, for the variational distributions implies that properties such as correlations and multi-modal structures of the true posterior distribution cannot be captured. (iii) Continuous relaxations of discrete distributions applied in the case of discrete latent variables do not allow for keeping the discrete nature of latents fully intact.

With Evolutionary Variational Optimization (EVO), this thesis has, as one of its major contributions, developed a novel generic training method for probabilistic generative models with binary latents that avoids the aforementioned sub-optimalities. The approximate inference scheme employed by EVO differs conceptually in several aspects from approximation methods used by standard VAEs: By choosing truncated posteriors as variational distributions (Eq. (2.2)), EVO makes no assumption of a-posteriori independence of the latents, nor of a mono-modal form of the posterior distribution. Instead, EVO's approximation scheme relies on the assumption that the posterior mass is concentrated on small subsets of the latent space. As a further distinction, EVO does not involve a DNN-based parameterization of variational distributions and does not apply gradient-based optimization of variational parameters. Instead, the parameters of the truncated posteriors, i.e., the subsets $\mathcal{K}^{(n)}$, are optimized using evolutionary algorithms, and they are optimized per data point, implying that EVO does not suffer the amortization gap.

EVO can be considered as a 'black box' inference method for generative models with binary latents in the sense that it is generically applicable to models with analytically tractable joint probability without requiring model-specific derivations. The truncated variational distributions and the optimization of their parameters through evolutionary algorithms is fully defined in terms of the model's joint (Alg. 1). Moreover, the inference scheme of EVO is compatible with gradient-based optimization of model parameters using automatic differentiation tools (Alg. 2), implying that EVO allows for realizing a fully automated training algorithm. EVO can be distinguished from related 'black box' variational inference approaches such as BBVI (Ranganath et al., 2014) in that (i) BBVI, in contrast to EVO, uses a stochastic gradient-based optimization of the variational objective, which (ii) assumes the variational distributions to take a specific, i.e., factored form, and (iii) BBVI requires to derive gradients and sampling procedures for the concrete variational distribution used. Further approaches related in this context but conceptually different to EVO due to their use of explicit forms for the variational distribution include works such as Kingma and Welling (2014); Kucukelbir et al. (2017); Rezende and Mohamed (2015); Rezende et al. (2014).

EVO's generic applicability and viability have been validated through the studies presented in Chs. 2 to 5, in which the approach was successfully used for the optimization of data models of varying complexity, including linear sparse coding models with binary prior and spike-and-slab prior (BSC and SSSC, respectively), Noisy-OR Bayes nets (NOR), maximal causes analysis models (MCA), and VAEs with binary latents (TVAE). The approach has here shown to be scalable to large problem sizes, as exemplified, e.g., by the SSSC models considered in Ch. 2, which used up to 900 latents and on the order of 100K data points of size $14 \times 14 \times 3$ pixels for training (Fig. 2.7 F), or the VAEs in Ch. 4, which used decoder architectures with up to 1024 binary latents, two hidden layers with up to 512 units, and 50K data points of size $32 \times 32 \times 3$ pixels for training. At the same time, for large scale problems, EVO (or more specifically, the here used implementations of the algorithm) showed to be associated with considerable runtimes (e.g., on the order of several hours for processing single images; cf. Sec. 2.S6.1). While this issue could be addressed by distributing the training over many (i.e., up to thousands of) CPU cores, the runtimes still remained considerably higher, e.g., compared to approaches using amortized inference (Tab. 4.S3). Amortized inference schemes have, at the same time, by definition a lower computational cost compared to EVO (compare Sec. 4.S2.4), and their implementations of DNN-based encoding models can fully exploit the efficiency of deep learning tools such as Tensorflow (Abadi et al., 2016) or PyTorch (Paszke et al., 2019). The efficiency of the implementations of EVO, on the other hand, can presumably still significantly be improved.

After confirming the viability and scalability of EVO, the studies presented extensively investigated the performance of, in particular, BSC, SSSC and VAE models accelerated by EVO (termed EBSC, ES3C, and TVAE, respectively) in concrete applications, focusing on denoising and inpainting tasks. The large body of performance evaluations and the insights they provide may be considered further major contributions of this thesis. One goal of these investigations was to provide systematic comparisons between EVO and other approximate inference schemes. One observation that could be made in this respect was that the performance of the approaches considered has here shown to depend on the type of data. For instance, for image patch modeling and Gaussian denoising tasks, EVO showed to provide performance improvements over the sampling-based method by Zhou et al. (2012), the mean field approach by Titsias and Lázaro-Gredilla (2011), the inference scheme used by Sheikh et al. (2014) which is based on truncated posteriors and preselection procedures, and the Gumbel-softmax approach by Jang et al. (2017) (Figs. 2.5 B and 4.2). In turn, for other types of data, specifically for image data consisting of single objects (i.e., CIFAR-10; Krizhevsky, 2009), performances of EVO were improved by, e.g., the Gumbel-softmax approximation (Tabs. 2.S3 and 4.S4). One striking observation was that EVO and the Gumbel-softmax approximation resulted in considerably different data encodings in terms of sparsity of the encoding and structure of the learned generative fields (Figs. 2.S4 and 4.2 and Tab. 4.S4).

A further goal of the conducted performance evaluations was to provide insight into how algorithms developed here such as ES3C and TVAE compare to DL approaches and other methods that are not necessarily based on probabilistic models and are optimized specifically for denoising tasks. Among these approaches, N2V (Krull et al., 2019a), DivN (Prakash et al., 2021b), and S2S (Quan et al., 2020a) may be considered the most similar to ES3C and TVAE in the sense that all these algorithms are applicable in a setting with only noisy data available. In comparison to N2V, DivN, and S2S, the ES3C and TVAE approaches have shown to be highly competitive (Figs. 3.4 and 4.1 D). Maybe more remarkably, in some conditions of the benchmarks in Figs. 2.5 D, 2.6, and 4.1 D, the performances of ES3C and TVAE (which were obtained under 'zero-shot' conditions, i.e., using only a single noisy image for training) turned out to be competitive even with results obtained by methods that use significantly more a-priori information, specifically supervised DL models such as DnCNN (Zhang et al., 2017), TNRD (Chen and Pock, 2017) or BDGAN (Zhu et al., 2019), all of which leverage large databases with clean images for training. This result provides an important insight: Expressive probabilistic generative models optimized using flexible approximate inference schemes can be comparably effective, for the considered denoising application, compared to DL models that derive much of their power from intricate DNNs and large training data sets. Besides, Figs. 3.2, 3.4 and 4.S12 may provide insight into the generalizability of the performance of the methods developed here w.r.t. different data types: These results exemplify that, for denoising, the strong performance of ES3C and TVAE is not specific to natural image data, but can also be observed, e.g., for microscopy imaging data or acoustic data.

