DISS. ETH NO. ?

ON TRAINING DEEP GENERATIVE MODELS WITH LATENT VARIABLES

A dissertation submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZÜRICH
(Dr. sc. ETH Zürich)

presented by

ĐORĐE MILADINOVIĆ
MSc. ETH Zürich in Computer Science

born on 09 April 1990
citizen of the Republic of Serbia

accepted on the recommendation of

Prof. Dr. Joachim M. Buhmann, examiner
Prof. Dr. Thomas Hofmann, co-examiner
Prof. Dr. Stefan Roth, co-examiner

2021

DOI: 10.3929/ethz-a-

To...

ABSTRACT

Humans understand the world through concepts. They form high-level abstractions to represent raw information in a simple way. *Conceptual thinking* is one of the key aspects of human intelligence as it enables knowledge reuse, simplifies understanding of cause-effect relationships, and empowers creativity. We hypothesize that, in order to make further progress in our quest to artificial intelligence, machine learning algorithms must be capable of (i) inferring concepts from data e.g. a shape of an object from an image of that object; and (ii) creating data from concepts e.g. synthesizing a text message that reflects one's thoughts. The main topic of this thesis are *deep generative models with latent variables (DGLs)* – a family of probabilistic models capable of performing both tasks. DGLs represent a unified framework that is generally applicable to both *learning data representations* and *learning data generative processes*. Despite remarkable progress in this area in the recent years, many methodological challenges remain that prevent successful training of these models on real-life data. The goal of this thesis is to highlight some of those challenges and propose novel algorithmic solutions to address them.

In the first part of this thesis, we study DGL-based modeling of sequential data, applied to text and videos. We discuss one of main current challenges in training sequential DGLs – learning of meaningless encoders and uninformative latent spaces. We propose a new architecture and regularization procedure to alleviate this issue. In the second part, we present a state-of-the-art variational autoencoder (VAE) for image modeling. It is based on a hierarchical network of stochastic latent variables and a newly introduced spatial dependency network that, as we demonstrate, outperforms convolutional neural networks considerably on image-generation tasks. In the third part, we integrate the novel VAE into a framework for automatic analysis of sleep patterns from brain signals. We first train a supervised architecture to classify sleep stages and then apply VAE to detect anomalies in data. The final framework is presented in the form of a web server that at the time of writing has already processed more than 10.000 sleep recordings from labs around all the world.

ZUSAMMENFASSUNG

Deutsche Zusammenfassung hier.

ACKNOWLEDGEMENTS

I would like to thank ...

CONTENTS

NOTATION

Mathematical notation

$\mathcal{X}$	Data space
x	Scalar | scalar-based data point
$\boldsymbol{x}$	Vector | vector-based data point
$\boldsymbol{X}$	Random variable over data space where $\boldsymbol{x} \sim \boldsymbol{X}$
$\vec{\boldsymbol{X}}$	Non-iid sequence (or a tuple) of random variables i.e. $(\boldsymbol{X_1}, \boldsymbol{X_2}, ..., \boldsymbol{X_T})$
W	(Weight) matrix
X^{L_k}	3-D tensor which is the output of the layer k in an image decoder
θ	Parameters of the generative distribution
ϕ	Parameters of the variational distribution
$p(\boldsymbol{x})$, $p(\boldsymbol{z})$	Probability density function
$p(\boldsymbol{x}\|\boldsymbol{z})$	Conditional probability density function
i.i.d.	Independent and identically distributed
$\mathcal{N}$	Gaussian distribution
$\mathbb{E}$	Expectation operator
$\mathcal{L}$	Evidence lower bound
$I(\boldsymbol{X}, \boldsymbol{Y})$	Mutual information between $\boldsymbol{X}$ and $\boldsymbol{Y}$
$tanh$	Hyperbolic tangent function
σ	Sigmoid function

Abbreviations

CNN	Convolutional neural network
DGL	Deep generative model with latent variables
ELBO	Evidence lower bound
GAN	Generative adversarial network

GRU	Gated recurrent unit
HMM	Hidden Markov model
KL	Kullback-Leibler divergence
LSTM	Long-short term memory
MLE	Maximum likelihood estimation
RNN	Recurrent neural network
SDN	Spatial dependency network
TF	Teacher forcing
VAE	Variational autoencoder

1

INTRODUCTION

The important thing in science is not so much to obtain new facts as to discover new ways of thinking about them.

— Sir William Brag

A human does not perceive an image as a grid of pixels but rather at a conceptual level. The objects on an image are more often discerned based on their color, their shape, their relation to other objects, and less often based on low-level details such as subtle differences in textures. The ability of humans to think on a conceptual level is sometimes referred to a as *conceptual thinking* (Kiefer and Pulvermüller, 2012), and it is arguably one of the most important hallmarks of human intelligence. In order to enable machines to think and make decisions as good as humans, they must be able to perform conceptual reasoning and think in terms of high-level abstractions. For instance, a machine needs to understand what makes a 'dog' a dog: its four legs, a distinctive muzzle, and a wagging tail. The ability to extract abstract patterns from raw sensory streams is central to the ability to extrapolate knowledge to novel situations. In this thesis, we hypothesize that the key pit stop on our road to artificial intelligence is to develop tools that can represent complex real-life sensory data in a simple way. Simple data representations abstract away most of the information that is likely to be (but not necessarily) irrelevant for the *a priori* unknown downstream tasks. Conceptual thinking is not only central to our capability to understand the world around us, but also to our ability to create. For example, when we synthesize sentences to communicate our thoughts, it is crucial that these sentences are good representatives of high-level concepts that we are trying to convey. Low-level details such as language construction are typically less important. In this thesis, we will focus on machine learning models that are able to do both, represent complex data in simple ways, and fantasize novel variations of that data based on high-level concepts. Crucially, main methods introduced here belong to the area of *unsupervised learning* i.e. no labels are assumed to accompany the data.

Concretely, this thesis explores *deep generative models with latent variables*. Let us decompose the two key concepts. *Deep generative models* is a family of machine learning models based on deep neural networks that aims to construct probabilistic models of data. In other words, deep generative models perform density estimation. By leveraging on the advances in the area of deep learning (Goodfellow et al., 2016), deep generative models are able to describe complex real-life data such as images and text, significantly outperforming classic approaches to density estimation. The fact that they are 'generative' implies that new data can be synthesized from learned probabilistic models. As opposed to discriminative models that aim to learn to predict from observations, generative models solve a more general problem; learning joint data distributions. This has a plethora of applications ranging from style transfer in images (Zhu et al., 2017) to unconditional text generation (Brown et al., 2020). One crucial advantage of generative models is that they do not require labels and can be used as a proxy towards solving other, possibly unknown downstream tasks. *Latent variables* are an old concept from probabilistic modeling (Bishop, 2006). The main idea is to supplement the set of observed variables with hidden (latent) ones. One benefit of this approach is that it increases the flexibility in distribution modeling. The second benefit is of latent variables is that they provide a natural interpretation in terms of concept learning. Namely, latent variables can be used as a mean to learn new data representations. By imposing constrains on latent space, one can extract different high-level properties of data e.g. a color of an object. Finally, deep generative models with latent variables is a class of deep generative models that contain latent variables. They are appealing as they represent a unified framework for generative modeling and representation learning. In this thesis, we will explore some challenges and propose new approaches to train and apply these models to real-life data, including text, videos, images and biological data.

1.1 THESIS CONTRIBUTIONS AND OUTLINE

In the introductory part of this thesis, we do not provide a comprehensive review of all related work and background material. We assume that the reader has basic knowledge of algebra, calculus and probability theory. The background covered in Chapter 2 is focused on work related to deep

generative models and latent variables. Deep generative models with latent variables are then studied in three different contexts.

In Chapter 3, we investigate training of sequence models. Deep generative models with latent variables for sequential data are commonly trained as autoregressive models with latent variables. They can be equivalently interpreted as variational autoencoders (VAEs) with autoregressive decoders. Previous works (Bowman et al., 2016; Chen et al., 2016) in this area have observed that the main challenge in training this models is that the autoregressive part is powerful enough to model the data so the latent variables end up being ignored during training. This is widely known as *posterior collapse* as the posterior over latent variables given observations 'collapses' to zero. The consequence of posterior collapse is that rendered encoder is useless and the latent representations are uninformative, thus the benefits of latent variables are not leveraged on. We investigate this issue from an information-theoretic point of view showing that teacher forcing that arises as a consequence of maximum likelihood estimation is the main cause of this phenomenon. We then discuss a new training procedure that regulates teacher forcing, thus proposing an advanced solution to train autoregressive models with latent variables without posterior collapse. We provide evidence that our method performs better than existing solutions on different datasets in textual and video domains.

In Chapter 4, we explore density estimation of image data. We show that image modeling can be successfully performed even without autoregressive decoders, thus alleviating all the issues with uninformative representations. We present a new state-of-the-art VAE architecture that achieves practically the same performance as autoregressive models in terms of density estimation. The key ingredients of our VAE are (i) newly introduced spatial dependency network (SDN) that models spatial dependencies and coherences at all layers of a deep generative neural network that represents VAE decoder; and (ii) a hierarchical (ladder) architecture that enables smooth convergence even without utilizing teacher forcing. On multiple image datasets, we show that our VAE can generate perceptually appealing high-resolution images, achieves state-of-the-art results in density estimation, and can learn latent representations of high-quality in terms of structure and informativeness.

In Chapter 5, we focus on a particular application in the domain of sleep biology. We present SPINDLE, a platform for automatic analysis of sleep patterns from EEG/EMG brain recordings. The motivation is to replace

laborious and time-consuming process of sleep labeling that is a common practice in sleep research. We utilize both supervised approach to classify individual sleep states, and the newly developed VAE to design a robust tool for identifying anomalous patterns. Furthermore we present a dataset from four different sleep labs that contains almost 1M labels. The entire framework is provided as a web service, and at the moment of writing has already processed more than 10.000 sleep recordings.

Finally, we provide concluding remarks on this thesis in Chapter 6. We summarize the limitations of our study and provide possible directions for future work.

1.2 RELATED PUBLICATIONS AND PREPRINTS

Included in the thesis

- João Carvalho, João A Santinha, Đorđe Miladinović, and Joachim M Buhmann. Spatially Dependent U-Nets: Highly Accurate Architectures for medical imaging segmentation. arXiv preprint arXiv:2103.11713, 2021.
- Đorđe Miladinović, Aleksandar Stanic , Stefan Bauer, Jurgen Schmidhuber, and Joachim M. Buhmann. Spatial dependency networks: Neural layers for improved generative image modeling. In 9th International Conference on Learning Representations (ICLR 2021), May 2021.
- Đorđe Miladinović, Muhammad Waleed Gondal, Bernhard Schölkopf, Joachim M Buhmann, and Stefan Bauer. Disentangled state space representations. arXiv preprint arXiv:1906.03255, 2019a. Also in DeepGenStruct workshop In 7th International Conference on Learning Representations (ICLR 2019)
- Đorđe Miladinović, Christine Muheim, Stefan Bauer, Andrea Spinnler, Daniela Noain, Mojtaba Bandarabadi, Benjamin Gallusser, Gabriel Krummenacher, Christian Baumann, Antoine Adamantidis, et al. Spindle: End-to-end learning from eeg/emg to extrapolate animal sleep scoring across experimental settings, labs and species. PLoS computational biology, 15(4):e1006968, 2019.

Not included in the thesis

- Muhammad Waleed Gondal, Manuel Wuthrich, Đorđe Miladinović, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new dis- entanglement dataset. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.
- Raphael Suter, Đorđe Miladinović, Bernhard Schölkopf, and Stefan Bauer. Robustly disentangled causal mechanisms: Validating deep representa- tions for interventional robustness. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 6056–6065. PMLR, 09–15 Jun 2019.
- Patrick Schwab, Đorđe Miladinović, and Walter Karlen. Granger-causal Attentive Mixtures of Experts: Learning Important Features With Neural Networks. In AAAI Conference on Artificial Intelligence, 2019. 117118 Bibliography
- Stefan Bauer, Nico S Gorbach, Đorđe Miladinović, and Joachim M Buhmann. Efficient and flexible inference for stochastic systems. Advances in Neural Information Processing Systems 30, 10:6989–6999, 2018.

2

DEEP GENERATIVE MODELS & LATENT VARIABLES

If something is in me which can be called religious then it is the unbounded admiration for the structure of the world so far as our science can reveal it.
— Albert Einstein

The aim of this chapter is to provide the necessary background on *deep generative models* and *latent variables*. We begin by introducing *density estimation* as the original motivation to study these concepts. The goal is to understand when and why density estimation is necessary, and to provide an overview of the existing approaches. We then study models with latent variables from two different perspectives: *(i)* as a mean to increase the flexibility of parametric approaches to density estimation; and *(ii)* for the purpose of *representation learning*. Deep generative models are introduced as a modern deep neural network approach to density estimation. We explain the main ideas and then cover relevant literature including state-of-the-art methods. We conclude the chapter by summarizing the relevance of *deep generative models with latent variables* – the main topic of this thesis.

2.1 DENSITY ESTIMATION

Probability density function is one of the most fundamental concepts in probability and statistics. It is defined for any multidimensional random variable $\boldsymbol{X}$ as a descriptor of its probability distribution. The probability density function of $\boldsymbol{X}$ is given as

$$P(a \leq \boldsymbol{X} \leq b) = \int_a^b p(\boldsymbol{x}) d\boldsymbol{x} \text{ for all } a \leq b \tag{2.1}$$

Density estimation (Rosenblatt, 1956; Parzen, 1962; Silverman, 1986) is the process of constructing an estimate of the probability distribution of data

given a finite set of training samples drawn from that distribution[1]. Let us assume a parametric density model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ where $\boldsymbol{\theta}$ are the parameters of the model and $\boldsymbol{x} = (x_1, x_2, ..., x_D)^T$ is a D-dimensional column vector. The task in density estimation is to approximate the 'true' probability density function $p(\boldsymbol{x})$ from the training set $\mathcal{D}$ that consists of N samples $\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$ drawn from $p(\boldsymbol{x})$. This can be informally written as

$$\text{using } \mathcal{D} \text{ learn } \boldsymbol{\theta} \text{ such that } p_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx p(\boldsymbol{x}) \tag{2.2}$$

Because in reality we have access only to the empirical distribution $\hat{p}(\boldsymbol{x})$ defined by $\mathcal{D}$ but not $p(\boldsymbol{x})$, the care must be taken to prevent overfitting of $\boldsymbol{\theta}$ with respect to $\hat{p}(\boldsymbol{x})$. The quality of the model distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ with respect to $\hat{p}(\boldsymbol{x})$ can be evaluated using a distance or a divergence measure that estimates the discrepancy between the two distributions. There are many ways to learn $\boldsymbol{\theta}$ from the training set $\mathcal{D}$. One popular learning approach is based on *maximum likelihood estimation (MLE)* (Fisher, 1922; Bishop, 2006; Goodfellow et al., 2016) in which case the parameter learning can be formulated as

$$\boldsymbol{\theta} \leftarrow \arg\max_{\boldsymbol{\theta}} \left[\frac{1}{N} \sum_{i=1}^{N} \log\, p_{\boldsymbol{\theta}}(\vec{x}^{(i)}) \right] \tag{2.3}$$

It can be shown that in MLE, the discrepancy between the model distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and the empirical distribution $\hat{p}(\boldsymbol{x})$ is measured using *Kullback–Leibler (KL) divergence* (Cover and Thomas, 2006). Hence MLE learning can be equivalently formulated as

$$\boldsymbol{\theta} \leftarrow \arg\max_{\boldsymbol{\theta}}\ \mathrm{KL}(p_{\boldsymbol{\theta}}(\boldsymbol{x}) || \hat{p}(\boldsymbol{x})) \tag{2.4}$$

Existing approaches to density estimation are most commonly categorized based on how the model distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is specified. Prior to the deep learning revolution, there were two dominant approaches to density estimation; the parametric and the nonparametric one. *Deep generative models* (Goodfellow et al., 2016) represent a relatively new family of methods based on deep neural networks that has enabled unprecedented results in density estimation in the context of complex real-life data such as images (Brock et al., 2018), text (Brown et al., 2020) and speech (Oord et al., 2018). Before we cover the basics of all three, let us discuss the motivation for, and the situations in which density estimation is necessary.