Like for the vast majority of ML methods, the performance of the algorithms developed showed to rely on a careful choice of hyperparameters. These include hyperparameters related to the data model (e.g., dictionary size H), data preprocessing (e.g., patch sizes $\mathcal{P}_\mathrm{x}$ and $\mathcal{P}_\mathrm{y}$), and variational approximation (i.e., the number of variational states $S = |\mathcal{K}^{(n)}|$ and the hyperparameters of the evolutionary algorithms used by EVO, e.g., the number of parental states N_p, the number of children N_c, and the number of generations N_g; compare Sec. 2.2.1). For the majority of the experiments presented in Chs. 2 to 5, hyperparameter optimization (HPO) involved manual tuning, which may be argued to be inefficient and sub-optimal. First steps towards automating this step were taken in Ch. 4, in which HPO tools (Falkner et al., 2018; Nogueira, 2019) were used to tune, e.g., decoder architectures and learning rate scheduler hyperparameters of TVAE (using the variational lower bound as HPO objective). Exploiting recent advances in the field of automated machine learning (AutoML), specifically state-of-the-art methods for HPO and neural architecture search (e.g., Lindauer et al., 2022; Sass et al., 2022; Zimmer et al., 2021), more systematically appears valuable for any form of follow-up work of this thesis. A desirable goal in this context may be to conduct HPO fully unsupervised. For 'zero-shot' image enhancement applications, for example, future efforts may follow the methodology applied in Ch. 4 and use the variational lower bound (which can be computed based on noisy data and which has here shown to be informative about enhancement performance in terms of peak-signal-to-noise ratios, Fig. 2.4) as HPO objective; at the same time, future efforts may aim at seeking alternative HPO objectives that allow for compensating for differences in model and data dimensionality. By combining the feature of 'black box' training (as discussed above) with AutoML tools, workflows could be realized where models are implemented solely in terms of their joint distribution and driven to peak performance fully autonomously.

For such and other types of follow-up work, developments on the software side may make use of the implementations provided by the TVO library which was developed in the context of the presented studies. Besides being useful for reproducibility of the results reported here, the TVO library may represent a valuable resource for future practitioners and developers applying and working with the algorithms developed here. To ease usage and extensibility, the library is organized using a modular design with a clear separation of independent components such as routines specific to data models and variational optimization. TVO features multi-core CPU and GPU acceleration, enabling distributed execution of the algorithms. It incorporates continuous integration workflows and software documentation to promote high code quality. The TVO library has been open sourced (TVO developers, 2022 and also compare Guiraud, 2021), and it represents a further major contribution of the studies presented in this thesis.

6.1 Outlook

This work has opened the door to a variety of possible future research directions. One such direction may be the investigation of evolutionary variational optimization of further interesting and challenging data models, for example VAEs with spike-and-slab priors and convolutional decoder networks. Continued research could also aim at improving EVO through incorporation of more sophisticated evolutionary algorithms, starting, e.g., with more disruptive crossover operations (e.g., Eshelman, 1991). Efforts in this direction would concretely aim at increasing the ratio of states $\vec{s}^{\,\mathrm{new}}$ evolved per data point and epoch which satisfy the criterion defined by Eq. (2.5). A further goal for follow-up work may be the improvement of computational efficiency. Besides efforts on the implementation side, which may potentially be inspired by work by Hirschberger et al. (2022), it might be interesting to ask whether ideas of amortization could possibly be combined with EVO's inference scheme. With regards to image enhancement applications, future work could pick up ideas by Zoran and Weiss (2011) and incorporate the concept of expected patch log-likelihood into the image processing scheme employed here (Fig. 3.5). Future research could also investigate fully automated workflows incorporating 'black box' training and AutoML tools for hyperparameter optimization (as discussed above); interesting research questions in this respect may be the general viability of such approaches and the benefits they might provide in terms of task performance.

Publication List

Peer-Reviewed Journal Articles and Conference Papers

Jakob Drefs*, Enrico Guiraud*, Filippos Panagiotou, Jörg Lücke. Direct Evolutionary Optimization of Variational Autoencoders with Binary Latents. In Massih-Reza Amini et al. (Eds.): *Machine Learning and Knowledge Discovery in Databases*. ECML PKDD 2022. Lecture Notes in Computer Science, vol. 13715, pp. 357-372, 2023. Springer, Cham. `https://doi.org/10.1007/978-3-031-26409-2_22`. *Joint first authorship.

Jakob Drefs, Enrico Guiraud, Jörg Lücke. Evolutionary Variational Optimization of Generative Models. *The Journal of Machine Learning Research*, 23(21):1-51, 2022.

Hamid Mousavi, Mareike Buhl, Enrico Guiraud, Jakob Drefs, Jörg Lücke. Inference and Learning in a Latent Variable Model for Beta Distributed Interval Data. *Entropy*, 23(5), 2021. `https://doi.org/10.3390/e23050552`.

Hamid Mousavi*, Jakob Drefs*, Jörg Lücke. A Double-Dictionary Approach Learns Component Means and Variances for V1 Encoding. In Giuseppe Nicosia et al. (Eds.): *Machine Learning, Optimization, and Data Science*. LOD 2020. Lecture Notes in Computer Science, vol. 12566, pp. 240–244, 2020. Springer, Cham. `https://doi.org/10.1007/978-3-030-64580-9_20`. *Joint first authorship.

Abdul-Saboor Sheikh*, Nicol S. Harper*, Jakob Drefs, Yosef Singer, Zhenwen Dai, Richard E. Turner, Jörg Lücke. STRFs in primary auditory cortex emerge from masking-based statistics of natural sounds. *PLOS Computational Biology*, 15(1):e1006595, 2019. `https://doi.org/10.1371/journal.pcbi.1006595`. *Joint first authorship.

Enrico Guiraud, Jakob Drefs, Jörg Lücke. Evolutionary Expectation Maximization. In *Genetic and Evolutionary Computation Conference*, pp. 442–449. ACM, 2018. `https://doi.org/10.1145/3205455.3205588`.

Preprints

Jakob Drefs, Sebastian Salwig, Jörg Lücke. Visualization of SARS-CoV-2 Infection Scenes by 'Zero-Shot' Enhancements of Electron Microscopy Images. *bioRxiv* 2021.02.25.432265, 2021. `https://doi.org/10.1101/2021.02.25.432265`.

Hamid Mousavi, Jakob Drefs, Florian Hirschberger, Jörg Lücke. Maximal Causes for Exponential Family Observables. In revision for *The Journal of Machine Learning Research*, 2022. *arXiv preprint* arXiv:2003.02214.

Georgios Exarchakis, Jörg Bornschein, Abdul-Saboor Sheikh, Zhenwen Dai, Marc Henniges, Jakob Drefs, Jörg Lücke. ProSper – A Python Library for Probabilistic Sparse Coding with Non-Standard Priors and Superpositions. *arXiv preprint* arXiv:1908.06843, 2019.

Bibliography

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A System for Large-Scale Machine Learning. In *USENIX Symposium on Operating Systems Design and Implementation*, 2016.

K. Ahmadi and E. Salari. Single-Image Super Resolution using Evolutionary Sparse Coding Technique. *IET Image Processing*, 11(1):13–21, 2016.

E. Angelino, M. J. Johnson, and R. P. Adams. Patterns of Scalable Bayesian Inference. *Foundations and Trends in Machine Learning*, 9(2-3):119–247, 2016.