1 To simplify the further discussion without limiting the scope of this thesis, we will assume that all samples are independent and identically distributed (i.i.d.).

2.1.1 *Why density estimation?*

In its default formulation, density estimation is one of the most difficult problems in machine learning as it requires probabilistic models to describe *all* the characteristics of data, the ones that are relevant to some downstream task but also the ones that are not. This is in contrast to supervised learning, where a classification or a regression algorithm discards most of the information in the input keeping only the task-relevant details. For example, a neural network trained to differentiate images of cats and dogs may discard the background color. From this perspective, density estimation can be categorized into the field of unsupervised learning and is useful in the scenarios in which labels are not provided or the task is not known in advance. On top of that, in many real-world problems, such as data synthesis or compression, the concept of 'labels' is *de facto* non-existent thus mitigating density estimation is impossible. Let us cover some common use cases of density estimation:

- **Data synthesis:** In many real-life scenarios we are required to generate new and realistic samples that look like samples that we already have. For example, if we are building a chat bot, the task may involve synthesizing textual replies that both convey certain information that a customer inquired about and that appear human-like.
- **Data manipulation:** Some tasks require data to be manipulated in a semantically meaningful way. For example, knowledge of the underlying distribution is beneficial when performing perceptually appealing image interpolation. This is because one can interpolate by going only through high-density regions. A näive approach such as interpolating in the pixel space results in generating intermediate samples which do not look plausible to a human eye.
- **Missing value imputation:** In some cases, we are given only a partial observation with certain parts of it missing e.g. an incomplete image in the problem of image completion. One way to approach this problem is to chose missing pixels such that the joint distribution of the whole image is maximized with respect to the learned probability distribution.
- **Anomaly detection:** The task in anomaly (or outlier/novelty) detection is to identify test samples that were likely not drawn from the

same distribution as the training data. For example, if the task is to detect the abnormalities in heart activity given the abundance of data on the activity of a healthy hearth. In terms of density estimation, this would mean fitting a probability model on training data and then using it to identifying test samples that are improbable.

- **Denoising:** In denoising, the task is to recover the original observation $\boldsymbol{x}$ from its noisy version $\widetilde{\boldsymbol{x}}$ that is given. The noise is typically introduced due to the imperfection of the recording setup. If the true underlying distribution is known (or approximated), one elegant solution is to find a close neighbor of $\widetilde{\boldsymbol{x}}$ that is most likely under that distribution.

- **Data compression:** The task of data compression is to encode data in a maximally compressed way in terms of number of bits used i.e. bit-rate. If the probability distribution of data is known, optimal losless compression schemes can be created such as Huffman coding (Huffman, 1952).

2.1.2 *Parametric approach*

In the parametric approach (Silverman, 1986; Bishop, 2006), one specifies a probabilistic model with a relatively low number of adaptive parameters. These parameters define the family of probability of distributions that the model can select from. Given a learning algorithm and a training dataset, the parameters are adapted such that the assumed probabilistic model approximates the empirical distribution of the training data as closely as possible, as explained at the beginning of this chapter. For example, one of the simplest and yet most widely used parametric density models is the *Gaussian distribution*

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{\boldsymbol{\Sigma}^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{2.5}$$

Here, $\boldsymbol{\mu}$ is the vector of parameters that represents learnable means and $\boldsymbol{\Sigma}$ are the parameters that represent learnable covariances for each of the D dimensions. The challenge with parametric approaches is to ensure that the chosen density model is a good fit for the true distribution that generates the data. This is often difficult in practice as the densities of the real-life data are usually of complex forms, exhibiting highly nonlinear

relations between dimensions, whereas most common parametric models are unimodal.

2.1.3 *Nonparametric approach*

Hand-crafting parametric probabilistic models to be a good fit to training data while allowing for efficient parameter learning is a very challenging task. To mitigate this difficulty, an alternative approach is to estimate the density in a nonparametric fashion (Izenman, 1991; Bishop, 2006). Nonparametric approaches make fewer assumptions about the form of the underlying data distribution and thus may be more suitable for complex multimodal densities. The simplest method from this class is the *histogram approach*. In this method, each dimension x_i of the vector $\boldsymbol{x}$ is split into intervals of a prespecified length Δ_i. The idea is then to count the number of training samples that fall into each 'bin' that was formed by slicing $\boldsymbol{x}$. Because the final probability density must be normalized, the counts need to be divided by the total number of training samples and by the volume of the bins. This can be represented as

$$p(\boldsymbol{x}) = \frac{n(\boldsymbol{x})}{N \prod_{i=1}^{D} \Delta_i} \tag{2.6}$$

where $n(\boldsymbol{x})$ is the number of training samples that fall into to the same bin as the vector $\boldsymbol{x}$. The main issue is that the amount of training data required grows exponentially with the number of dimensions D. This is known as "the curse of dimensionality" (Bishop, 2006). Even more complex methods from this class, for example kernel density estimator and nearest-neighbor approach, also suffer from the same issue and are consequently difficult to apply to very high-dimensional data such as images.

2.1.4 *Deep generative modeling approach*

At the moment of writing, the most popular approach to density estimation is based on deep learning/neural networks, at least when it comes to modeling highly complex data such as images and text. The deep generative modeling approach is parametric in the sense that a parametric density model of the data is assumed and the parameters of that model are adapted through the process of learning. The main difference is that neural

networks contain very large number of parameters and can thus approximate functions of *de facto* arbitrary complexity (Goodfellow et al., 2016). Hence the family of distributions that they can represent is considerably richer. Despite such a large number of free parameters, neural networks still offer efficient learning through backpropagation and numerous ways to regularize learning and introduce prior knowledge on modeled data. Modern neural networks contain billions of parameters, so from that perspective one can also think of them as nonparametric. As this approach is central to our work, we discuss it in more detail in Section 2.3.

2.2 LATENT VARIABLES

The idea behind latent variables is one of the most powerful ones in probabilistic modeling. The trick is to augment the original set of observed variables with an additional set of variables that are considered unobserved i.e. latent, hidden. Latent variable approach is appealing for multiple reasons. Firstly, as discussed in Section 2.2.1, it enables constructing flexible parametric probabilistic models. Secondly, as discussed in Section 2.2.2, if additional structural constrains are imposed to latent variables e.g. dimensionality much lower than the one of the original observation space, then the resulting probabilistic model can also be used for the purposes other than density estimation – dimensionality reduction in our example. In this thesis, to refer to all the problems that latent variables can be used for beyond density estimation, we use the umbrella term *representation learning* (Bengio et al., 2013). Finally, latent variable model has a very intuitive interpretation about how data is generated: We assume that any observation is produced from a two-step generative process. First, a random latent variable $\boldsymbol{z}$ is sampled from a prior distribution $p_0(\boldsymbol{z})$. Here, $\boldsymbol{z}$ models the abstract space of concepts such as the shape and the color of objects. In the second step, the observation $\boldsymbol{x}$ is sampled from the conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$. Hence $\boldsymbol{x}$ can be thought of as a high-dimensional projection e.g. an image of an object, projected from a space of concepts that indicate the shape and the color of that object. Latent variable approach to density modeling can be described as

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})p_0(\boldsymbol{z})d\boldsymbol{z} \tag{2.7}$$

The corresponding graphical model is given in Figure 2.1.

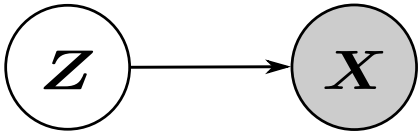


FIGURE 2.1: **Directed acyclic graph of a latent variable model.**

Let us now explore the core motivations for introducing latent variable models in more detail.

2.2.1 *Improving the flexibility of density models*

As noted in Section 2.1, one issue with the vanilla parametric approaches is the simplicity of the hand-crafted probabilistic models which are insufficiently flexible. Latent variable approach allows for constructing more flexible probabilistic models. In this setting, instead of operating on probability distribution over observed variables, one operates on the joint probability distribution over observed and latent variables. The corresponding distribution over observed variables can be obtained via marginalization. One well-known example method that belongs to the class of latent variable models is *Gaussian mixture model (GMM)*. In GMM, there exist a single latent variable that is discrete, taking values $z \in \{1, 2, ..., K\}$ where K is the number of categories. So called mixture distribution can then be formed as a linear superposition of Gaussian distributions

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}) \tag{2.8}$$

where π_k is the probability of k-th mixture component. Another example of latent variable models is factor analysis which is based on continuous latent variables (Bishop, 2006). As we discuss further below in Section 2.3, latent variables are also the basis for some of the deep generative model approaches.

2.2.2 *Representation learning*

The second key benefit of latent variable modeling is the possibility of obtaining 'useful' data representations, where the usefulness is determined based on the intended downstream task. The main goal of representation learning is to discover the high-level structure from raw observations in an unsupervised way e.g. inferring the color of an object from an image of that object. We can best understand this by analyzing common use cases:

- **Dimensionality reduction:** The goal of dimensionality reduction is to represent the high-dimensional observations in a low-dimensional manifold. If the number of dimensions of a latent variable $\boldsymbol{z}$ is low, then it can be interpreted as a low-dimensional representation of the high-dimensional observation $\boldsymbol{x}$.

- **Clustering:** In clustering, the goal is to split a set of observations into a relatively low number of groups such that the observations from the same groups are similar to each other, according to a prespecified similarity metric. By modeling $\boldsymbol{z}$ as a discrete random variable that has a number of possible values corresponding to the number of clusters, we can performing clustering via latent variable assignment. Gaussian mixture models employ this strategy.

- **Semi-supervised learning:** In the supervised setting e.g. classification, the task is to learn mapping from observations to labels. In many cases the data is ubiquitous but the labels are scarce. One common strategy which allows leveraging on both labeled and unlabeled data is to perform pre-training i.e. representation learning on unlabeled data, and then use the pre-trained representations and labeled data to construct the classifier.

- **Disentangled representation learning:** A popular assumption in the field of representation learning, which can be turned into a powerful prior, is that high-dimensional data is generated from a few independent factors of variation (Schmidhuber, 1992; Bengio et al., 2013). This idea has led to a surge of approaches that try to 'disentangle' latent dimensions by enforcing a factorized prior in the latent space. The hypothesis is that disentangled representations are suitable when downstream tasks are not known *apriori*.

- **Controlled data synthesis:** When synthesizing data, it is often desirable to have a fine-grain control over the generated output. For example, when generating a human face it would be beneficial to enable easy manipulation of different semantic aspects of a face such as eyes, facial hair etc. One way to achieve this is to employ a deep generative model with latent variables and then enforce that different latent dimensions correspond to different semantic aspects, as done in disentangled representation learning.

2.3 DEEP GENERATIVE MODELS

In Section 2.1.4, we briefly introduced deep generative models as an approach to density estimation that is based on deep neural networks and is applicable to modeling of complex data such as images and text. Even though there exist a plethora of different deep generative modeling approaches (Goodfellow et al., 2016), in the following we specifically focus on the the ones that are of high relevance for providing a context for the subsequent chapters. Nevertheless, for completeness, the remaining ones are in less detail covered at the end of the section.

2.3.1 *Autoregressive models*

Autoregressive models are based on the following factorization of the model distribution

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \prod_{i=1}^{D} p_{\boldsymbol{\theta}}(x_i|\boldsymbol{x}_{<i}) \tag{2.9}$$

where $\boldsymbol{x}_{<i} = (x_{i-1}, x_{i-2}, ..., x_1)$. This equality is a consequence of the well-known *chain rule* from the probability theory. The ordering of the dimensions is arbitrary and in practice is chosen depending on the type of data at hand e.g. in sentence modeling, typically it corresponds to the natural ordering of words in a sentence. The term *autoregressive* originates from time-series forecasting where one designs a model that predicts the next-step value of the time-series given all the previous steps i.e. the history of the sequence. Represented as a directed graphical model, the autoregressive model is given in Figure 2.2.

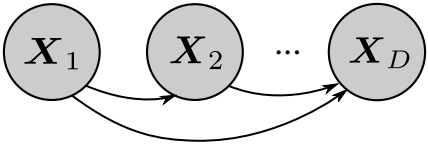


FIGURE 2.2: **Directed acyclic graph of an autoregressive model.**

Autoregressive models *per se* do not make any additional assumptions about data. However, they can be efficiently regularized since in practice we assume that conditional distributions have the same functional form. In deep generative modeling, these conditionals are described with neural networks. For example, popular architectures for text and speech modeling include recurrent neural networks based on gated units (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) and transformers (Vaswani et al., 2017). For images, most commonly convolutional neural networks are used (Van den Oord et al., 2016; Salimans et al., 2017). Another appealing property of autoregressive models is the explicitness in density estimation i.e. it is possible to directly compute $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ in contrast to deep generative models that model density only implicitly – see generative adversarial networks in Section 2.3.3. Consequently, the maximum likelihood learning is simple to implement (recall Equation 2.3) and the likelihood of an arbitrary observation can be computed in the test time.

In practice, autoregressive models are known to be powerful density estimators. Both for images and text autoregressive models achieve state-of-the-art likelihoods at the moment of writing. There are however certain disadvantages too. Firstly, the sampling time scales linearly with the number of dimensions D. This makes autoregressive models unsuitable for tasks in which fast sampling is required at test time. During training however, it is possible to leverage on *teacher forcing* (Williams and Zipser, 1989). For some architectures such as transformers, teacher forcing enables parallelized and thus efficient training. Secondly, the perceptual quality of samples in certain domains, such as images, is poorer in comparison to other models such as variational autoencoders and generative adversarial networks. This is not entirely surprising as there is evidence that increased log-likelihood does not necessarily imply increased perceptual quality (Theis et al., 2015). Thirdly, it is not always clear and intuitive how to define ordering of dimensions when performing factorization e.g. when generating images pixel-by-pixel. Finally, autoregressive models *per se* do

not have latent variables so it is not possible to perform representation learning-related tasks mentioned in Section 2.2.2. However, as we discuss in detail in Chapter 3, this thesis offers new solutions for training autoregressive models with latent variables.