L. Azzari and A. Foi. Variance Stabilization for Noisy+Estimate Combination in Iterative Poisson Denoising. *IEEE Signal Processing Letters*, 23(8):1086–1090, 2016.

T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, 1996.

A. J. Bell and T. J. Sejnowski. The “independent components” of natural scenes are edge filters. *Vision Research*, 37(23):3327–38, 1997.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy Layer-Wise Training of Deep Networks. In *Advances in Neural Information Processing Systems*, 2007.

Y. Bengio, N. Léonard, and A. C. Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432*, 2013.

T. Bepler, K. Kelley, A. J. Noble, and B. Berger. Topaz-Denoise: General Deep Denoising Models for cryoEM and cryoET. *Nature Communications*, 11(1):1–12, 2020.

A. Berliner, G. Rotman, Y. Adi, R. Reichart, and T. Hazan. Learning Discrete Structured Variational Auto-Encoder using Natural Evolution Strategies. In *International Conference on Learning Representations*, 2022.

R. Bhalley. Sc09 spoken numbers. `https://www.kaggle.com/rahulbhalley/sc09-spoken-numbers/`, 2018. Accessed: 2021-11-06.

S. A. Bigdeli, G. Lin, T. Portenier, L. A. Dunbar, and M. Zwicker. Learning Generative Models using Denoising Density Estimators. *arXiv preprint arXiv:2001.02728*, 2020.

E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep Universal Probabilistic Programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.

J. Bornschein, M. Henniges, and J. Lücke. Are V1 receptive fields shaped by low-level visual occlusions? A comparative study. *PLOS Computational Biology*, 9(6):e1003062, 2013.

A. Bouchard-Côté and M. I. Jordan. Optimization of Structured Mean Field Objectives. In *Conference on Uncertainty in Artificial Intelligence*, 2009.

S. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating Sentences from a Continuous Space. In *SIGNLL Conference on Computational Natural Language Learning*, 2016.

D. M. Bradley and J. A. Bagnell. Differentiable Sparse Coding. In *Advances in Neural Information Processing Systems*, 2008.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. volume 33, pages 1877–1901, 2020.

H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain Neural Networks compete with BM3D? In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Y.-J. Cao, L.-L. Jia, Y.-X. Chen, N. Lin, C. Yang, B. Zhang, Z. Liu, X.-X. Li, and H.-H. Dai. Recent Advances of Generative Adversarial Networks in Computer Vision. *IEEE Access*, 7:14985–15006, 2019.

S. Chaudhury and H. Roy. Can fully convolutional networks perform well for general image restoration problems? In *International Conference on Machine Vision Applications*, 2017.

G. Chen, F. Zhu, and P. Ann Heng. An Efficient Statistical Method for Image Noise Level Estimation. In *IEEE International Conference on Computer Vision*, 2015a.

G. Chen, F. Zhu, and P. Ann Heng. Noise Level Estimation for Signal Image. *GitHub repository.* `https://github.com/zsyOAOA/noise_est_ICCV2015`, 2015b. Accessed: 2021-12-01.

Y. Chen and T. Pock. Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2017.

Y.-J. Chen, Y.-J. Chang, S.-C. Wen, Y. Shi, X. Xu, T.-Y. Ho, Q. Jia, M. Huang, and J. Zhuang. Zero-Shot Medical Image Artifact Reduction. In *IEEE International Symposium on Biomedical Imaging*, 2020.

R. Child, S. Gray, A. Radford, and I. Sutskever. Generating Long Sequences with Sparse Transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Y. Cong, M. Zhao, K. Bai, and L. Carin. GO Gradient for Expectation-Based Objectives. In *International Conference on Learning Representations*, 2019.

P. Covington, J. Adams, and E. Sargin. Deep Neural Networks for YouTube Recommendations. In *ACM Conference on Recommender Systems*, 2016.

C. Cremer, X. Li, and D. Duvenaud. Inference Suboptimality in Variational Autoencoders. In *International Conference on Machine Learning*, 2018.

A. Creswell and A. A. Bharath. Denoising Adversarial Autoencoders. *IEEE Transactions on Neural Networks and Learning Systems*, 30(4):968–984, 2018.

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image Denoising by Sparse 3D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Processing*, 16 (8):2080–2095, 2007.

B. Dai, Y. Wang, J. Aston, G. Hua, and D. Wipf. Connections with robust pca and the role of emergent sparsity in variational autoencoder models. *The Journal of Machine Learning Research*, 19(1):1573–1614, 2018.

Z. Dai and J. Lücke. Autonomous Document Cleaning – A Generative Approach to Reconstruct Strongly Corrupted Scanned Texts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(10):1950–1962, 2014.

P. Dayan, G. E. Hinton, R. Neal, and R. S. Zemel. The Helmholtz Machine. *Neural Computation*, 7:889 – 904, 1995.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38, 1977.

W. Deng, J. Hu, and J. Guo. Extended SRC: Undersampled Face Recognition via Intraclass Variant Dictionary. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1864–1870, 2012.

A. Dimitriev and M. Zhou. CARMS: Categorical-Antithetic-REINFORCE Multi-Sample Gradient Estimator. In *Advances in Neural Information Processing Systems*, 2021.

L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear Independent Components Estimation. *arXiv preprint arXiv:1410.8516*, 2014.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density Estimation using Real NVP. In *International Conference on Learning Representations*, 2017.

C. Dittmayer, J. Meinhardt, H. Radbruch, J. Radke, B. I. Heppner, F. L. Heppner, W. Stenzel, G. Holland, and M. Laue. Why misinterpretation of electron micrographs in SARS-CoV-2-infected tissue goes viral. *The Lancet*, 396(10260):e64–e65, 2020.

A. Doerr. Cryo-electron tomography. *Nature Methods*, 14:34, 2017.

C. Donahue, J. McAuley, and M. Puckette. Adversarial Audio Synthesis. In *International Conference on Learning Representations*, 2019.

W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu. Denoising Prior Driven Deep Neural Network for Image Restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

Z. Dong, A. Mnih, and G. Tucker. Coupled Gradient Estimators for Discrete Latent Variables. In *Advances in Neural Information Processing Systems*, 2021.

J. Drefs, E. Guiraud, and J. Lücke. Evolutionary Variational Optimization of Generative Models. *The Journal of Machine Learning Research*, 23(21):1–51, 2022.

J. M. Ede and R. Beanland. Partial Scanning Transmission Electron Microscopy with Deep Learning. *Scientific Reports*, 10(1):8332, 2020.

T. Ehret and P. Arias. Implementation of VBM3D and Some Variants. *Image Processing On Line*, 11:374–395, 2021.

M. Elad and M. Aharon. Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

Y. C. Eldar and G. Kutyniok. *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

Y. Ephraim and D. Malah. Speech Enhancement Using a Minimum-Mean Square Error Short-Time Spectral Amplitude Estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(6):1109–1121, 1984.

L. J. Eshelman. The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. *Foundations of Genetic Algorithms*, 1:265–283, 1991.

EVO developers. EVO - Evolutionary Variational Optimization of Generative Models. *GitHub repository*. `https://github.com/tvlearn/evo`, 2022. Last accessed: 2022-06-22.