2.3.2 *Variational autoencoders*

Variational autoencoders (VAEs) (Kingma and Welling, 2013; Rezende et al., 2014) belong to the class of deep generative models based on latent variables – a class which is in literature sometimes referred to as as deep latent-variable models (DLVMs) (Kingma and Welling, 2019). VAE is graphically represented in Figure 2.3. In DLVMs in general, it is difficult to estimate the

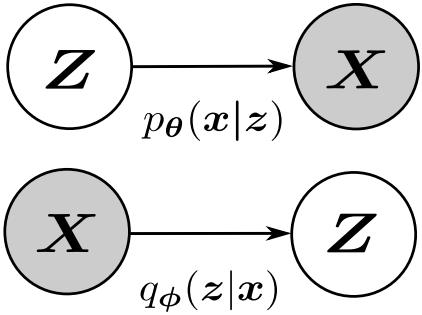


FIGURE 2.3: **Directed acyclic graph of a variational autoencoder:** *(above)* generative model; *(below)* recognition model;

likelihood of observations and the posterior over latent variables (Kingma and Welling, 2019). However, the framework of VAEs provides a principled solution that allows for computationally efficient learning based on stochastic gradient descent (SGD). There are several important ideas that enable this. Firstly, VAEs jointly learn deep generative models that map latent variables into observations and the corresponding inference models that perform the inverse procedure. This mechanism is somewhat inspired by the sleep-wake algorithm used in the Helmholtz Machine (Dayan et al., 1995) that is possibly the first method to employ an explicit recognition network (encoder) in the context of generative modeling. The main advantage of VAEs is the efficiency of learning which is based on a single objective

instead of multiple ones present in the Helmholtz Machine. The recognition network in VAEs is an approximation of the true posterior of latent variables given an observation, typically described as

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \approx p(\boldsymbol{z}|\boldsymbol{x}) \tag{2.10}$$

where $\boldsymbol{\phi}$ is a set of parameters defining the family of distributions $q_{\boldsymbol{\phi}}$ that we search through in order to find the one that best approximates the 'true posterior' $p(\boldsymbol{z}|\boldsymbol{x})$. Because the idea of specifying and optimizing an approximate posterior originates from variational inference (Bishop, 2006; Blei et al., 2017), the parameters $\boldsymbol{\phi}$ are referred to as the *variational parameters* (Kingma and Welling, 2019). In contrast to classic variational inference (Bishop, 2006; Blei et al., 2017), there exist no per-observation optimization loop, but a neural network that models the observation-specific inference procedure. For this reason VAEs are said to perform *amortized variational inference* (Gershman and Goodman, 2014). This contributes to the efficiency both during training and inference. Perhaps the main benefit of introducing the approximate posterior is that it enables VAEs to approximate the likelihood of data, hence opening the door to maximum likelihood training. In contrast to autoregressive models however, VAEs cannot compute the exact likelihood of data but its evidence lower bound (ELBO). In practice, ELBO is often quite close to the exact likelihood. Following (Kingma and Welling, 2019), ELBO can be derived as

$$\begin{aligned}
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\right] && (2.11)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right] && (2.12)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right] && (2.13)\\
&= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right]}_{=\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})=ELBO} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right]}_{\mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))} && (2.14)
\end{aligned}$$

where KL denotes KL-divergence. If the KL term is zero, then the the marginal distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is equal to the ELBO term $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$. In other words, if the approximate posterior is equal to the true one, ELBO is not an approximation of data likelihood but the exact value of it. ELBO can be further decomposed as

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] \tag{2.15}$$

One positive aspect about ELBO-based approach is that it allows for joint optimization of $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. The problematic part is that the gradient $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$ is generally intractable. The unbiased estimator of $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$ can be easily computed, however, computing $\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$ is more challenging as it requires differentiation through the sampling procedure

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = \nabla_{\boldsymbol{\phi}}\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] \tag{2.16}$$

This brings us to the second key idea that VAEs employ. *The reparametrization trick* enables efficient computation of the gradient from Equation 2.16 (Kingma and Welling, 2013; Rezende et al., 2014). If the latent variables are continuous[2] and if the encoder and decoder are differentiable functions e.g. neural networks, then ELBO can be easily differentiated with respect to both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ by applying a change of variables. The trick is to 'externalize randomness'. A random latent variable $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ can be represented as

$$\boldsymbol{z} = g(\boldsymbol{\epsilon},\boldsymbol{\phi},\boldsymbol{x}) \text{ where } p(\boldsymbol{\epsilon}) = p(\boldsymbol{\epsilon}|\boldsymbol{\phi},\boldsymbol{x}) \tag{2.17}$$

The gradient with respect to $\boldsymbol{\phi}$ from Equation 2.16 can then be computed as

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = \nabla_{\boldsymbol{\phi}}\mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right] \tag{2.18}$$

As a consequence of the change of variables, a simple one-sample Monte Carlo procedure can be used to approximate the gradient of ELBO with respect to $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ as follows

$$\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{2.19}$$

$$\boldsymbol{z} = g(\boldsymbol{\epsilon},\boldsymbol{\phi},\boldsymbol{x}) \tag{2.20}$$

$$\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}) - \nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \tag{2.21}$$

In addition, the following decomposition of the joint distribution is normally performed

$$p_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p_0(\boldsymbol{z}) \tag{2.22}$$

$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is a neural network with domain-specific observation model e.g. isotropic Gaussian in case of images and a cross-entropy in case of words.

2 There are tricks to deal with the discrete latent variables too (Jang et al., 2016; Maddison et al., 2016; Van Den Oord et al., 2017).

The probabilistic models of $p_0(\boldsymbol{z})$, $p(\boldsymbol{\epsilon})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ are in most of the cases chosen to be isotropic Gaussian distributions. Albeit very simple, in practice it turns out that they allow for modeling complex distributions when paired with powerful neural networks that model the conditionals $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. Furthermore, additional tricks can be employed to allow for modeling of more flexible posteriors or priors without disrupting efficiency of the training (Kingma et al., 2016; Chen et al., 2016).

One disadvantage of VAEs in comparison to autoregressive models is that they do not compute the exact data likelihood for a given model. However, there exist a computational procedure to 'tighten' the lower bound which is based on importance sampling (Burda et al., 2015). This is in practice only applicable during inference as the training time would be significantly affected. The main advantage of VAEs is the existence of latent variables which makes VAEs applicable to a plethora of representation learning-related tasks that were discussed in detail in Section 2.2.2. Moreover, when a non-autoregressive observation model is chosen to describe $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, the sampling time is considerably faster. While autoregressive models are still state-of-the-art in density modeling across multiple data modalities, there has been recent success in closing the gap between autoregressive models and VAEs (Vahdat and Kautz, 2020; Child, 2020; Miladinović et al., 2021). Our contribution to this trend is discussed in Chapter 4.

2.3.3 *Generative adversarial networks*

Generative adversarial networks (GANs) are a class of deep generative models that is very different from the ones we discussed previously, in many different ways. First of all, GANs are not explicit density estimators in the sense that they do not define a density function $p_\theta(\boldsymbol{x})$ that approximates the true distribution $p_\theta(\boldsymbol{x})$. For this reason they are generally not suitable for most of the density estimation-related tasks listed in Section 2.1.1 e.g. one cannot perform anomaly detection using GANs. GANs do not rely on maximum likelihood estimation procedure from Equation 2.3. Instead, training GANs implies finding the Nash equilibrium of the corresponding game (described further below), which turns out to be a more difficult problem than simply optimizing maximum likelihood-based objective function. GANs can in some sense be regarded as a latent variable model since the *generator* adheres to the same graphical models as genera-

tive part of VAEs, as shown in Figure 2.4. However, since the recognition network is missing, the representation learning tasks from Section 2.2.2 cannot be tackled using GANs in the default setting. Instead, GANs em-

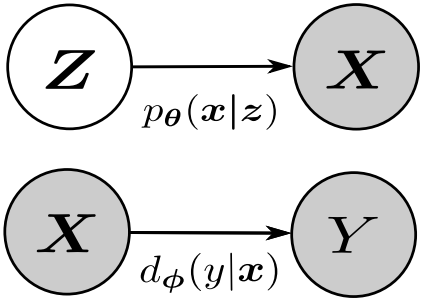


FIGURE 2.4: **Directed acyclic graph of a generative adversarial network:** *(above)* generator model; *(below)* discriminator model;

ploy another network called discriminator that attempts to discriminate between samples drawn from the original distribution and the samples drawn from the generative model. This game is visualized in Figure 2.5. To formulate the minimax game between the discriminator and the generator, we can first define a cost function that the discriminator tries to minimize and the generator tries to maximize.

$$\mathcal{J}_{\theta,\phi}(x) = -\frac{1}{2}\mathbb{E}_{x\sim p(x)}\log d_{\phi}(y=1|x) - \frac{1}{2}\mathbb{E}_{z\sim p_0(z)}\log d_{\phi}(y=0|x)p_{\theta}(x|z) \tag{2.23}$$

The minimax game is then given as

$$\theta \leftarrow \arg\min_{\theta}\arg\max_{\phi} -\mathcal{J}_{\theta,\phi}(x_i) \tag{2.24}$$

The key advantage of GANs in comparison to other deep generative models it that they are generally believed to produce the most realistic samples when it comes to image modeling. For this reason, we have witnessed the adoption of GANs across many areas of computer vision, including image-to-image translation (Zhu et al., 2017), super-resolution (Ledig et al., 2017) and realistic high-resolution image generation (Brock et al., 2018). One of the contributions of our work is to show that, when trained properly, VAEs

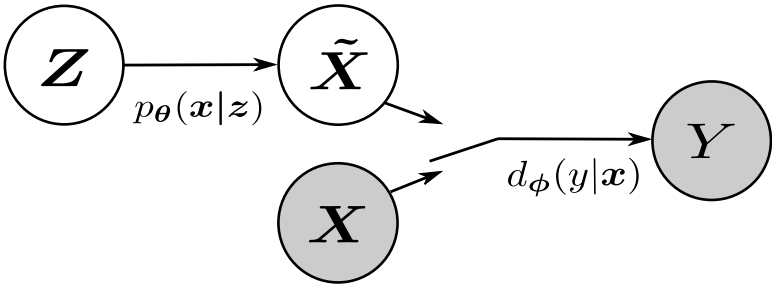


FIGURE 2.5: **GAN game:** Generator tries to 'fool' the discriminator by generating realistic samples from a randomly sampled latent variable ($\boldsymbol{z}$). Discriminator takes either a sample generated by the model ($\tilde{\boldsymbol{x}}$) or the one taken from the training dataset ($\boldsymbol{x}$), and then tries to predict from which of the two sources the sample comes from. Based on the label of the actual source (y), the learning update is performed.

can also generate perceptually appealing samples that come close to the ones generated by GANs. More about this we discuss in Chapter 4. Apart from the above-mentioned ones, other disadvantages of GANs include: *(i)* training instability – the generator and discriminator loss often continue to oscillate without converging to a clear stopping point. Due to the lack of a robust stopping criteria, it is difficult to know when exactly the GAN has finished training; *(ii)* potential for *mode collapse* – GAN can often get stuck producing one of a few types of samples over and over again; and *(iii)* difficulty of evaluation – since the density model is not explicit, it is difficult to evaluate to perform model selection and compare GANs to other approaches. Due to the exceptional popularity of GANs however, many tricks have been developed in the last years to tackle these issues.

2.3.4 *Normalizing flows*

The framework of *normalizing flows* was first introduced by (Tabak et al., 2010; Tabak and Turner, 2013) but was later popularized by (Rezende and Mohamed, 2015) for the purpose of improving variational inference, and by (Dinh et al., 2014) in the context of density estimation. The idea is to create a sequence of invertible transformations that convert a very

simple distribution like isotropic Gaussian that models latent space into a complex one that models the observation space e.g. images. The output of each transformation is a valid probability distribution. Because the transformations are invertible, the mapping between latent and observation space is two-directional. The relationship between the latent variable $\boldsymbol{z}$ and the observation $\boldsymbol{x}$ with respect to sequence of transformations $f = f_1 \circ f_2 \circ f_3 \circ ... \circ f_K$ can be represented as (Kingma and Dhariwal, 2018)

$$\boldsymbol{z} \overset{f_1}{\longleftrightarrow} \boldsymbol{h_1} \overset{f_2}{\longleftrightarrow} \boldsymbol{h_2} \overset{f_3}{\longleftrightarrow} ... \overset{f_K}{\longleftrightarrow} \boldsymbol{x} \tag{2.25}$$

where $\boldsymbol{h_1}$, $\boldsymbol{h_2}$, etc. can be regarded as intermediate latent states. The transformations that make a normalizing flow can be seen as applying the rule of *change of variables*. The probability density function of the observation $\boldsymbol{x}$ can be computed as

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log p_{\boldsymbol{\theta}}(\boldsymbol{z}) + \sum_{k=1}^{K} \log |det(\frac{d\boldsymbol{h_k}}{d\boldsymbol{h_{k-1}}})| \tag{2.26}$$

The key challenge is to design transformations that are invertible and have an easy-to-compute Jacobian while having enough capacity to model complex mappings (Dinh et al., 2014; Kingma and Dhariwal, 2018; Kingma et al., 2016). Core advantages of the normalizing flow framework include: *(i)* exact log-likelihood computation, like in autoregressive models; *(ii)* efficient sampling and inference, like in VAEs; *(iii)* latent variable modeling that enables utilization of normalizing flows in the context of representation learning. The main disadvantage is the difficulty in constructing flows that are both powerful and tractable. Arguably for this reason, normalizing flows are still lagging behind other generative models in terms of quantitative density estimation performance and the quality of generated images. Moreover, the latent representation $\boldsymbol{z}$ as well as all intermediate representations are constrained to be of the same size as the observation $\boldsymbol{x}$. This makes normalizing flows unsuitable for the tasks such as dimensionality reduction.

2.3.5 *Other deep generative models*

Many different deep generative models have emerged during the past several decades and it would be a rather challenging task to cover all of them in great detail. In order to paint a big picture however, here we mention

the ones that (in our opinion) appear to be most popular at the time of writing. For more details, we refer the reader to related literature (Goodfellow et al., 2016). *Energy-based models* (LeCun et al., 2006) capture dependencies between random variables i.e. the complex relationships between dimensions of $\boldsymbol{x}$ using so-called *energy function*. Energy function maps different configurations of variables into energy values. If the configuration is meaningful (as suggested by training data) then the energy will be low, otherwise it will be high. Therefore, energy-based methods learn energy landscapes in such a way to assign low energy to training data that is assumed to have meaningful configurations and high everywhere else to create a contrast. The advantage of energy-based models is that they are less restrictive in terms of family of distributions they can capture since there is no strict underlying assumption on how data is generated, no latent variables or conditional factorization. The probability model is unnormalized which removes additional constrains on the models used for describing energy function. However, this also means that computing the exact likelihood and synthesizing samples from is intractable which makes both training and inference challenging. Finally, one of the most recent families of deep generative models are *denoising diffusion probabilistic models* (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020). The idea is to create a parametrized Markov chain that converts observations into pure noise through a sequence of reversible transformations. Because the process is reversible, one can sample observations from noise.

2.4 CONCLUSIONS

In this chapter, we first introduced the idea behind density estimation. We have seen that density estimation is a very general approach to unsupervised learning and that it has numerous real-life applications such as data synthesis and anomaly detection. We then discussed why simple, traditional parametric and nonparametric approaches to density estimation are not well suited for modeling of image and text data which contain complex, nonlinear relationships between observation variables. We introduced deep generative models as a modern approach to density estimation that uses deep neural networks to model those relationships. We also discussed the motivation behind latent variable models from two different perspectives: (i) to enrich the family of probability distributions of parametric models; and (ii) to enable representation learning. The takeaway of this chapter

is that *deep generative models with latent variables* provide a principled unified framework that allows simultaneous learning of extremely rich family of probability distributions and structured data representations. Hence they offer solutions for many important tasks in the domain of unsupervised learning. Next, in Chapter 3 and Chapter 4 we will in detail explore practical challenges in learning deep generative models with latent variables with the particular focus on variational autoencoders, first for sequential data such as text and videos and then for image-based data, respectively.

3

DEEP GENERATIVE LATENT-VARIABLE SEQUENCE MODELING

This chapter is the only part of the thesis that is at the moment half-done.

— Djordje Miladinovic

In this chapter....