G. Exarchakis and J. Lücke. Discrete Sparse Coding. *Neural Computation*, 29:2979–3013, 2017.

G. Exarchakis, O. Oubari, and G. Lenz. A Sampling-Based Approach for Efficient Clustering in Large Datasets. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

J. M. Fadili and J.-L. Starck. Sparse Representations and Bayesian Image Inpainting. In *International Conferences SPARS'05*, 2005.

J. Fajtl, V. Argyriou, D. Monekosso, and P. Remagnino. Latent Bernoulli Autoencoder. In *International Conference on Machine Learning*, 2020.

S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *International Conference on Machine Learning*, 2018.

L. Floridi and M. Chiriatti. GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds and Machines*, 30(4):681–694, 2020.

J. Foerster, G. Farquhar, M. Al-Shedivat, T. Rocktäschel, E. Xing, and S. Whiteson. DiCE: The Infinitely Differentiable Monte Carlo Estimator. In *International Conference on Machine Learning*, 2018.

P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

D. Forster and J. Lücke. Can clustering scale sublinearly with its clusters? A variational EM acceleration of GMMs and k-means. In *International Conference on Artificial Intelligence and Statistics*, 2018.

D. Forster, A.-S. Sheikh, and J. Lücke. Neural Simpletrons: Learning in the Limit of Few Labels with Directed Generative Networks. *Neural Computation*, 30(8):2113–2174, 2018.

A. Freno. Practical Lessons from Developing a Large-Scale Recommender System at Zalando. In *ACM Conference on Recommender Systems*, 2017.

H. R. Gelderblom, F. Kaulbars, and A. Schnartendorff. SARS-Coronavirus (Coronaviren). SARS in Vero-Zellen. Transmissions-Elektronenmikroskopie, Ultraduennschnitt. `https://www.rki.de/DE/Content/Infekt/NRZ/EM/Aufnahmen/EM_Tab_SARS.html`. Robert Koch Institut. Accessed on February 23, 2021.

T. Gerkmann and R. C. Hendriks. Unbiased MMSE-Based Noise Power Estimation With Low Complexity and Low Tracking Delay. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(4):1383–1393, 2012.

Z. Ghahramani and M. I. Jordan. Learning from incomplete data. 1995. A.I. Memo No. 1509.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

D. E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.

I. Goodfellow, A. C. Courville, and Y. Bengio. Large-Scale Feature Learning With Spike-and-Slab Sparse Coding. In *International Conference on Machine Learning*, 2012.

I. Goodfellow, A. Courville, and Y. Bengio. Scaling Up Spike-and-Slab Models for Unsupervised Feature Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1902–1914, 2013.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*. 2014.

W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted Nuclear Norm Minimization with Application to Image Denoising. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

E. Guiraud. *Scalable Unsupervised Learning for Deep Discrete Generative Models*. PhD thesis, Università degli Studi e INFN Milano (IT), 2021.

E. Guiraud, J. Drefs, and J. Lücke. Evolutionary Expectation Maximization. In *Genetic and Evolutionary Computation Conference*, 2018.

E. Guiraud, J. Drefs, and J. Lücke. Direct Evolutionary Optimization of Variational Autoencoders With Binary Latents. *arXiv preprint arXiv:2011.13704*, 2020.

M. R. Gupta and Y. Chen. Theory and Use of the EM Algorithm. *Foundations and Trends® in Signal Processing*, 4(3):223–296, 2011.

M. Haft, R. Hofman, and V. Tresp. Generative binary codes. *Formal Pattern Analysis & Applications*, 6:269–84, 2004.

J. Hajewski and S. Oliveira. An Evolutionary Approach to Variational Autoencoders. In *Computing and Communication Workshop and Conference*, 2020.

M. Haldar, M. Abdool, P. Ramanathan, T. Xu, S. Yang, H. Duan, Q. Zhang, N. Barrow-Williams, B. C. Turnbull, B. M. Collins, and T. Legrand. Applying Deep Learning to Airbnb Search. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

M. Henniges, G. Puertas, J. Bornschein, J. Eggert, and J. Lücke. Binary Sparse Coding. In *International Conference on Latent Variable Analysis and Signal Separation*, 2010.

M. Henniges, R. E. Turner, M. Sahani, J. Eggert, and J. Lücke. Efficient Occlusive Components Analysis. *The Journal of Machine Learning Research*, 15:2689–2722, 2014.

I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations*, 2017.

F. Hirschberger. vc-GMM. *Bitbucket repository.* `https://bitbucket.org/fhirschberger/clustering`, 2019. Accessed: 2019-09-17.

F. Hirschberger, D. Forster, and J. Lücke. A Variational EM Acceleration for Efficient Clustering at Very Large Scales. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9787–9801, 2022.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780, 1997.

M. D. Hoffman, D. M. Blei, C. Wang, and J. W. Paisley. Stochastic Variational Inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

P. O. Hoyer. Modeling receptive fields with non-negative sparse coding. *Neurocomputing*, 52-54:547–52, 2003.

HpBandSterCode developers. HpBandster. *GitHub repository*, 2019. URL `https://github.com/automl/HpBandSter`. Accessed: 2021-10-05.

E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. d. Carvalho. A Survey of Evolutionary Algorithms for Clustering. *IEEE Transactions on Systems, Man, and Cybernetics*, 39(2):133–155, 2009.

C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural Autoregressive Flows. In *International Conference on Machine Learning*, 2018.

M. C. Hughes and E. B. Sudderth. Fast Learning of Clusters and Topics via Sparse Posteriors. *arXiv preprint arXiv:1609.07521*, 2016.

A. Hyvärinen and E. Oja. A Fast Fixed-Point Algorithm for Independent Component Analysis. *Neural Computation*, 1997.

A. Hyvärinen and P. Pajunen. Nonlinear Independent Component Analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and Locally Consistent Image Completion. *ACM Transactions on Graphics*, 36(4):1–14, 2017.

A. Ilin and H. Valpola. On the Effect of the Form of the Posterior Approximation in Variational Learning of ICA Models. *Neural Processing Letters*, 22(2):183–204, 2005.

R. Imamura, T. Itasaka, and M. Okuda. Zero-Shot Hyperspectral Image Denoising With Separable Image Prior. In *IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

K. Ito and L. Johnson. The LJ Speech Dataset. `https://keithito.com/LJ-Speech-Dataset/`, 2021. Accessed: 2021-11-06.

A. G. Ivakhnenko and V. G. Lapa. Cybernetic predicting devices. Technical report, Purdue University School of Electrical Engineering, 1965.

L. Ivanova, A. Buch, K. Döhner, A. Pohlmann, A. Binz, U. Prank, M. Sandbaumhüter, R. Bauerfeind, and B. Sodeik. Conserved tryptophan motifs in the large tegument protein pul36 are required for efficient secondary envelopment of herpes simplex virus capsids. *Journal of Virology*, 90(11):5368–5383, 2016.

V. Jain and S. Seung. Natural Image Denoising with Convolutional Networks. In *Advances in Neural Information Processing Systems*, 2009.

E. Jang. Categorical Variational Autoencoder Using the Gumbel-Softmax Estimator. *GitHub repository*, 2016. URL `https://github.com/ericjang/gumbel-softmax`. Accessed: 2021-08-03.