3.1 AUTOREGRESSIVE SEQUENCE MODELS AND LATENT VARIABLES

At the time of writing, by far the most popular approach to probabilistic modeling of natural sequences is to combine autoregressive models with the method of maximum likelihood estimation (MLE). This applies to speech (Oord et al., 2018), text (Radford et al., 2019; Brown et al., 2020), videos (Babaeizadeh et al., 2018) and even images as they can be interpreted as sequences of pixels (Van Oord et al., 2016; Van den Oord et al., 2016; Parmar et al., 2018). In autoregressive models, joint probability density function of random sequence $\vec{X}$ is factorized into a product of conditional probabilities such that the conditional distribution of each element in the sequence depends only on the values of the preceding elements, denoted as:

$$p_\theta(\vec{X}) = \prod_i p_\theta(X_i|X_{<i}) \tag{3.1}$$

For example, the sequence element X_i may represent a i^{th} word in the sentence $\vec{X}$ or i^{th} pixel in the image $\vec{X}$. The symbol θ denotes a set of learnable parameters e.g. weights of a neural network that models $p_\theta(X_i|X_{<i})$. Given N training sequences $\{\vec{x}^{(i)}\}_{i=1}^N$, MLE learning of the parameters θ is then performed as:

$$\theta \leftarrow \arg\max_\theta \left[\frac{1}{N} \sum_{i=1}^{N} \log\, p_\theta(\vec{x}^{(i)}) \right] \tag{3.2}$$

where typically, in order to prevent overfitting to the training dataset, early stopping is applied on an isolated validation dataset.

This strategy has proven to be rather effective both in terms of quality of sequence generation and quantitatively, when compared to the alternative solutions in terms of test log-likelihood. The unprecedented success of autoregressive models can be attributed to: *(i)* the abundance of computational power and data which enables relatively fast and parallel training of neural networks with huge number of parameters (Brown et al., 2020); *(ii)* the emergence of well-crafted neural architectures architectures such as transformers (Vaswani et al., 2017) and convolutional networks (Yang et al., 2017) that are suitable for autoregressive models, effective in learning sequential patterns, and allow for parallelized training.

However, in addition to an empirically dominant density estimator, it is often desirable to have the holistic sequence information and uncertainty integrated into a (latent) vector. There are multiple reasons for this. Firstly, this would enable fine-grained control of sequence generation. The randomness in sequence generation in this case would exist at the level of sequences and not at the level of individual samples, which is the case in purely autoregressive models. If in addition, learned distributed representation is 'disentangled' in the sense of factorized latent dimensions (Schmidhuber, 1992; Bengio et al., 2013; Miladinović et al., 2019), then sequence generation could possibly be controlled with respect to independent semantic aspects i.e. in a very fine-grained manner. Secondly, embedding a sequence into a vector would enable sequence representation learning. Learned representations could then be used in downstream analysis, for dimensionality reduction or sequence compression.

But how can one include latent variables into the autoregressive models, to enable learning of distributed sequence representations? One viable strategy is to combine autoregressive models with variational autoencoders (VAEs): The idea is to train a VAE with an autoregressive decoder. This strategy has been previously employed in multiple data domains, for example text (Bowman et al., 2015) images (Chen et al., 2016) and videos (Babaeizadeh et al., 2018). The issue with this approach is that autoregressive decoder is sufficiently expressive to describe the data alone, so it does not utilize latent variables, especially in the presence of additional constraints such as the preassumed prior distribution of latent space, as in VAEs. The underutilized latent variables are naturally uninformative, meaning useless in practice. In the following section, we will have a closer

look at this problem and try to better understand if it is possible to remedy the degenerate learning of VAEs with autoregressive decoders.

3.2 LEARNING (UN)INFORMATIVE REPRESENTATIONS

3.2.1 *Problem setting – VAE with autoregressive decoder*

Consider the following task. We would like to approximate the true probability density function $p_*(\vec{x})$ of random sequence $\vec{X} \sim p_*(\vec{x})$ using a model $p_\theta(\vec{x})$ parameterized by θ. Given a set of observations $\{\vec{x}^{(i)}\}_{i=1}^N$, estimating θ is based on MLE principle, as given in Eq (3.2). Importantly, learned model should be of the form $p_\theta(\vec{x}) = \int p_\theta(\vec{x}|z)p_0(z)dz$ to explicitly expose the representation z. In the framework of VAEs, for any observed $\vec{x}$, the evidence lower bound (ELBO) of $p_\theta(\vec{x})$ is maximized:

$$\mathcal{L}_{\phi,\theta}(\vec{x}) = \mathbb{E}_{q_\phi(z|\vec{x})}[\log\, p_\theta(\vec{x}|z)] - \,\mathrm{KL}(q_\phi(z|\vec{x})||p_0(z)) \tag{3.3}$$

where $q_\phi(z|\vec{x})$ is the approximate posterior parametrized by ϕ and $p_0(z)$ is the assumed prior distribution. For the reasons mentioned in the previous section, VAE decoder is described as an autoregressive model:

$$p_\theta(\vec{x}|z) = \prod_i p_\theta(x_i|\vec{x}_{<i}, z) \tag{3.4}$$

This effectively improves density estimation, however, the problem is that there is no explicit way of controlling (through parameters θ) the utilization of z. In practice, it often happens that $p_\theta(x_i|\vec{x}_{<i}, z) \approx p_\theta(x_i|\vec{x}_{<i})$ rendering z *de facto* uninformative of $\vec{x}$. As a consequence of the unused z and the additional KL term constraint in Eq (3.3), the posterior $q_\phi(z|\vec{x})$ tends to 'collapse' to the prior $p_0(z)$ – this is known as 'posterior collapse' (Van Den Oord et al., 2017).

3.2.2 *Measuring latent information content*

What issues specific to autoregressive decoding are the cause of the degenerate behavior? To reason about this, we can measure the information content in latent space with respect to the decoder output for which we introduce a random variable $\vec{X}$ to denote it.

Definition 1 *Let $p(\vec{\hat{x}}|\vec{x}, z)$ define a probability density function for sequences of random variables $\vec{\hat{X}}$ and $\vec{X}$ of length T[1] and a random variable Z such that $p(\vec{\hat{x}}|\vec{x}, z) = \prod_{i=1}^{T} p_d(\hat{x}_i|\vec{x}_{<i}, z)$ where p_d is an arbitrary probability density function. We define 'latent information content' as $I(\vec{\hat{X}}; Z)$ where I is the mutual information (Cover and Thomas, 2006).*

Note that by introducing $\vec{\hat{X}}$ we do not lose on generality as simply substituting $\hat{x}_i$ with x_i in p_d renders the decoder from Eq (3.4). This simply allows us to differentiate $I(\vec{\hat{X}}; Z)$ from latent information content with respect to the observation $I(\vec{X}; Z)$, so we can study autoregressive decoding in isolation. It is also crucial to highlight that in MLE training $I(\vec{\hat{X}}; \vec{X})$ is maximized, so having low $I(\vec{\hat{X}}; Z)$ implies low $I(\vec{X}; Z)$ as the encoder has no incentive to make useful Z. Next, we examine the issue through the lens of teacher forcing. The illustration is given in Figure 3.1.

3.2.3 *Teacher forcing and the consequence of it*

Teacher forcing (Williams and Zipser, 1989) is a technique for training RNNs that replaces past predictions with past observations. In autoregressive decoding, it emerges as a consequence of MLE training and (in Definition 1) implies using $p_d(\hat{x}_i|\vec{x}_{<i}, z)$ instead of $p_d(\hat{x}_i|\vec{\hat{x}}_{<i}, z)$. On one hand, this significantly improves training convergence. On the other hand, latent information content is reduced:

Proposition 1 *(The consequence of teacher forcing) Assume the setting from Def 1. The following equality holds: $I(\vec{\hat{X}}; Z) = \sum_{i=1}^{T} I(\hat{X}_i; \vec{X}_{<i}, Z) - \sum_{i=2}^{T} I(\hat{X}_i; \vec{X}_{<i}|Z)$. We denote $\sum_{i=2}^{T} I(\hat{X}_i; \vec{X}_{<i}|Z)$ as 'the level of teacher forcing'.*

1 We assume equally long sequences for simplicity and without loss of generality.

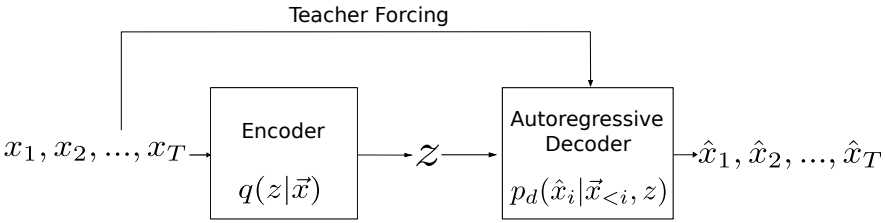


FIGURE 3.1: **MLE training of a sequence autoencoder.** Autoregressive decoder reconstructs input sequence using the information transmitted via the: (i) latent representation z; or (ii) 'side information channel' created as a consequence of teacher forcing. Utilization of the side channel implies the loss of information in z.

Proof. Information-theoretic rules are based on (Cover and Thomas, 2006).

$$
\begin{aligned}
I(\vec{\hat{X}}; Z) &= \sum_{i=1}^{T} I(\hat{X}_i; Z|\vec{\hat{X}}_{<i}) \quad \textit{(chain rule for information)} \\
&= \sum_{i=1}^{T} I(\hat{X}_i; \vec{X}_{<i}, Z|\vec{\hat{X}}_{<i}) - \sum_{i=2}^{T} I(\hat{X}_i; \vec{X}_{<i}|\vec{\hat{X}}_{<i}, Z) \quad \textit{(chain rule for information)} \\
&= \sum_{i=1}^{T} I(\hat{X}_i; \vec{X}_{<i}, Z) - \sum_{i=2}^{T} I(\hat{X}_i; \vec{X}_{<i}|Z) \quad \textit{(follows from Def 1)}
\end{aligned}
$$

Note that to simplify notation, we used $\vec{X}_{<1}$ and $\vec{\hat{X}}_{<1}$ to denote 'empty' random variable.

Takeaway *Teacher forcing enables autoregressive decoders to leverage on past observations but this implies increase in* $\sum_{i=2}^{T} I(\hat{X}_i; \vec{X}_{<i}|Z)$ *hence decrease in* $I(\vec{\hat{X}}; Z)$.

Additional observations

(1) $\sum_{i=1}^{T} I(\hat{X}_i; \vec{X}_{<i}, Z)$ in Prop 1 can be simply increased by using an observation-noise model of lower variance.

(2) Non-autoregressive decoders are described as $\prod_{i=1}^{T} p_d(\hat{x}_i|z)$ so $\sum_{i=2}^{T} I(\hat{X}_i; \vec{X}_{<i}|Z)$ is 0 by design. However, teacher forcing is unutilized.

(3) Structural constraints on Z 'encourage' low $I(\vec{X}; Z)$, for example: (i) KL term in Eq (3.3); (ii) low dimensionality of Z; or (iii) unexpressive encoder $q(z|\vec{x})$.

(4) 'Posterior collapse' in VAEs is a symptom of learning uninformative representations. Teacher forcing is its cause and should be distinguished from local optima (Figure 3.2).

(5) The generality of the setting in Figure 3.1 indicates that issue of uninformative representations can emerge in arbitrary autoencoding settings, e.g. choosing Gaussian $p_d(\hat{x}_i|\vec{x}_{<i}, z)$ and Dirac Delta $q(z|\vec{x})$ renders a deterministic autoencoder.

(6) One plausible way to restrict the teacher forcing information flow is to apply dropout on the ground truth sequence $\vec{X}$ during decoding. We now explore this in more detail.

3.3 RELATED WORK

3.3.1 *Posterior collapse*

The most studied effect of learning uninformative representations is the phenomenon dubbed 'posterior collapse' – the collapse of approximate posterior to the prior in VAEs. It was observed in the context of analyzing text (Bowman et al., 2016; Yang et al., 2017), images (Chen et al., 2016; Razavi et al., 2018), videos (Babaeizadeh et al., 2018), speech (Chorowski et al., 2019) and graphs (Kipf et al., 2018).

Previous solutions can be divided into two complementary categories, tackling two major causes of posterior collapse (Figure 3.2): **(i)** those dedicated to solving optimization challenges due to local optima of the objective function; **(ii)** those that implicitly or explicitly regulate teacher forcing. Note that Figure 3.2 omits some other causes, in particular: **(i)** learned or fixed excessively large observation-noise variance of the decoder (Lucas et al., 2019; Dai et al., 2020) (also discussed in Section 3.2.3); **(ii)** benign pruning of unnecessary latent dimensions (Dai et al., 2020).

Existing solutions that mitigate local optima (Bowman et al., 2016) suggest 'KL annealing' to gradually introduce the KL term into training. 'Free

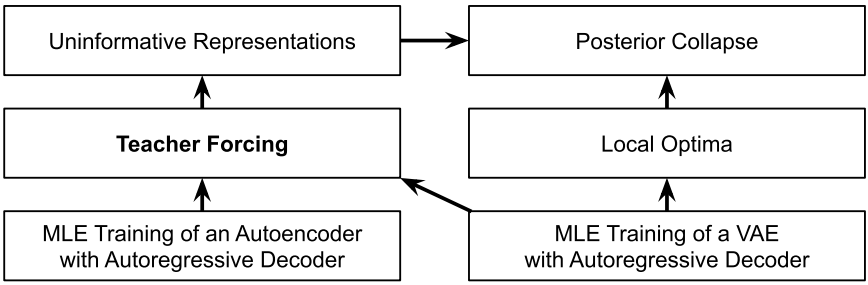


FIGURE 3.2: **Causes and consequences of teacher forcing.**

bits' technique (Kingma et al., 2016) avoids penalizing the KL term if its magnitude is under some predefined threshold. (Van Den Oord et al., 2017) use a discrete VAE. (Babaeizadeh et al., 2018; Razavi et al., 2019) introduce a warm-up phase in the spirit of KL annealing. (Razavi et al., 2018) constrain the variational family of the encoder. (He et al., 2019) aggressively optimize the encoder before each model update. (Dieng et al., 2019) use skip connections to incentivize the utilization of latent variables, effectively modifying the loss landscape.

Existing solutions that regulate teacher forcing (Bowman et al., 2016) apply 'word dropout' to randomly drop words during the decoding in attempt to 'weaken the decoder'. (Chen et al., 2016; Semeniuta et al., 2017; Yang et al., 2017) constrain the receptive field of the decoder neural network, effectively reducing the window of autoregression. Unfortunately this is not applicable for RNN architectures which have unbounded receptive field. Dynamic dropout falls into this category and is most similar to word dropout; the comparison is discussed in detail in Section 3.4.

3.3.2 *Exposure bias*

'Exposure bias' is a term that originates from natural language processing (Ranzato et al., 2015) and refers to teacher forcing-induced gap between training and inference. The problem appears during inference when trained model is required to generate sequences without utilizing ground truth history which was accessible during training due to teacher forcing. Since the model was never trained to continue its own predictions, this re-

sults in quick error accumulation. The developed solutions are interesting in our context because they too attempt to regulate teacher forcing.

Existing solutions The most popular strategy is to switch between conditioning on ground truth and model predictions where the latter is preferred towards the end of training (Daumé et al., 2009; Ross et al., 2011; Venkatraman et al., 2015; Bengio et al., 2015; Ranzato et al., 2015). Multi-step predictors (Finn et al., 2016; Kipf et al., 2018; Hafner et al., 2019) fall into this category. The main difficulty is to define an optimal sequence continuation conditioned on the model predictions. The simplest approach is to always use k-th word of the target sequence to compare with k-th word of the predicted sequence (Venkatraman et al., 2015; Bengio et al., 2015). Other related approaches extend this idea by including domain-specific sequence-level loss (Ranzato et al., 2015; Bahdanau et al., 2016; Zhang et al., 2019b) such as BLEU score (Papineni et al., 2002).