E. Jang, S. Gu, and B. Poole. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017.

T. Jebara. *Machine Learning: Discriminative and Generative*. Kluwer, 2004.

Y. Jernite, Y. Halpern, and D. Sontag. Discovering Hidden Variables in Noisy-Or Networks using Quartet Tests. In *Advances in Neural Information Processing Systems*, 2013.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37:183–233, 1999.

Y. Karklin and M. S. Lewicki. Learning higher-order structures in natural images. *Network: Computation in Neural Systems*, 14(3):483–499, 2003.

Y. Karklin and M. S. Lewicki. Emergence of complex cell properties by learning to generalize in natural scenes. *Nature*, 457(7225):83–86, 2009.

Y. Kassahun, B. Yu, A. T. Tibebu, D. Stoyanov, S. Giannarou, J. H. Metzen, and E. Vander Poorten. Surgical robotics beyond enhanced dexterity instrumentation: a survey of machine learning techniques and their role in intelligent and autonomous surgical actions. *International Journal of Computer Assisted Radiology and Surgery*, 11 (4):553–568, 2016.

Z. Ke, J. Oton, K. Qu, M. Cortese, V. Zila, L. McKeane, T. Nakane, J. Zivanov, C. J. Neufeldt, B. Cerikan, J. M. Lu, J. Peukes, X. Xiong, H.-G. Kräusslich, S. H. W. Scheres, R. Bartenschlager, and J. A. G. Briggs. Structures and distributions of SARS-CoV-2 spike proteins on intact virions. *Nature*, 588(7838):498–502, 2020.

Y. Kim, S. Wiseman, A. Miller, D. Sontag, and A. Rush. Semi-Amortized Variational Autoencoders. In *International Conference on Machine Learning*, 2018.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

D. P. Kingma and P. Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems*, 2018.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved Variational Inference with Inverse Autoregressive Flow. In *Advances in Neural Information Processing Systems*, 2016.

D. P. Kingma, T. Salimans, B. Poole, and J. Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, 2021.

B. R. Kiran, D. M. Thomas, and R. Parakkal. An Overview of Deep Learning Based Methods for Unsupervised and Semi-Supervised Anomaly Detection in Videos. *Journal of Imaging*, 4(2):36, 2018.

S. Koho, E. Fazeli, J. E. Eriksson, and P. E. Hänninen. Image Quality Ranking Method for Microscopy. *Scientific Reports*, 6(1):1–15, 2016a.

S. Koho, E. Fazeli, J. E. Eriksson, and P. E. Hänninen. PyImageQualityRanking: An Image quality ranking tool for Microscopy. *GitHub repository.* `https://github.com/sakoho81/pyimagequalityranking`, 2016b. Accessed: 2021-11-29.

W. Kool, H. van Hoof, and M. Welling. Estimating Gradients for Discrete Random Variables by Sampling without Replacement. In *International Conference on Learning Representations*, 2020.

J. Kos, I. Fischer, and D. Song. Adversarial Examples for Generative Models. In *IEEE Security and Privacy Workshops*, 2018.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

A. Krull, T.-O. Buchholz, and F. Jug. Noise2Void - Learning Denoising from Single Noisy Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019a.

A. Krull, T.-O. Buchholz, and F. Jug. Noise2Void - Learning Denoising from Single Noisy Images. *GitHub repository*. `https://github.com/juglab/n2v`, 2019b. Accessed: 2020-06-10.

A. Krull, T. Vičar, M. Prakash, M. Lalit, and F. Jug. Probabilistic Noise2Void: Unsupervised Content-Aware Denoising. *Frontiers in Computer Science*, 2:5, 2020a.

A. Krull, T. Vicar, M. Prakash, M. Lalit, and F. Jug. Convallaria dataset for microscopy image denoising benchmark as used in Probabilistic Noise2Void paper, 2020b. URL `https://doi.org/10.5281/zenodo.5156913`.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic Differentiation Variational Inference. *The Journal of Machine Learning Research*, 18(14):1–45, 2017.

A. Kumar and D. Florencio. Speech Enhancement in Multiple-Noise Conditions Using Deep Neural Networks. In *Interspeech*, 2016.

M. M. Lamers, J. Beumer, J. van der Vaart, K. Knoops, J. Puschhof, T. I. Breugem, R. B. G. Ravelli, J. P. van Schayck, A. Z. Mykytyn, H. Q. Duimel, E. van Donselaar, S. Riesebosch, H. J. H. Kuijpers, D. Schipper, W. J. van de Wetering, M. de Graaf, M. Koopmans, E. Cuppen, P. J. Peters, B. L. Haagmans, and H. Clevers. SARS-CoV-2 productively infects human gut enterocytes. *Science*, 369(6499):50–54, 2020.

M. Laue, A. Kauter, T. Hoffmann, J. Michel, and A. Nitsche. Electron microscopy of SARS-CoV-2 particles - Dataset 02, 2020a. URL `https://doi.org/10.5281/zenodo.3985103`.

M. Laue, A. Kauter, T. Hoffmann, J. Michel, and A. Nitsche. Electron microscopy of SARS-CoV-2 particles - Dataset 03, 2020b. URL `https://doi.org/10.5281/zenodo.3985110`.

M. Laue, A. Kauter, T. Hoffmann, J. Michel, and A. Nitsche. Electron microscopy of SARS-CoV-2 particles - Dataset 07, 2020c. URL `https://doi.org/10.5281/zenodo.3986580`.

M. Laue, A. Kauter, T. Hoffmann, L. Möller, J. Michel, and A. Nitsche. Morphometry of SARS-CoV and SARS-CoV-2 particles in ultrathin plastic sections of infected Vero cell cultures. *Scientific Reports*, 11(1):3515, 2021.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989.

H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, 2007.

J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. Noise2Noise: Learning Image Restoration without Clean Data. In *International Conference on Machine Learning*, 2018.

M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–365, 2000.

X. Li. Patch-based Image Interpolation: Algorithms and Applications. In *International Workshop on Local and Non-Local Approximation in Image Processing*, 2008.

Y. Li, S. Liu, J. Yang, and M.-H. Yang. Generative Face Completion. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee. Enhanced Deep Residual Networks for Single Image Super-Resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.

M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *The Journal of Machine Learning Research*, 23(54):1–9, 2022.

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data.* Wiley Series in Probability and Statistics. John Wiley & Sons, 1st edition, 1987.

F. Liu, F. Xu, Y. Ding, and S. Yang. Learning Structured Sparse Representation for Single Sample Face Recognition. In *International Workshop on Biometrics and Forensics*, 2018.

R. Liu, J. Regier, N. Tripuraneni, M. Jordan, and J. Mcauliffe. Rao-Blackwellized Stochastic Gradients for Discrete Distributions. In *International Conference on Machine Learning*, 2019.

G. Lorberbom, A. Gane, T. Jaakkola, and T. Hazan. Direct Optimization through arg max for Discrete Variational Auto-Encoder. In *Advances in Neural Information Processing Systems*, 2019.