In our context, scheduled sampling (Bengio et al., 2015) is applicable and was examined in our experiments. In fact, the third variant of dynamic dropout can be regarded as an extension of the scheduled sampling as discussed in detail in Section 3.4. On the other hand, the methods based on the sequence-level loss are considerably more involving implementation-wise and require that the loss is specified, however, it is not entirely clear which loss is suitable for learning informative representations in general.

3.3.3 *Dropout*

Dropout (Srivastava et al., 2014) was originally introduced as a technique for regularization of deep neural networks. The idea is to randomly drop individual activations of a neural network layer based on some predetermined dropout probability. In the context of our problem, dropout is applicable to reducing the information flow in the decoder. Variational dropout (Kingma et al., 2015) was developed based on a Bayesian interpretation of neural networks. It enables the adaptation of dropout regularization rates across neural network weights. Information dropout (Achille and Soatto, 2018) is based on the same principle but applied to representation learning. The two methods are similar to ours in the sense that the regularization is performed via an objective function, however, they are of continuous nature and are not specifically tailored to autoregressive setting.

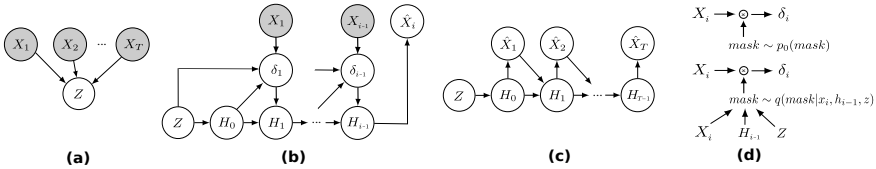


FIGURE 3.3: **VAE with an autoregressive RNN decoder. (a)** In sequence encoder, latent variable Z is inferred from the input sequence $\vec{X}$. **(b)** During maximum-likelihood training, ground truth sequence history is used (teacher forcing) and next-step prediction is performed. δ_i is an auxiliary variable we introduced to better explain the method; **(c)** During inference (at test time), the previous predictions of the model are used; **(d)** In regular dropout, the dropout mask is sampled from a prespecified prior *(scheme above)*. In dynamic dropout, it is sampled from the posterior conditioned on the specified context which in the most general case is given as $[X_i, H_{i-1}, Z]$ *(scheme below)*.

3.4 VAE-RNN AND DYNAMIC DROPOUT

Keeping the notation from Section 3.2.1, we now introduce the dynamic dropout method by applying it to a VAE framework with an RNN-based autoregressive decoder (Bowman et al., 2016). VAE-RNN is illustrated in Figure 3.3. Note how we explicitly differentiate between the maximum likelihood training regime in which teacher forcing is applied (Figure 3.3b) and the inference regime in which previous predictions are fed to RNN (Figure 3.3c). Sequence encoder is in Figure 3.3a graphically presented in the most general setting, though in practice we implement it using another RNN (Bowman et al., 2016). To simplify the explanation to follow, we introduce an auxiliary random variable $\delta_i = f(X_i, H_{i-1}, Z)$. This is without loss of generality as simply specifying $\delta_i = X_i$ renders the vanilla RNN setting. We next present three variants of the dynamic dropout method. The basic idea is illustrated in Figure 3.3d and the corresponding code written in Pytorch (Paszke et al., 2019) is provided at `https://github.com/djordjemila/dd`.

3.4.1 *Dynamic word dropout*

In the framework from Figure 3.3, word dropout (Iyyer et al., 2015; Gal and Ghahramani, 2016; Bowman et al., 2016) method can be described as $\delta_i = (1 - D_i) \cdot X_i$ where $D_i \sim \text{Bern}(p_d)$. We have $D_i \in \{0, 1\}$ and p_d denoting the probability of 'dropping' X_i that is specified before the training. Note that when computing δ_i, each dimension of X_i is multiplied by the same scalar $(1 - D_i)$.

Dynamic word dropout is likewise based on the gating-like principle, acting as a binary switch that either drops the information fully or not at all. However, the probability of dropping X_i is not predetermined but inferred on the fly. Concretely, we now have $\delta_i = (1 - D_i) \cdot X_i$ where $[1 - D_i, D_i] \sim \text{STGumbelSoftmax}([1 - Y_i, Y_i])$. STGumbelSoftmax is the Straight-Through Gumbel-Softmax estimator defined in (Jang et al., 2016; Maddison et al., 2016). Note that here we assume that STGumbelSoftmax returns one-hot vector representation. STGumbelSoftmax enables us to efficiently compute $D_i \in \{0, 1\}$ while being able to differentiate through the sampling procedure and calculate low-variance gradients during backpropagation. $Y_i \in (0, 1)$ is computed as $Y_i = g(X_i, H_{i-1}, Z)$ hence the dropout probability crucially depends on the current context (Figure 3.3d). Intuitively, this allows RNN to dynamically select which input to keep and which to drop. g can be implemented as a one or two-layered dense neural network with a sigmoid activation function.

Because *per se* there is no incentive to drop useful information stored in X_i, in the naïve setting, the network $g(X_i, H_{i-1}, Z)$ would simply learn to generate the values $Y_i \to 0$ i.e. it would never drop X_i. To that end, a prior dropout probability is specified and then integrated into the modified evidence lower bound as follows:

$$\mathcal{L}(\vec{x}) = \mathbb{E}_{q(\vec{h}|\vec{x})}\left[\sum_{i=1}^{T} \log\, p(x_i|h_{i-1})\right] \tag{3.5}$$

$$- \text{KL}(q(z|\vec{x})||p_0(z)) \tag{3.6}$$

$$- \underbrace{\mathbb{E}_{q(\vec{y}|\vec{x})}\left[\beta \sum_{i=1}^{T-1} \text{KL}(\text{Bern}(y_i)||\text{Bern}(p_d))\right]}_{\text{dynamic dropout regularization term}} \tag{3.7}$$

If the Kullback-Leibler divergence between the two Bernoulli distributions is expanded, the term in (3.7) reads as:

$$-\mathbb{E}_{q(\vec{y}|\vec{x})}\left[\beta\sum_{i=1}^{T-1}(y_i\log\frac{y_i}{p_d}+(1-y_i)\log\frac{(1-y_i)}{(1-p_d)})\right] \tag{3.8}$$

There are overall two hyperparameters: p_d specifies our prior belief on the portion of data to be dropped and β is the magnitude of the regularization. In the limiting case $\beta \to \infty$, dynamic word dropout acts as word dropout. In contrast to word dropout, the regularization is made explicit and a part of the loss function. This is relevant because: **(a)** autoregressive decoder can dynamically determine what information to keep with higher probability; **(b)** dropout rate can possibly vary over the course of training.

3.4.2 *Dynamic activation dropout*

In the similar spirit we design a dynamic variant of the vanilla dropout (Srivastava et al., 2014). In the vanilla dropout we have $\delta_i = D_i \odot X_i$ where D_i is a vector of the same size as X_i and $\odot$ is the element-wise product. Each dimension j of D_i, denoted as D_{ij}, is given as $D_{ij} \sim \text{Bern}(p_d)$ where p_d is the prespecified dropout probability. Following the same reasoning as above, we can extend the vanilla dropout by inferring a vector Y_i (instead of a scalar) and reparametrizing each of its dimensions $Y_{ij} \in (0, 1)$ using STGumbelSoftmax. The regularization term from Eq (3.8) can be analogously computed for Y_{ij}.

3.4.3 *Dynamic scheduled sampling*

The third variant is the extension of scheduled sampling (Bengio et al., 2015). In dynamic scheduled sampling we have $\delta_i = (1 - D_i) \cdot X_i + D_i \cdot \hat{X}_i$. The difference to dynamic word dropout is that the input is not dropped by replaced with model prediction. The loss function remains the same.

3.4.4 *On specifying the context*

Figure 3.3d presents the general case how the to condition the mask generation on. It is also possible to condition on Z only in which case the dropout rates will vary across sequences but not throughout. We used this strategy in the video modeling experiments whereas for language modeling we found it critical to include the hidden state to enable adapting the dropout probabilities across a sentence.

3.5 RESULTS

3.5.1 *Language modeling*

We evaluated dynamic dropout and the competing methods on the language modeling task using the Penn Treebank dataset (Marcus et al., 1993), applying the training/validation/test split and preprocessing from (Mikolov et al., 2010). The main goal of our experiments was to show that dynamic dropout is empirically superior to existing approaches in terms of teacher forcing regularization and consequently learning of informative latent variables. We also provide insights in how dynamic dropout internally operates.

EXPERIMENTAL SETUP Our implementation of VAE-RNN model is similar to the one presented in (Bowman et al., 2016). In addition, we introduced some modifications which we found useful for improving the overall training stability. In particular, we used GRU units (Cho et al., 2014) with a hidden state of size 512, both in encoder and decoder. The size of the latent space was set to 64 and the size of learnable embedding layer was set to 512. We applied Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.003, $\beta = (0.9, 0.999)$ and an exponential decay of 0.9. We used mini-batches of size 128. We did not use skip connections which we found leading to unsatisfactory performance in terms of learning informative representations. The training was performed using standard maximum likelihood objective by maximizing the evidence lower bound from Eq (3.3) for baselines and Eq (3.5-3.7) for dynamic dropout. Early stopping was applied based on the performance on the validation set when the re-

construction (RC) term i.e. the negative value of the left hand side of Eq (3.3), was at its lowest – the main idea was to ensure that the uncertainty modeling is stored in the latent space of VAE-RNN. To mitigate undesirable local optima (discussed in Section 3.3) we applied linear KL annealing procedure (Bowman et al., 2016) across 9000 batch updates. For each baseline, we evaluated 10 random seeds on a single GPU of type 'GeForce GTX 1080 Ti' and then chose the top performing runs on the validations set with respect to the RC term.

EVALUATION STRATEGY AND METRICS To validate the information content in the latent space of VAE-RNN we observed the KL term – the right hand side of Eq (3.3) – which is expected to be high for informative representations, and the RC term which is ideally as low as possible, as measured on the test set. Very low KL term typically indicates 'posterior collapse' as noted in Section 3.3. Similar procedure was performed in (Bowman et al., 2016). In addition, we evaluated the quality of reconstruction in terms of (Sacre)BLEU score (Post, 2018)[2] between input and output test sentences, importantly, without using teacher forcing during generation i.e. following the generation process from Figure 3.3c. Lastly, we measured the reconstruction accuracy as a percentage of correctly predicted words.

BASELINES Our main baseline was word dropout which was previously observed to have superior performance (Bowman et al., 2016). We swept through the dropout values $\in \{30\%, 40\%.., 70\%\}$. We also tested the unregularized baseline. We were not able to get any good results without applying teacher forcing. Furthermore, we experimented with the vanilla dropout (Srivastava et al., 2014) and scheduled sampling (Bengio et al., 2015) applying the same dropout/teacher forcing rates. The best performing variant of our method was dynamic word dropout for which we first performed preliminary tests to roughly estimate the desirable range of β and then swept through values $\beta \in \{1, 10, 20, ..., 100\}$ and for the dropout probability prior $\in \{50\%, 60\%.., 90\%\}$.

QUANTITATIVE ANALYSIS Table 3.1 presents the comparison of the competing methods for the top performing runs of selected baselines. We can observe that dynamic dropout offers superior performance in terms of all

2 We computed BLEU using: https://github.com/mjpost/sacrebleu

Model	-ELBO ↓	RC ↓	KL ↑	Accuracy [%] ↑	BLEU ↑
Baseline (no regularization)	115	97	18	32	7
+ Vanilla dropout (60%)	111	95	16	33	8
+ Vanilla dropout (70%)	112	96	16	33	8.5
+ Scheduled sampling (60%)	111	95	16	33	8.2
+ Scheduled sampling (70%)	113	94	19	33	8.7
+ Word dropout (50%)	**109**	94	15	33	8
+ Word dropout (60%)	110	91	19	34	9.6
+ Word dropout (70%)	113	92	21	34	9.8
+ Word dropout (80%)	116	91	**25**	35	10.5
+ Dynamic word dropout (60%, $\beta = 60$)	112	**86**	**25**	**37**	**11.5**

TABLE 3.1: **Results of modeling sentences on the Penn Treebank test set.**

the latent information content-related metrics *de facto* without sacrificing the density estimation performance in terms of evidence lower bound.

DYNAMIC DROPOUT ANALYSIS We qualitatively examined the dropout patterns by monitoring the words that dynamic word dropout tended to 'keep' i.e. drop with lower probability. Example sentences are given in Table 3.2. The intuition is that the words which VAE-RNN did not memorize and that it was not able to predict with high certainty during decoding were more likely to be kept. We observed that typically the key words of input sentences were not kept and we speculate that these were memorized in the latent space as they contain the semantics of the sentence. Instead, the linking words, prepositions and articles were generally kept with higher probability. We also noticed the decrease in the dropout probability towards the end of a sentence. This also makes sense: As the memory stored in the RNN hidden states 'fades away' across the processed sequence, the chances of keeping a word increase.

QUALITATIVE ANALYSIS We qualitatively compared the difference between sentence reconstructions in the cases when: *(i)* no regularization was applied; and *(ii)* dynamic dropout was applied. Some example sentence reconstructions are given in Table 3.3. We generally found both methods to be able to produce sentences which appear as plausible English, however, the regularized model arguably reproduced the semantics more closely in comparison to the input sentence.

who is going to be **in the** space next door
this has both made investors uneasy and **the** corporations more vulnerable
although they represent only n n of the population they control nearly one-third **of** discretionary income
the economy does however depend on the confidence **of** businesses consumers and foreign investors
if they could n't sell some of them put the shares **back on** the shelf

TABLE 3.2: The words that were kept with higher probability when applying dynamic word dropout (bolded).

Input sentence	proceeds will be used to eliminate and restructure bank debt
No dropout	proceeds will be used for remodeling and other products
Dyn. dropout	proceeds will be used to reduce debt
Input sentence	pricing details were n't immediately available
No dropout	pricing details were n't available to comment
Dyn. dropout	pricing details were n't available
Input sentence	the announcement caused the company 's stock to surge $ n to close at $ n a share
No dropout	the announcement follows a sharper $ n billion quarterly loss from the sale of new shares since june
Dyn. dropout	the announcement follows the company 's stock price to $ n a share up $ n

TABLE 3.3: Sentence reconstructions of an unregularized and a dynamic word dropout-regularized model.

3.5.2 *Video modeling*

We further inspected the effects of the teacher forcing regularization on the data of continuous nature. In particular, we applied dynamic dropout

and the corresponding baselines on the task of modeling video sequences with informative latent variables. We analyzed two datasets: **(i)** synthetic renderings of a bouncing ball (Miladinović et al., 2019) where the ball kinematics depends on the gravity that varies across sequences. This dataset was useful for visualizing the effects of the regularization as a simple neural architecture was sufficient to provide visually appealing video renderings; **(ii)** KTH action dataset (Schuldt et al., 2004) which contains real-life videos of people performing various activities. We found modeling KTH dataset significantly more challenging and this is where we performed the quantitative analysis.

DATASET PREPARATION AND PREPROCESSING For the bouncing ball dataset we generated 10000/1000 videos to make a training/validation split following (Miladinović et al., 2019). We did not use the test set as we only performed qualitative analysis on this data. The sequences were 20 frames long. We used 8-gravity setting as opposed to to the 4-gravity one described in (Miladinović et al., 2019). We used Bernoulli distribution as an observation model. The generation script is provided in the code that accompanies the paper. For the KTH dataset, we split the videos into sequences shorter than the original ones in order to enlarge the dataset. We rendered 3013/335 samples of training/validation data. We then removed all the videos containing the background only. The sequences are 20 frames long. The generation script is provided in the code that accompanies the paper.

PRELIMINARY QUALITATIVE ANALYSIS To better understand the benefits and the repercussions of teacher forcing, we evaluated dynamic dropout on the bouncing balls data sets and compared the resulting model with: *(i)* the one trained with unregulated teacher forcing; and *(ii)* the one in which teacher forcing was not applied at all. This is illustrated in Figure 3.4. The teacher forcing baseline provides a model which is optimized for next-step prediction and reaches the highest test log-likelihood (the dataset is simple enough so we did not observe any overfitting). The reconstructed sequence is hence meaningful and smooth, however, it is not a good representative of the original sequence indicating that no useful information is stored in the latent space. On the other hand, the baseline in which no teacher forcing was applied can reconstruct the sequence well in terms of conditional log-likelihood, however, the rendered sequence is not visually appealing

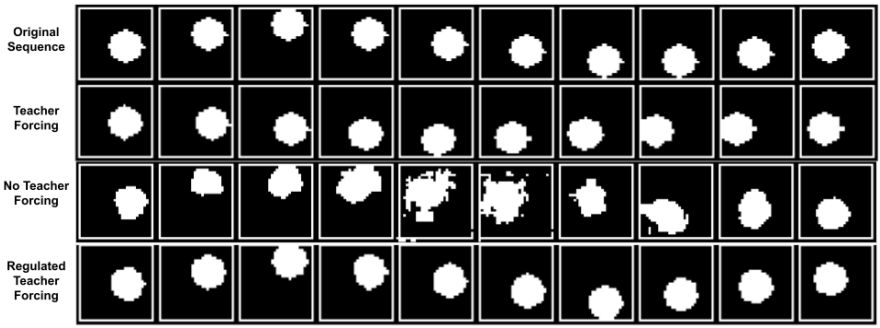


FIGURE 3.4: **Qualitative analysis of teacher forcing and the effects of the regularization on the bouncing balls data set.**

and smooth. Finally, the regularized model renders a plausible sequence which is a good representative of the original sequence

EXPERIMENTAL SETUP We now describe the experimental setup used for the KTH dataset modeling experiments. The VAE-RNN architecture we used is very similar to the one described in Sections 3.4 and 3.5.1 except that instead of using word embeddings we used 4-layered convolutional networks. Furthermore, we used GRU hidden state of size 512, Adam with $\beta = (0.9, 0.999)$ and learning rate of 0.001 with the exponential decay of 0.9995. The mini-batches were of size 128 and the latent space was of size 128. In contrary to our language experiments, we did not observe any issues with respect to local optima so did not apply KL annealing or any related technique. Early stopping was performed when RC term stopped improving on the validation set for the reasons mentioned earlier. KTH is made of grayscale images so we used discretized logistic distribution (Kingma et al., 2016) to describe them. Again, we chose the top among 10 random seeds.

EVALUATION STRATEGY, METRICS AND BASELINES We validated the quality of models based on: *(i)* the magnitude of the KL term; *(ii)* the quality of video reconstruction in the inference regime when no teacher forcing was applied (Figure 3.3c). For this we used metrics typically used in video analysis: structural similarity (SSIM), Peak Signal-to-Noise Ratio (PSNR) and mean squared error (MSE), measured between the input sequence and

Model	-ELBO [$\times 10^3$] ↓	RC [$\times 10^3$] ↓	KL ↑	PSNR ↑	MSE [$\times 10^{-3}$] ↓	SSIM ↑
Baseline (no regularization)	279	279	190	24.9	5.3	0.62
+ Scheduled sampling (70%)	286	286	198	26.1	3.8	**0.68**
+ Vanilla dropout (10%)	286	286	200	26.2	3.9	**0.68**
+ Word dropout (50%)	286	286	201	25.8	4.1	0.67
+ **Dynamic sch. sampling** (60%, $\beta = 10$)	286	286	**206**	**26.7**	**3.6**	**0.68**

TABLE 3.4: **Results of modeling video sequences on the KTH validation set.**

FIGURE 3.5: **Reconstructing KTH sequence *(on top)* using a regularized baseline *(middle)* and an unregularized one *(bottom)*.**

the reconstructed output. Note that these metrics are analogous to BLEU used in our language modeling experiments. We also observed whether the evidence lower bound was sufficiently low i.e. we tried to identify the models which provide a good trade-off between the quality of density estimation and the informativeness of the latent space. In contrast to language experiments however, we found that this trade-off was more pronounced i.e. more strongly affected by the magnitude of the dropout. This is likely because there was less overfitting (compare the performance of the unregularized baselines in the two different tasks). To that end, we used the same baselines but selected only the configurations which had approximately equal RC term and then compared the remaining metrics.

QUANTITATIVE AND QUALITATIVE ANALYSIS Selected runs of the KTH experiments are given in Table 3.4. We can observe that dynamic dropout generally gives better results across multiple metrics. Example video sequence reconstructions are given in Figure 3.5.

DYNAMIC DROPOUT ANALYSIS The subsequent sequence samples in video data are rather similar due to temporal coherence. We speculate that this was the reason why we obtained better results when conditioning the dropout inference on the latent variable Z only. This also why it was not possible to make meaningful interpretations about which image frames

where kept with higher probability like in the language experiments. However, we did observe interesting behaviour in terms of average dropout rate change over the course of training. This is illustrated in Figure 3.6. We can see that as the training progresses, the model leverages on teacher forcing with an increasing magnitude. This is somewhat counter-intuitive and in contrast to some previous work which suggest that the annealing schedule of teacher forcing should be decreased throughout training (Bengio et al., 2015). While in principle one could hand-craft a similar schedule for the vanilla dropout variants, this is very much data dependent and difficult to tune. Note also that the high-frequency fluctuations in dropout rate can be explained as batch-to-batch variance in batch average dropout probabilities.

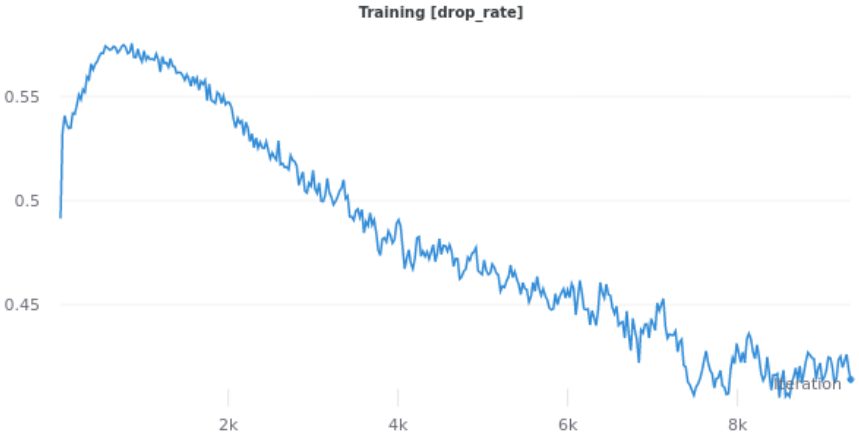


FIGURE 3.6: **The change in average drop rate across training, in number of batch updates.** The prior was 60% and $\beta = 10$. The x axis in the figure represents the number of batch updates (iterations) and the y axis represents the dropout rate percentage $\in (0, 1)$

3.6 SUMMARY

In this chapter we explored the adverse consequences of teacher forcing on learning informative representations in autoregressive models with latent variables. We proposed (three variants of) a novel dropout-based regularization technique to preserve the information in latent space while main-

taining strong density estimation performance. Dynamic dropout was introduced in the context of a VAE-RNN-based architecture and was examined with respect to real-life natural language and video sequences. We showed that dynamic dropout offers more flexibility than vanilla variants and that it can lead to better models. In the future, it would be beneficial to investigate the applicability of dynamic dropout to: *(i)* other types of data e.g. images or speech; *(ii)* other architectures such as transformers or convolutional network-based autoregressive models; *(iii)* solving exposure bias.

4

DEEP GENERATIVE LATENT-VARIABLE IMAGE MODELING

What I cannot create, I do not understand.
— Richard Feynman

In Chapter 3, we have analyzed the challenges in training autoregressive sequence models with latent variables. We have seen that due to teacher forcing that arises as a consequence of maximum likelihood estimation, the näive training renders meaningless encoders and uninformative latent representations. We have proposed to train sequence models using dynamic dropout – a technique that regulates teacher forcing encouraging information to flow through latent space, making it more informative. In this chapter, we explore a way to train deep generative latent-variable models for images *without teacher forcing*, alleviating negative consequences to latent space information content. In particular, we present a purely VAE architecture that is based on two key components: *(i)* a *hierarchical (ladder) network* that contains shortcut connections between the encoder and the decoder, improving overall training convergence; and *(ii)* a novel powerful decoder in the form of a *spatial dependency network (SDN)* for better modeling of spatial coherence and dependencies in images. This is similar to autoregressive modeling of temporal coherence between subsequent sequence elements. SDN can been seen as a multi-directional autoregressive model that operates at different layers of a deep neural network without leveraging on teacher forcing; We show on multiple different data sets that SDN-based VAEs obtain state-of-the-art results in density estimation, generate perceptually appealing images, and learn meaningful latent representations. Lastly, we also show that the newly proposed SDN is applicable to image generation tasks beyond VAEs by combining them with U-Nets to perform medical image segmentation.

4.1 SPATIAL DEPENDENCY NETWORKS

We start this chapter by introducing spatial dependency network (SDN) as a new type of neural network designed for constructing image decoders i.e. image generators. The motivation is to use it in the context of a deep generative model with latent variables. In Section 4.2, we discuss how to apply spatial dependency network to model the decoder of a variational autoencoder (VAE), resulting in a powerful density model with meaningful encoder and informative latent space. In Section 4.3 and Section 4.4, we analyze SDN and the corresponding VAEs experimentally.

4.1.1 *Introduction and motivation*

The abundance of data and computation are often identified as core facilitators of the deep learning revolution. In addition to this technological leap, historically speaking, most major algorithmic advancements critically hinged on the existence of inductive biases, incorporating prior knowledge in different ways. Main breakthroughs in image recognition (Cireşan et al., 2012; Krizhevsky et al., 2012b) were preceded by the long-standing pursuit for shift-invariant pattern recognition (Fukushima and Miyake, 1982) which catalyzed the ideas of weight sharing and convolutions (Waibel, 1987; LeCun et al., 1989). Recurrent networks (exploiting temporal recurrence) and transformers (modeling both temporal recurrence and the "attention" bias) revolutionized the field of natural language processing (Mikolov et al., 2011; Vaswani et al., 2017). Visual representation learning is also often based on priors e.g. independence of latent factors (Schmidhuber, 1992; Bengio et al., 2013) or invariance to input transformations (Becker and Hinton, 1992; Chen et al., 2020). Clearly, one promising strategy to move forward is to introduce more structure into learning algorithms, and more knowledge on the problems and data.

Along this line of thought, we explore a way to improve the architecture of deep neural networks that generate images, here referred to as (deep) image generators, by incorporating prior assumptions based on topological image structure. More specifically, we aim to integrate the priors on spatial dependencies in images. We would like to enforce these priors on all intermediate image representations produced by an image generator, including the last one from which the final image is synthesized. To that end, we in-

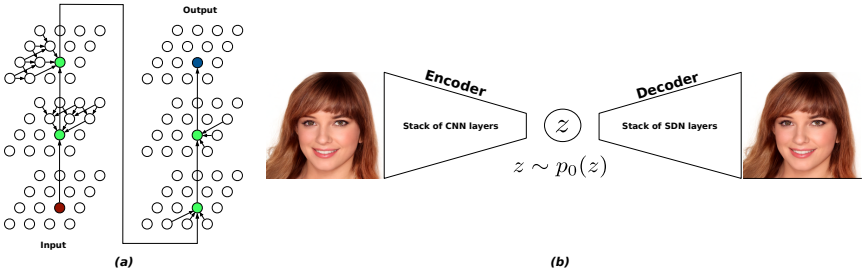


FIGURE 4.1: **(a) DAG of a spatial dependency layer.** The input feature vector (red node) is gradually refined (green nodes) as the computation progresses through the four sub-layers until the output feature vector is produced (blue node). In each sub-layer, the feature vector is corrected based on contextual information. Conditional maps within sub-layers are implemented as learnable deterministic functions with shared parameters; **(b) VAE with SDN decoder reconstructing a 'celebrity'.**

troduce a class of neural networks designed specifically for building image generators – spatial dependency network (SDN). Concretely, spatial dependency layers of SDN (SDN layers) incorporate two priors: *(i)* spatial coherence reflects our assumption that feature vectors should be dependent on each other in a spatially consistent, smooth way. Thus in SDN, the neighboring feature vectors tend to be more similar than the non-neighboring ones, where the similarity correlates with the 2-D distance. The graphical model of a spatial dependency layer (Figure 4.1a) captures this assumption. Note also that due to their unbounded receptive field, SDN layers model long-range dependencies; *(ii)* spatial dependencies between feature vectors should not depend on their 2-D coordinates. Mathematically speaking, SDN should be equivariant to spatial translation, in the same way convolutional networks (CNN) are. This is achieved through parameter (weight) sharing both in SDN and CNN;

The main focus of this chapter is the application of SDN to variational autoencoders (VAEs) (Kingma and Welling, 2013). Our motivation is to improve the performance of VAE generative models by endowing their image decoders with spatial dependency layers (Figure 4.1b). However, note that SDN could also be applied to other generative models, e.g. generative adversarial networks (Goodfellow et al., 2014). More generally, SDN could be potentially used in any task in which image generation is required, such

as image-to-image translation, super-resolution, image inpainting, image segmentation, or scene labeling.

In Section 4.3. we examine SDN in two different VAE settings. In the context of real-life-image density modeling, SDN-empowered hierarchical VAE is shown to reach considerably higher test log-likelihoods than the baseline CNN-based architectures and can synthesize perceptually appealing and coherent images even at high sampling temperatures. In a synthetic data setting, we observe that enhancing a non-hierarchical VAE with an SDN facilitates learning of factorial latent codes, suggesting that unsupervised 'disentanglement' of representations can be bettered by using more powerful neural architectures, where SDN stands out as a good candidate model. Finally in Section 4.4, unrelated to the main topic of this thesis but important for demonstrating the applicability of SDN beyond VAEs, we show that when combined with a U-Net, SDN leads to state-of-the-art results in medical image segmentation.

4.1.2 *Architecture*

The architectural design of SDN is motivated by our intent to improve modeling of spatial coherence and higher-order statistical correlations between pixels. By SDN, we refer to any neural network that contains at least one spatial dependency layer (Figure 4.2). Technically speaking, each layer of an SDN is a differentiable function that transforms input to output feature maps in a 3-stage procedure. We first describe the SDN layer in detail, then address practical and computational concerns.