I. Loshchilov and F. Hutter. CMA-ES for Hyperparameter Optimization of Deep Neural Networks. In *International Conference on Learning Representations Workshop*, 2016.

Y. Lu and P. C. Loizou. A geometric approach to spectral subtraction. *Speech Communication*, 50(6):453–466, 2008.

J. Lucas, G. Tucker, R. B. Grosse, and M. Norouzi. Don't Blame the ELBO! A Linear VAE Perspective on Posterior Collapse. In *Advances in Neural Information Processing Systems*, 2019.

J. Lücke. Truncated Variational Expectation Maximization. *arXiv preprint, arXiv:1610.03113*, 2019.

J. Lücke and J. Eggert. Expectation Truncation And the Benefits of Preselection in Training Generative Models. *The Journal of Machine Learning Research*, 11:2855–2900, 2010.

J. Lücke and D. Forster. k-means as a variational EM approximation of Gaussian mixture models. *Pattern Recognition Letters*, 125:349–356, 2019.

J. Lücke and M. Sahani. Maximal Causes for Non-linear Component Extraction. *The Journal of Machine Learning Research*, 9:1227–1267, 2008.

J. Lücke and A.-S. Sheikh. Closed-Form EM for Sparse Coding and Its Application to Source Separation. In *International Conference on Latent Variable Analysis and Signal Separation*, 2012.

J. Lücke, Z. Dai, and G. Exarchakis. Truncated Variational Sampling for "Black Box" Optimization of Generative Models. In *International Conference on Latent Variable Analysis and Signal Separation*, 2018.

Y. Luo and N. Mesgarani. Conv-TasNet: Surpassing Ideal Time–Frequency Magnitude Masking for Speech Separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8):1256–1266, 2019.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary Deep Generative Models. In *International Conference on Machine Learning*, 2016.

D. J. C. Mackay. Local Minima, Symmetry-breaking, and Model Pruning in Variational Free Energy Minimization. *Inference Group, Cavendish Laboratory, Cambridge, UK*, 2001.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

C. J. Maddison, A. Mnih, and Y. W. Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*, 2017.

J. Mairal, M. Elad, and G. Sapiro. Sparse Representation for Color Image Restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, 2008.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-Local Sparse Models for Image Restoration. In *IEEE International Conference on Computer Vision*, 2009.

Y. Mäkinen, L. Azzari, and A. Foi. Python wrapper for BM3D denoising. `https://pypi.org/project/bm3d`, 2019. Accessed: 2020-04-18.

X. Mao, C. Shen, and Y.-B. Yang. Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections. In *Advances in Neural Information Processing Systems*, 2016.

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision*, 2001.

M. Michelashvili and L. Wolf. Speech Denoising by Accumulating Per-Frequency Modeling Fluctuations. *arXiv preprint arXiv:1904.07612*, 2020.

A. Mnih and K. Gregor. Neural Variational Inference and Learning in Belief Networks. In *International Conference on Machine Learning*, 2014.

S. Mohamed, K. A. Heller, and Z. Ghahramani. Evaluating Bayesian and L1 Approaches for Sparse Unsupervised Learning. In *International Conference on Machine Learning*, 2012.

T. Monk, C. Savin, and J. Lücke. Optimal neural inference of stimulus intensities. *Scientific Reports*, 8:10038, 2018.

K. Muhammad, A. Ullah, J. Lloret, J. D. Ser, and V. H. C. de Albuquerque. Deep Learning for Safe Autonomous Driving: Current Challenges and Future Directions. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4316–4336, 2021.

W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective.* MIT press, 2012.

J. W. Myers, K. B. Laskey, and K. A. DeJong. Learning Bayesian Networks from Incomplete Data using Evolutionary Algorithms. In *Genetic and Evolutionary Computation Conference*, 1999.

Nanographics. Real SARS-CoV-2 virion in 3D. `https://nanographics.at/projects/coronavirus-3d/`, 2021. Website publication, not yet peer-reviewed, accessed on February 12, 2021.

V. S. Narayanaswamy, J. J. Thiagarajan, and A. Spanias. On the Design of Deep Priors for Unsupervised Audio Restoration. *GitHub repository*, 2021a. URL `https://github.com/vivsivaraman/designaudiopriors`. Accessed: 2021-10-27.

V. S. Narayanaswamy, J. J. Thiagarajan, and A. Spanias. On the Design of Deep Priors for Unsupervised Audio Restoration. In *Interspeech*, 2021b.

R. Neal and G. Hinton. A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants. *Learning in Graphical Models*, pages 355–368, 1998.

NIAID. SARS-CoV-2. `https://www.flickr.com/photos/niaid/albums/72157712914621487`, 2020. Website publication, accessed on February 12, 2021.

F. M. F. Nogueira. Bayesian Optimization Package. *GitHub repository*, 2019. URL `https://github.com/fmfn/BayesianOptimization`. Accessed: 2018-09-24.

E. J. Nustede and J. Anemüller. Towards speech enhancement using a variational U-Net architecture. In *European Signal Processing Conference*, 2021.

S. Oehmcke and O. Kramer. Knowledge Sharing for Population Based Neural Network Training. In *Joint German/Austrian Conference on Artificial Intelligence*, 2018.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

B. A. Olshausen and D. M. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

M. Opper and C. Archambeau. The Variational Gaussian Approximation Revisited. *Neural Computation*, 21(3):786–792, 2009.

M. Opper and O. Winther. Expectation Consistent Approximate Inference. *The Journal of Machine Learning Research*, 6:2177–2204, 2005.

D. W. Otter, J. R. Medina, and J. K. Kalita. A Survey of the Usages of Deep Learning for Natural Language Processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):604–624, 2021.

D. M. Paiton, C. G. Frye, S. Y. Lundquist, J. D. Bowen, R. Zarcone, and B. A. Olshausen. Selectivity and robustness of sparse coding networks. *Journal of Vision*, 20(12):10–10, 2020.

V. Papyan and M. Elad. Multi-scale patch-based image restoration. *IEEE Transactions on Image Processing*, 25(1):249–261, 2016.

V. Papyan, Y. Romano, J. Sulam, and M. Elad. Convolutional Dictionary Learning via Local Processing. In *IEEE International Conference on Computer Vision*, 2017.

S. Parameswaran, C.-A. Deledalle, L. Denis, and T. Q. Nguyen. Accelerating GMM-Based Patch Priors for Image Restoration: Three Ingredients for a 100× Speed-Up. *IEEE Transactions on Image Processing*, 28(2):687–698, 2018.

Y. Park, C. Kim, and G. Kim. Variational Laplace Autoencoders. In *International Conference on Machine Learning*, 2019a.

Y. Park, C. D. Kim, and G. Kim. Variational Laplace Autoencoders. *GitHub repository*, 2019b. URL `https://github.com/yookoon/VLAE`. Accessed: 2021-08-14.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, 2019.

D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context Encoders: Feature Learning by Inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

M. B. Paulus, C. J. Maddison, and A. Krause. Rao-Blackwellizing the Straight-Through Gumbel-Softmax Gradient Estimator. In *International Conference on Learning Representations*, 2021.

F. Pernkopf and D. Bouchaffra. Genetic-based EM algorithm for learning Gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1344–1348, 2005.

A. Potapczynski, G. Loaiza-Ganem, and J. P. Cunningham. Invertible Gaussian Reparameterization: Revisiting the Gumbel-Softmax. In *Advances in Neural Information Processing Systems*, 2020.

M. Prakash, M. Lalit, P. Tomancak, A. Krull, and F. Jug. Mouse skull nuclei dataset for microscopy image denoising benchmark as used in PPN2V paper, 2019a. URL `https://doi.org/10.5281/zenodo.5156960`.

M. Prakash, M. Lalit, P. Tomancak, A. Krull, and F. Jug. Mouse actin dataset for microscopy image denoising benchmark as used in PPN2V paper, 2019b. URL `https://doi.org/10.5281/zenodo.5156937`.

M. Prakash, M. Lalit, P. Tomancak, A. Krull, and F. Jug. Fully Unsupervised Probabilistic Noise2Void. In *IEEE International Symposium on Biomedical Imaging*, 2020.

M. Prakash, A. Krull, and F. Jug. DivNoising: Diversity Denoising with Fully Convolutional Variational Autoencoders. *GitHub repository.* `https://github.com/juglab/DivNoising`, 2021a. Accessed: 2021-10-20.

M. Prakash, A. Krull, and F. Jug. Fully Unsupervised Diversity Denoising with Convolutional Variational Autoencoders. In *International Conference on Learning Representations*, 2021b.

J. Prellberg and O. Kramer. Limited Evaluation Evolutionary Optimization of Large Neural Networks. In *Joint German/Austrian Conference on Artificial Intelligence*, 2018.

ProSper developers. ProSper. *GitHub repository.* `https://github.com/ml-uol/prosper`, 2019. Last accessed: 2022-06-22.

G. Puertas, J. Bornschein, and J. Lücke. The maximal causes of natural scenes are edge filters. In *Advances in Neural Information Processing Systems*. 2010.

Y. Quan, M. Chen, T. Pang, and H. Ji. Self2Self With Dropout: Learning Self-Supervised Denoising From Single Image. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020a.

Y. Quan, M. Chen, T. Pang, and H. Ji. Self2Self With Dropout: Learning Self-Supervised Denoising From Single Image. *GitHub repository.* `https://github.com/scut-mingqinchen/self2self`, 2020b. Accessed: 2021-10-19.

R. Ranganath, S. Gerrish, and D. Blei. Black Box Variational Inference. In *International Conference on Artificial Intelligence and Statistics*, 2014.

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. In *International Conference on Machine Learning*, 2017.

D. J. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. In *International Conference on Machine Learning*, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*, 2014.

A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck. A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. In *International Conference on Machine Learning*, 2018.

J. T. Rolfe. Discrete Variational Autoencoders. In *International Conference on Learning Representations*, 2017.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

S. Roth and M. J. Black. Fields of Experts. *International Journal of Computer Vision*, 82 (2):205–229, 2009.

M. Rotmensch, Y. Halpern, A. Tlimat, S. Horng, and D. Sontag. Learning a Health Knowledge Graph from Electronic Medical Records. *Scientific Reports*, 7(1):5994, 2017.

A. Roy, A. Vaswani, A. Neelakantan, and N. Parmar. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*, 2018.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

D. E. Rumelhart, G. E. Hintont, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *arXiv preprint arXiv:1703.03864*, 2017.

R. Sarki, K. Ahmed, H. Wang, and Y. Zhang. Automatic Detection of Diabetic Eye Disease Through Deep Learning Using Fundus Images: A Survey. *IEEE Access*, 8:151133–151149, 2020.

R. Sass, E. Bergman, A. Biedenkapp, F. Hutter, and M. Lindauer. DeepCAVE: An Interactive Analysis Tool for Automated Machine Learning. In *ICML2022 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2022.

L. K. Saul and M. I. Jordan. Exploiting Tractable Substructures in Intractable Networks. *Advances in Neural Information Processing Systems*, 1995.

L. K. Saul, T. Jaakkola, and M. I. Jordan. Mean Field Theory for Sigmoid Belief Networks. *Journal of Artificial Intelligence Research*, 4(1):61–76, 1996.

U. Schmidt, Q. Gao, and S. Roth. A Generative Perspective on MRFs in Low-Level Vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient Estimation Using Stochastic Computation Graphs. In *Advances in Neural Information Processing Systems*, 2015.

R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020.

M. Seeger. Bayesian Inference and Optimal Design for the Sparse Linear Model. *The Journal of Machine Learning Research*, 9:759–813, 2008.

J. G. Serra, M. Testa, R. Molina, and A. K. Katsaggelos. Bayesian K-SVD Using Fast Variational Inference. *IEEE Transactions on Image Processing*, 26(7):3344–3359, 2017.

A.-S. Sheikh and J. Lücke. Select-and-Sample for Spike-and-Slab Sparse Coding. In *Advances in Neural Information Processing Systems*, 2016.

A.-S. Sheikh, J. A. Shelton, and J. Lücke. A Truncated EM Approach for Spike-and-Slab Sparse Coding. *The Journal of Machine Learning Research*, 15:2653–2687, 2014.

A.-S. Sheikh, N. S. Harper, J. Drefs, Y. Singer, Z. Dai, R. E. Turner, and J. Lücke. STRFs in primary auditory cortex emerge from masking-based statistics of natural sounds. *PLOS Computational Biology*, 15(1):e1006595, 2019.

J. A. Shelton, J. Bornschein, A.-S. Sheikh, P. Berkes, and J. Lücke. Select and Sample – A Model of Efficient Neural Inference and Learning. *Advances in Neural Information Processing Systems*, 2011.

J. A. Shelton, A.-S. Sheikh, J. Bornschein, P. Sterne, and J. Lücke. Nonlinear spike-and-slab sparse coding for interpretable image encoding. *PLoS ONE*, 10:e0124088, 2015.

J. A. Shelton, J. Gasthaus, Z. Dai, J. Lücke, and A. Gretton. GP-Select: Accelerating EM Using Adaptive Subspace Preselection. *Neural Computation*, 29(8):2177–2202, 2017.

S. Shivkumar, R. Lange, A. Chattoraj, and R. Haefner. A probabilistic population code based on neural samples. In *Advances in Neural Information Processing Systems*, 2018.

A. Shocher, N. Cohen, and M. Irani. “Zero-Shot” Super-Resolution using Deep Internal Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

L. N. Smith. Cyclical learning rates for training neural networks. In *IEEE Winter Conference on Applications of Computer Vision*, 2017.

J. W. Soh, S. Cho, and N. I. Cho. Meta-Transfer Learning for Zero-Shot Super-Resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

K. O. Stanley and R. Miikkulainen. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

C. Steinruecken, E. Smith, D. Janz, J. Lloyd, and Z. Ghahramani. The Automatic Statistician. In *Automated Machine Learning*. Springer, 2019.

M. Suganuma, S. Shirakawa, and T. Nagao. A Genetic Programming Approach to Designing Convolutional Neural Network Architectures. In *Genetic and Evolutionary Computation Conference*, 2017.