PROJECT-IN STAGE Using a simple feature vector-wise affine transformation, the 3-D input representation X^{L_k} at the layer L_k is transformed into the 3-D intermediate representation I^{L_k} as

$$I^{L_k}_{i,j} = X^{L_k}_{i,j} W^{(1)} + \mathbf{b}^{(1)} \tag{4.1}$$

where i and j are the corresponding 2-D coordinates in X^{L_k} and I^{L_k}, $\mathbf{b}^{(1)}$ is a vector of learnable biases and $W^{(1)}$ is a learnable matrix of weights whose dimensions depend on the number of channels of X^{L_k} and I^{L_k}. In the absence of overfitting, it is advisable to increase the number of channels to enhance the memory capacity of I^{L_k} and enable unimpeded information flow in the correction stage. Notice that described transformation corresponds

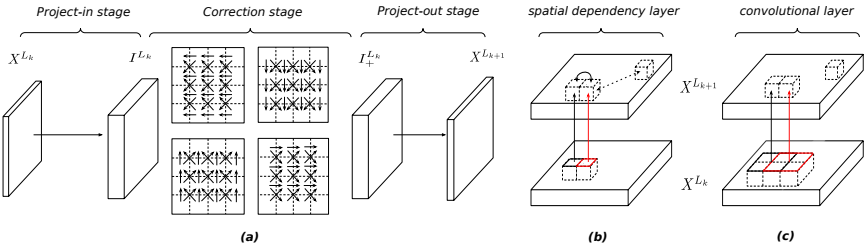


FIGURE 4.2: **Spatial dependency layer. (a)** a 3-stage pipeline; **(b)** dependencies between the input X^{L_k} and the output $X^{L_{k+1}}$ modeled by a spatial dependency layer L_k. Solid arrows represent direct, and dashed indirect (non-neighboring) dependencies. Note that the sub-layers from Figure 4.1a are regarded as 'latent' here; **(c)** dependencies modeled by a 2×2 convolutional layer.

to the convolutional (CNN) operator with a kernel of size 1. If the scales between the input X^{L_k} and the output $X^{L_{k+1}}$ differ (e.g. if converting an 8×8 to a 16×16 tensor), in this stage we additionally perform upsampling by applying the corresponding transposed CNN operator, also known as 'deconvolution' (Zeiler and Fergus, 2014).

CORRECTION STAGE The elements of I^{L_k} are 'corrected' in a 4-step procedure. In each step, a unidirectional sweep through I^{L_k} is performed in one of the 4 possible directions: *(i)* bottom-to-top; *(ii)* right-to-left; *(iii)* left-to-right; and *(iv)* top-to-bottom; During a single sweep, the elements of the particular column/row are computed in parallel. The corresponding correction equations implement controlled updates of I^{L_k} using a gating mechanism, as done in popular recurrent units (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). Specifically, we use a multiplicative update akin to the one found in GRUs (Cho et al., 2014), but adapted to the spatial setting. The procedure for the bottom-to-top sweep is given in Algorithm 1 where W^* denote learnable weights, B^* are learnable biases, i is the height (growing bottom-to-top), j is the width and c is the channel dimension. Gating elements $r_{i,j,c}$ and $z_{i,j,c}$ control the correction contributions of the *a priori* value of $I_{i,j,c}$ and the proposed value $n_{i,j,c}$, respectively. For the borderline elements of I^{L_k}, missing neighbors are initialized to zeros (hence the zero padding). The sweeps in the remaining 3 directions are performed analogously.

ALGORITHM 1: Bottom-to-top sweep of the correction stage

Input: I^{L_k} – intermediate representation; N – the scale of I^{L_k};
Output: $I_+^{L_k}$ – corrected intermediate representation; # after all 4 sweeps

Complexity: $\mathcal{O}(N)$
$I^{L_k} = \tanh(I^{L_k})$ # done for the first direction only
$I^{L_k} = zero\_padding(I^{L_k})$ # done for the first direction only
for $i = 1$ **to** N and ($j = 1$ **to** N **in parallel**) **do**
$r_{i,j,c} = \sigma([W^{r1} I_{i,j}]_c + [W^{r2} I_{i-1,j}]_c + [W^{r3} I_{i-1,j-1}]_c + [W^{r4} I_{i-1,j+1}]_c + B^r_{i,j,c})$
$z_{i,j,c} = \sigma([W^{z1} I_{i,j}]_c + [W^{z2} I_{i-1,j}]_c + [W^{z3} I_{i-1,j-1}]_c + [W^{z4} I_{i-1,j+1}]_c + B^z_{i,j,c})$
$n_{i,j,c} = \tanh(r_{i,j,c}[W^{n1} I_{i,j}]_c + [W^{n2} I_{i-1,j}]_c + [W^{n3} I_{i-1,j-1}]_c + [W^{n4} I_{i-1,j+1}]_c + B^n_{i,j,c})$
$I_{i,j,c} = z_{i,j,c} I_{i,j,c} + (1 - z_{i,j,c}) n_{i,j,c}$
end for

PROJECT-OUT STAGE Corrected (*a posteriori*) representation $I_+^{L_k}$ is then mapped to the output $X^{L_{k+1}}$, which has the same number of channels as the input X^{L_k}:

$$X_{i,j}^{L_{k+1}} = I_{+,i,j}^{L_k} W^{(3)} + \mathbf{b}^{(3)} \tag{4.2}$$

where $W^{(3)}$ is the stage-level learnable weight matrix and $\mathbf{b}^{(3)}$ is the corresponding bias vector.

COMPUTATIONAL CONSIDERATIONS A valid concern in terms of scaling SDN to large images is the computational complexity of $\mathcal{O}(N)$, N being the scale of the output. We make two key observations:

1. one can operate with less than 4 directions per layer, alternating them across layers to achieve the 'mixing' effect. This improves runtime with some sacrifice in performance;

2. in practice, we found it sufficient to apply spatial dependency layers only at lower scales (e.g. up to 64×64 on CelebAHQ256), with little to no loss in performance. We speculate that this is due to redundancies in high-resolution images, which allows modelling of relevant spatial structure at lower scales;

To provide additional insights on the computational characteristics, we compared SDN to CNN in a unit test-like setting (Appendix A.1.2). Our implementation of a 2-directional SDN has roughly the same number of parameters as a 5×5 CNN, and is about 10 times slower than a 3×3 CNN. However, we also observed that in a VAE context, the overall execution time discrepancy is (about 2-3 times) smaller, partially because SDN converges in fewer iterations (see Table 4.2).

AVOIDING VANISHING GRADIENTS The problem might arise when SDN layers are plugged in a very deep neural network (such as the one discussed in Section 4.2). To remedy this, in our SDN-VAE experiments we used the variant of an SDN layer with a 'highway' connection, i.e. gated residual (Srivastava et al., 2015), as shown in Figure 4.3.

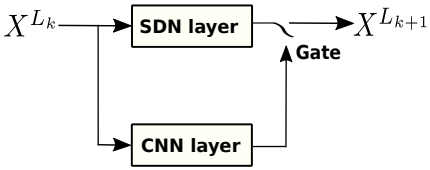


FIGURE 4.3: **Residual SDN layer.**

4.1.3 *Related work*

CONVOLUTIONAL NETWORKS Both spatial dependency and convolutional layers exploit locality (a feature vector is corrected based on its direct neighbors) and equivariance to translation (achieved through parameter sharing). Additionally, SDN enforces spatial coherence. There are also differences in the dependency modeling (depicted in Figures 4.2b and 4.2c). Firstly, when conditioned on the input X^{L_k}, the output feature vectors of a convolutional layer are statistically independent: $X_{i,j}^{L_{k+1}} \perp\!\!\!\perp X_{l,m}^{L_{k+1}} | X^{L_k} \; \forall (i,j) \neq (l,m)$, where i and j are feature map coordinates. In spatial dependency layers, all feature vectors are dependent: $X_{i,j}^{L_{k+1}} \not\perp\!\!\!\perp X_{l,m}^{L_{k+1}} | X^{L_k} \; \forall (i,j) \neq (l,m)$. Secondly, the unconditional dependency neighborhood of a convolutional layer is bounded by the size of its receptive field – a conceptual limitation not present in spatial dependency layers. Hence, spatial dependency layers can model long-range spatial dependencies.

AUTOREGRESSIVE MODELS AND NORMALIZING FLOWS SDN design is inspired by how autoregressive models capture spatial dependencies due to the pixel-by-pixel generation process (Theis and Bethge, 2015). SDN-VAE improves on this is by modeling spatial dependencies in multiple directions. Autoregressive models are inherently uni-directional; the generation order of pixels can be changed but is fixed for training and sampling. Thus in some sense, SDN transfers the ideas from autoregressive to non-autoregressive settings in which there are no ordering constraints. Also, most autoregressive models use teacher forcing during training to avoid sequential computation, but sampling time complexity is quadratic. SDN has linear complexity in both cases and can operate at smaller scales only. Parallel computation of SDN is similar to the one found in PixelRNN (Van Oord et al., 2016), but instantiated in the non-autoregressive setting with no directionality or conditioning constraints.

One can also draw parallels with how autoregressive models describe normalizing flows, for example, IAF (Kingma et al., 2016) and MAF (Papamakarios et al., 2017). In this case, each flow in a stack of flows is described with an autoregressive model that operates in a different direction, or more generally, has a different output ordering. In its standard 4-directional form (Figure 4.1a), SDN creates dependencies between all input and output feature vectors; this renders a full, difficult-to-handle Jacobian matrix. For this reason, SDN is not directly suitable for parameterizing normalizing flows in the same way. In contrast, the main application domain of SDN is rather the parameterization of deterministic, unconstrained mappings.

SELF-ATTENTION The attention mechanism has been one of the most prominent models in the domain of text generation (Vaswani et al., 2017). Recently, it has also been applied to generative image modeling (Parmar et al., 2018; Zhang et al., 2019a). Both SDN and self-attention can model long-range dependencies. The key difference is how this is done. In SDN, only neighboring feature vectors are dependent on each other (see Figures 4.1 and 4.2) hence the spatial locality is exploited. Gated units are used to ensure that the information is propagated across large distances. On the other hand, self-attention requires an additional mechanism to incorporate positional information, but the dependencies between non-neighboring feature vectors are direct. We believe that the two models should be treated as complementary; self-attention excels in capturing long-range dependencies while SDN is better in enforcing spatial coherence, equivariance to

translation, and locality. Note finally that standard self-attention is costly, with quadratic complexity in time and space in the number of pixels, which makes it $\mathcal{O}(N^4)$ in the feature map scale, if implemented naively.

OTHER RELATED WORK SDNs are also notably reminiscent of Multi-Dimensional RNNs (Graves et al., 2007) which were used for image segmentation (Graves et al., 2007; Stollenga et al., 2015) and recognition (Visin et al., 2015), but not in the context of generative modeling. One technical advantage of our work is the linear training time complexity in the feature map scale. Another difference is that SDN uses GRU, which is less expressive than LSTM, but more memory efficient.

4.2 SDN-VAE: A NEW HIERARCHICAL VARIATIONAL AUTOENCODER

In general, autoregressive models are known to be empirically dominant in density estimation of images, with a considerable gap in comparison to the competing methods (Van Oord et al., 2016; Van den Oord et al., 2016; Parmar et al., 2018). This was also discussed in Chapter 1 and Chapter 2. We now show how SDN can be used to create a state-of-the-art hierarchical variational autoencoder that is *de facto* on par with purely autoregressive image models in density estimation. The main advantage of SDN-VAE in comparison to autoregressive models is that it is blessed with all the virtues of a latent-variable models as discussed in Chapter 1 and Chapter 2. The basis of SDN-VAE is the IAF-VAE model (Kingma et al., 2016). Apart from integrating spatial dependency layers, we introduce additional modifications for improved performance, training stability, and reduced memory requirements. The basics of VAEs were already covered in Section 2.3.2. We next summarize IAF-VAE and the related work. We then elaborate on the novelties in the SDN-VAE design.

4.2.1 *Hierarchical variational autoencoders*

A hierarchical (ladder) variational autoencoder is a variant of a variational autoencoder that is characterized by two key components. The first key component are shortcut (skip) connections that allow information to flow from encoder to decoder at different layers of a deep neural network. This

idea was first suggested in the context of vanilla autoencoders in the form of so called *ladder networks* (Valpola, 2015). Shortcut connections allow the encoder to discard some information on each layer where the shortcut is placed. This way, the deepest layers only contain high-level information e.g. the identity of an object, and the low-level information e.g. image texture, is discarded early in the encoding process without sacrificing the reconstruction performance. This strategy has been shown to be beneficial for training very deep autoencoder-based architectures without running into convergence issues. The second key component is to introduce a hierarchy of stochastic latent variables (Burda et al., 2015). In a ladder network-based VAE, one introduces stochastic latent variables at all the layers that contain shortcut connections. This idea was introduced by (Sønderby et al., 2016) and the architecture was named *ladder VAE*. In summary, ladder VAEs (here referred to as hierarchical VAEs) are the combination of ladder networks and VAEs based on hierarchy of stochastic latent variables. A hierarchy of L layers of latent variables can be described as a factorized prior distribution

$$p(\boldsymbol{z}) = p(\boldsymbol{z}_L) \prod_{i=1}^{L-1} p(\boldsymbol{z}_i|\boldsymbol{z}_{>i}) \tag{4.3}$$

On the other hand, there are two ways to specify the inference network in a hierarchical variational autoencoder. In the first variant, only the encoder is used to infer latent variables. It is known as *bottom-up inference* (Burda et al., 2015; Sønderby et al., 2016). The second variant is the one we use in our work and it utilizes both the encoder and the encoder to infer latent variables. It is known as *bi-directional inference* (Kingma et al., 2016). In bi-directional inference, we have

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = q_\phi(\boldsymbol{z}_L|\boldsymbol{x}) \prod_{i=1}^{L-1} q_\phi(\boldsymbol{z}_i|\boldsymbol{z}_{>i}, \boldsymbol{x}) \tag{4.4}$$

The basis of SDN-VAE is IAF-VAE (Kingma et al., 2016). It is illustrated in Figure 4.4. Apart from the bi-directional inference procedure, IAF-VAE additionally introduces *inverse autoregressive flows* to enrich simple isotropic Gaussian-based posterior with a normalizing flow (Rezende et al., 2014). It also uses ResNet block as a residual-based (Srivastava et al., 2015; He et al., 2016) version of the ladder layer. For more details, we refer the reader to the original paper (Kingma et al., 2016).

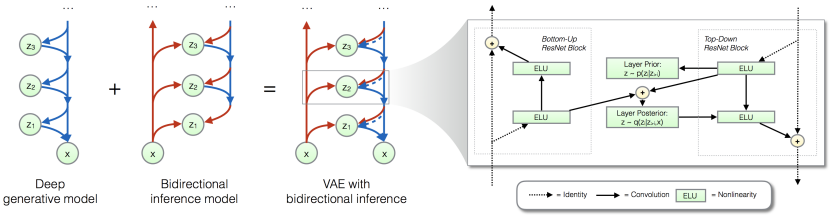


FIGURE 4.4: **Hierarchical IAF-VAE (Kingma et al., 2016).**

OTHER RELATED WORK BIVA (Maaløe et al., 2019) is an extension of IAF-VAE which adds shortcut connections across latent layers, and a stochastic bottom-up path. In our experiments, however, we were not able to improve performance in a similar way. Concurrently to our work, NVAE (Vahdat and Kautz, 2020) reported significant improvements upon the IAF-VAE baseline, by leveraging on many technical advancements including: batch normalization, depth-wise separable convolutions, mixed-precision, spectral normalization and residual parameterization of the approximate posterior. Both methods use the discretized mixture of logistics (Salimans et al., 2017).

4.2.2 *SDN-VAE training and architecture*

Our main contribution to the IAF-VAE architecture is the integration of residual spatial dependency layers, or ResSDN layers (Figure 4.3). We replace the convolutional layers on the top-down (generation) path of the IAF-VAE ResNet block, up to a particular scale: up to 32×32 for images of that resolution, and up to 64×64 for 256×256 images. For the convenience of the reader, we borrowed the scheme from the original paper and clearly marked the modifications (Figure 4.5). Other notable modifications include: *(a)* gated residual instead of a sum residual in the ResNet block (Figure 4.5), for improved training stability; *(b)* more flexible observation model based on a discretized mixture of logistics instead of a discretized logistic distribution, for improved performance; and *(c)* mixed-precision (Micikevicius et al., 2018), which reduced memory requirements thus allowing training with larger batches. Without ResSDN layers, we refer to this architecture as IAF-VAE+.