J. Sulam, R. Muthukumar, and R. Arora. Adversarial Robustness of Supervised Sparse Coding. *Advances in Neural Information Processing Systems*, 2020.

J. Sun and M. F. Tappen. Learning Non-Local Range Markov Random field for Image Restoration. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

Y. Tai, J. Yang, X. Liu, and C. Xu. MemNet: A Persistent Memory Network for Image Restoration. In *IEEE International Conference on Computer Vision*, 2017.

M. Tammen and S. Doclo. Deep Multi-Frame MVDR Filtering for Single-Microphone Speech Enhancement. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021.

C. Tian, Y. Xu, and W. Zuo. Image denoising using deep CNN with batch renormalization. *Neural Networks*, 121:461–473, 2020.

G. Tian, Y. Xia, Y. Zhang, and D. Feng. Hybrid Genetic and Variational Expectation-Maximization Algorithm for Gaussian-Mixture-Model-Based Brain MR Image Segmentation. *IEEE Transactions on Information Technology in Biomedicine*, 15(3):373–380, 2011.

M. K. Titsias and M. Lázaro-Gredilla. Spike and Slab Variational Inference for Multi-Task and Multiple Kernel Learning. In *Advances in Neural Information Processing Systems*, 2011.

B. Titze and C. Genoud. Volume scanning electron microscopy for imaging biological ultrastructure. *Biology of the Cell*, 108(11):307–323, 2016.

J. Tohka, E. Krestyannikov, I. D. Dinov, A. M. Graham, D. W. Shattuck, U. Ruotsalainen, and A. W. Toga. Genetic algorithms for finite mixture model based voxel classification in neuroimaging. *IEEE Transactions on Medical Imaging*, 26(5):696–711, 2007.

J. M. Tomczak and M. Welling. VAE with a VampPrior. In *International Conference on Artificial Intelligence and Statistics*, 2018.

F. Tonolini, B. S. Jensen, and R. Murray-Smith. Variational Sparse Coding. In *Conference on Uncertainty in Artificial Intelligence*, 2020.

R. E. Turner and M. Sahani. *Two problems with variational expectation maximisation for time series models*, chapter 5, pages 109–130. Cambridge University Press, 2011.

TVO developers. TVO - Truncated Variational Optimization. *GitHub repository.* `https://github.com/tvlearn/tvo`, 2022. Last accessed: 2022-06-22.

I. Ulusoy and C. M. Bishop. Generative versus discriminative methods for object recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep Image Prior. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

A. van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, 2013.

A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves. Conditional Image Generation with PixelCNN Decoders. In *Advances in Neural Information Processing Systems*, 2016.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017.

J. H. van Hateren and A. van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1394):359–366, 1998.

E. van Krieken, J. M. Tomczak, and A. T. Teije. Storchastic: A Framework for General Stochastic Automatic Differentiation. In *Advances in Neural Information Processing Systems*, 2021.

E. Vértes and M. Sahani. Flexible and accurate inference and learning for deep generative models. In *Advances in Neural Information Processing Systems*, 2018.

T. Šingliar and M. Hauskrecht. Noisy-OR Component Analysis and its Application to Link Analysis. *The Journal of Machine Learning Research*, 7:2189–2213, 2006.

M. J. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference.* Foundations and Trends in Machine Learning. Now Publisher, 2008.

M. J. Wainwright and E. P. Simoncelli. Scale Mixtures of Gaussians and the Statistics of Natural Images. In *Advances in Neural Information Processing Systems*, 2000.

D. Wang and J. Chen. Supervised Speech Separation Based on Deep Learning: An Overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1702–1726, 2018.

F. Wang, T. R. Henninen, D. Keller, and R. Erni. Noise2Atom: Unsupervised Denoising for Scanning Transmission Electron Microscopy Images. *Applied Microscopy*, 50(1):1–9, 2020.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

N. L. Westhausen and B. T. Meyer. Dual-Signal Transformation LSTM Network for Real-Time Noise Suppression. In *Interspeech*, 2020.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

D. Wrapp, N. Wang, K. S. Corbett, J. A. Goldsmith, C.-L. Hsieh, O. Abiona, B. S. Graham, and J. S. McLellan. Cryo-EM structure of the 2019-nCoV spike in the prefusion conformation. *Science*, 367(6483):1260–1263, 2020.

Y. Xia, S. Braun, C. K. A. Reddy, H. Dubey, R. Cutler, and I. Tashev. Weighted Speech Distortion Losses for Neural-Network-Based Real-Time Speech Enhancement. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020.

J. Yan, J. Schaefferkoetter, M. Conti, and D. Townsend. A method to assess image quality for low-dose PET: analysis of SNR, CNR, bias and image noise. *Cancer Imaging*, 16(1): 1–12, 2016.

C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

H. Yao, Y. Song, Y. Chen, N. Wu, J. Xu, C. Sun, J. Zhang, T. Weng, Z. Zhang, Z. Wu, L. Cheng, D. Shi, X. Lu, J. Lei, M. Crispin, Y. Shi, L. Li, and S. Li. Molecular Architecture of the SARS-CoV-2 Virus. *Cell*, 183(3):730–738, 2020.

R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. Semantic Image Inpainting with Deep Generative Models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

R. Yoshida and M. West. Bayesian Learning in Sparse Graphical Factor Models via Variational Mean-Field Annealing. *The Journal of Machine Learning Research*, 11: 1771–1798, 2010.

G. Yu, G. Sapiro, and S. Mallat. Solving Inverse Problems With Piecewise Linear Estimators: From Gaussian Mixture Models to Structured Sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2012.

K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing*, 26 (7):3142–3155, 2017.

K. Zhang, W. Zuo, and L. Zhang. FFDNet: Toward a Fast and Flexible Solution for CNN-Based Image Denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep Learning based Recommender System: A Survey and New Perspectives. *ACM Computing Surveys*, 52(1):1–38, 2019.

Z. Zhang, Y. Wang, C. Gan, J. Wu, J. B. Tenenbaum, A. Torralba, and W. T. Freeman. Deep Audio Priors Emerge from Harmonic Convolutional Networks. In *International Conference on Learning Representations*, 2020.

D. Zheng, S. H. Tan, X. Zhang, Z. Shi, K. Ma, and C. Bao. An Unsupervised Deep Learning Approach for Real-World Image Denoising. In *International Conference on Learning Representations*, 2021.

M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin. Non-Parametric Bayesian Dictionary Learning for Sparse Image Representations. In *Advances in Neural Information Processing Systems*, 2009.

M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin. Nonparametric Bayesian Dictionary Learning for Analysis of Noisy and Incomplete Images. *IEEE Transactions on Image Processing*, 21(1):130–144, 2012.

S. Zhu, G. Xu, Y. Cheng, X. Han, and Z. Wang. BDGAN: Image Blind Denoising Using Generative Adversarial Networks. In *Chinese Conference on Pattern Recognition and Computer Vision*, 2019.

L. Zimmer, M. Lindauer, and F. Hutter. Auto-Pytorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090, 2021.

D. Zoran and Y. Weiss. From Learning Models of Natural Image Patches to Whole Image Restoration. In *IEEE International Conference on Computer Vision*, 2011.