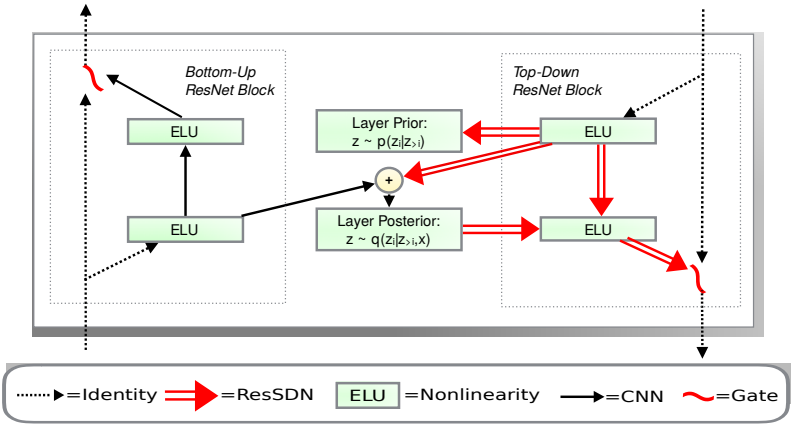


FIGURE 4.5: **SDN-VAE ResNet block as modified IAF-VAE ResNet block from Figure 4.4**. Two notable changes (marked in red) include applying: *(a)* ResSDN layers instead of convolutional layers on the top-down path; *(b)* gated rather than a sum residual.

4.3 MAIN RESULTS

4.3.1 *Density estimation, synthesis and manipulation*

Proposed SDN-VAE and the contributions of spatial dependency layers to its performance were empirically evaluated in terms of: *(a)* density estimation, where SDN-VAE was in a quantitative fashion compared to the convolutional network baseline, and to the related state-of-the-art approaches; *(b)* image synthesis, where learned distribution was analyzed in a qualitative way by sampling from it in different ways; *(c)* image manipulation, where learned distribution was inspected through the manipulation of test images; Exact details of our experiments and additional results in image synthesis are documented in Appendix A.1.1 and A.1.3 respectively.

DENSITY ESTIMATION The basics of density estimation were already covered in Section 2.1. From a set of i.i.d. images $\mathcal{D}_{train} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$, the true probability density function $p(\boldsymbol{x})$ is estimated via a parametric model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. The parameters $\boldsymbol{\theta}$ are learned using the maximum log-likelihood ob-

Type	Method	CIFAR-10	ImageNet32	CelebAHQ256
VAE-based	**SDN-VAE (ours)**	(2.87)	(3.85)	(0.70)
	IAF-VAE+ (ours)	3.05	4.00	0.71
	IAF-VAE (Kingma et al., 2016)	3.11	X	X
	BIVA (Maaløe et al., 2019)	3.08	X	X
	NVAE (Vahdat and Kautz, 2020)	2.91	3.92	(0.70)
Flow-based	GLOW (Kingma and Dhariwal, 2018)	3.35	4.09	1.03
	FLOW++ (Ho et al., 2019)	3.08	3.86	X
	ANF (Huang et al., 2020)	3.05	3.92	0.72
	SurVAE (Nielsen et al., 2020)	3.08	4.00	X
Autoregressive*	PixelRNN (Van Oord et al., 2016)	3.00	3.86	X
	PixelCNN (Van den Oord et al., 2016)	3.03	3.83	X
	PixelCNN++ (Salimans et al., 2017)	2.92	X	X
	PixelSNAIL (Chen et al., 2018)	**2.85**	3.80	X
	SPN (Menick and Kalchbrenner, 2018)	X	3.85	**0.61**
	IT (Parmar et al., 2018)	2.90	**3.77**	X

TABLE 4.1: **Density estimation results.** Negative test log-likelihood is measured in *bits per dimension (BPD)* - lower is better. As in previous works, only the most successful runs are reported. Circled are the best runs among non-autoregressive models and bolded are the best runs overall.
* by autoregressive we refer to the methods based on $p(\boldsymbol{x}) = \prod_i p(\boldsymbol{x_i}|\boldsymbol{x_1}, ..., \boldsymbol{x_{i-1}})$ factorization.

jective: $\boldsymbol{\theta} \leftarrow \arg\max_{\boldsymbol{\theta}} \left[\frac{1}{N} \sum_{i=1}^{N} \log\, p_{\boldsymbol{\theta}}(\vec{x}^{(i)}) \right]$. The test log-likelihood is computed on an isolated set of images $\mathcal{D}_{test} = \{\boldsymbol{x}^{(n)}\}_{k=1}^{K}$, to evaluate learned $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. SDN-VAE and the competing methods were tested on CIFAR-10 (Krizhevsky et al., 2009), ImageNet32 (Van Oord et al., 2016), and CelebAHQ256 (Karras et al., 2017). Quantitative comparison is given in Table 4.1. IAF-VAE+ denotes our technically enhanced implementation of IAF-VAE, but without the SDN modules (recall Section 4.2.2).

ABLATION STUDY Additional ablation study on SDN-VAE was conducted in order to explore in more detail whether the good performance on the density estimation experiments (Table 4.1) indeed comes from the proposed SDN layer. We used CIFAR-10 data on which both baseline and SDN-VAE models converged faster in comparison to two other data sets.

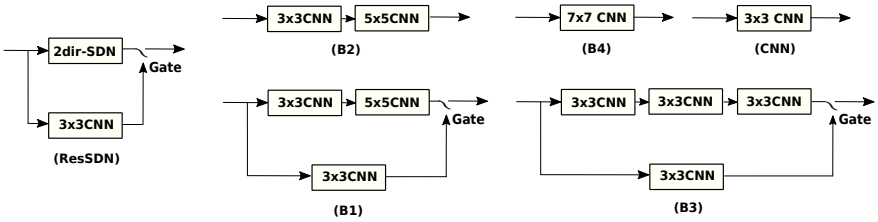


FIGURE 4.6: **Ablation study: ResSDN layer and the baseline blocks.**

Our main motivation was to understand whether the increase in performance comes from our architectural decisions or simply from the increase in the number of parameters. Namely, since the ResSDN layer (from Figure 4.3) is a network with a larger number of parameters in comparison to the baseline 3x3 convolutional layer, it would be justified to question if the increase in performance can be indeed attributed to ResSDN layers. To that end, we designed multiple baseline blocks to replace convolutional layers in the IAF-VAE+ architecture, by replicating the design protocol from Section 4.2.2 in which ResSDN layer was applied to create the SDN-VAE architecture. Baseline blocks are illustrated in Figure 4.6. The main idea is to explore whether possibly increased kernel of a CNN would be sufficient to model spatial dependencies, or whether a deeper network can bring the same performance to the basic VAE architecture. All baseline blocks were trained in the same setting as SDN-VAE i.e. all the parameters were kept fixed, except from the learning rate, which we halved after every 10 unsuccessful attempts to stabilize the training. IAF-VAE based on the CNN block was stable at the default learning rate of 0.002, B2 was stable at the learning rate of 0.001, while B4 and B1 were stable only at 0.0005. B2 was unstable. We also tested ResSDN at a learning rate of 0.001 to understand if there would be a drop in performance, but there wasn't any except from marginally slower convergence time. The best runs in terms of negative ELBO for each of the baseline blocks, along with our most successful run for SDN-VAE architecture are reported in Table 4.2. What we can observe is that the increase in capacity can indeed be correlated with good performance. However, even those VAE models which contained more parameters than the proposed SDN-VAE architecture were not able to converge to a lower negative ELBO. In fact, there exist a considerable performance gap between baseline blocks and the proposed ResSDN layer.

	Layer	Number of VAE parameters in millions	Time to converge in hours	Time to converge in iterations	**Best -ELBO** in BPD
Baselines	CNN	42M	17h	140K	3.081
	B1	192M	21h	75K	3.080
	B2	104M		unstable	
	B3	126M	23h	90K	2.945
	B4	130M	22h	124K	3.120
Ours	ResSDN	121M	45h	60K	**2.906**

TABLE 4.2: **The comparison of ResSDN layer to the baselines from Figure 4.6.** We replaced CNN layers in the IAF-VAE+ architecture (from Section 4.2.2) with baseline blocks and compared the density estimation performance in terms of negative ELBO. The experiments were conducted on CIFAR-10 dataset.

IMAGE SYNTHESIS AND MANIPULATION We additionally inspected the trained $p_{\theta}(\boldsymbol{x})$ by: *(a)* unconditionally generating samples from the prior, at different temperatures (Figure 4.8 left). Lower temperature implies that the sampling is performed with a smaller standard deviation which results in samples that are closer to the prior; *(b)* sampling in the neighborhood of an image not seen during training time (Figure 4.8 right); *(c)* interpolating between two images not seen during training time (Figure 4.7). All technical details on these experiments are in Appendix A.1.1. We also encourage the reader to see additional images that are provided in Appendix A.1.3.

FIGURE 4.7: **Linear interpolation between test images.** Given a pair of images from the test dataset (on the far right and far left), linear interpolation was performed on the layer 5 at a temperature of 0.9 (the sequence above) and on the layer 4 at a temperature of 1.0 (the sequence below).

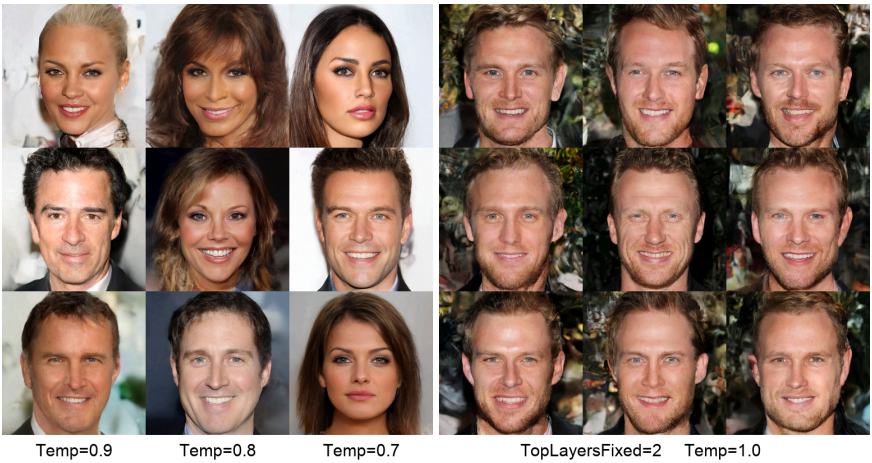


FIGURE 4.8: *(left)* **Unconditional sampling.** Samples were drawn from the SDN-VAE prior at varying temperatures; *(right)* **Sampling around a test image.** Conditioned on the top 2 layers of the latent code of the image in the center, the surrounding images were sampled at a temperature of 1.0.

DISCUSSION The results in Table 4.1 clearly suggest that SDN-VAE is not only superior to the convolutional network baseline, but also to all previous related approaches to non-autoregressive density modeling. The results in Table 4.2 provide additional evidence that SDN is the cause of increased performance, as even convolutional networks with more parameters, increased receptive field or more depth lead to inferior performance. In our image generation experiments we confirmed that SDN-VAE can be successfully scaled to large images. Generated samples are of exceptionally high quality. What is particularly challenging in VAE-based image generation, is sampling at high temperatures, however as shown in Figure A.2 of our Appendix, SDN-VAE can successfully synthesize coherent images even in this case. We strongly encourage the reader to have a look at our figures in Appendix, to better understand how different layers in a hierarchical VAE encode different types of information.

4.3.2 *Learning disentangled representations*

It has been hypothesized that a good representation of data 'disentangles' different factors of variation that correspond to different semantic aspects (Bengio et al., 2013). In a hierarchical VAE, this is done 'vertically' – each stochastic layer encodes different kind of information. Another setting is a simple, non-hierarchical one in which representations are typically disentangled 'horizontally' – each dimension in a latent vector ideally corresponds to a single factor of variation (Higgins et al., 2016). We evaluated the performance of SDN by plugging it into the β-VAE architecture (Higgins et al., 2016) which is a simple variation of the vanilla VAE architecture with an additional hyperparameter β that regulates the weight of the KL-term in a VAE. Higher values of β are found to lead to better disentangled representations, as measured using the corresponding metrics (Locatello et al., 2019). In this section, the gains in performance when using SDN were evaluated with respect to: *(a)* evidence lower bound (ELBO); *(b)* disentanglement of latent codes based on the corresponding metrics, to examine the effects of SDN decoder to the quality of learned latent representations. The aim of the experiments presented in this section is to show that newly introduced SDN architecture can help training better deep generative latent-variable image models in terms of structured representation learning.

EXPERIMENT SETTING A relatively simple VAE architecture with a stack of convolutional layers in both the encoder and decoder (Higgins et al., 2016) was used as a baseline model. SDN decoder was constructed using a single non-residual, one-directional instance of spatial dependency layer placed at the scale 32. The competing methods were evaluated on 3D-Shapes (Burgess and Kim, 2018), a synthetic dataset containing 64×64 images of scenes with rooms and objects of various colors and shapes. Following related literature (Locatello et al., 2019), there was no training-test split, so the reported ELBOs are measured on the training set. The remaining details are in Appendix A.1.4.

TRAINING WITH AN UNMODIFIED VAE OBJECTIVE Using the original VAE objective (Kingma and Welling, 2013), competing architectures were compared in terms of ELBO (Table 4.3). SDN leads to substantially better

VAE	KLD [BPD$\times 10^{-3}$]	**-ELBO** [BPD]
CNN	7.40 ± 0.07	4.44 ± 0.04
SDN	7.95 ± 0.07	$\mathbf{2.66 \pm 0.04}$

TABLE 4.3: **Density estimation on 3D-Shapes.**

estimates which come from better approximation of the conditional log-likelihood term, as shown in more detail in Figure A.4 in Appendix A.1.5.

TRAINING WITH β-VAE OBJECTIVE The same models were trained using β-VAE objective (Higgins et al., 2016). In the β-VAE algorithm, the 'disentanglement' of latent dimensions is controlled by an additional hyperparameter denoted as β, which effectively acts as a regularizer. Modified ELBO reads as: $\mathcal{L}_{\phi,\theta}(X,\beta) = \mathbb{E}_{q_\phi(Z|X)}[\log p_\theta(X|Z)] - \beta \, \mathrm{KL}(q_\phi(Z|X)||p_0(Z))$. To investigate the effects of SDN decoder on the shaping of the latent space, we measured disentanglement for different values of β, for two popular disentanglement metrics: β-VAE (Higgins et al., 2016) and FactorVAE (Kim and Mnih, 2018). The results are shown in Figure 4.9.

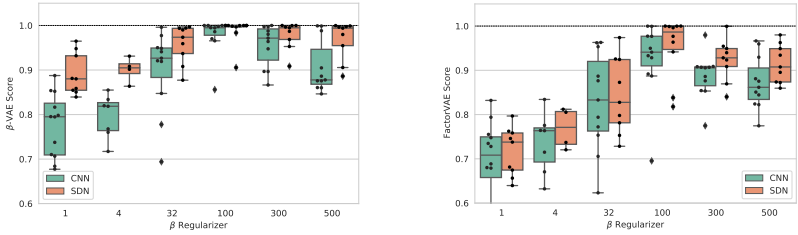


FIGURE 4.9: **β-VAE disentanglement results.** The plots compare the CNN and SDN-based VAE architectures, with respect to the β-VAE (left) and FactorVAE (right) disentanglement metrics.

DISCUSSION The results from this section bring two important conclusions. Firstly, we showed that SDN augmented VAE facilitates factorization of latent codes, offering better disentanglement in comparison to the baseline CNN decoder. Secondly, we confirmed the hypothesis from Chapter 3 that teacher forcing is the main issue when using latent-variable models with powerful decoders. SDN includes autoregressive computations at

each layer of a deep neural network but without utilizing teacher forcing. Thus SDN provides a way to almost match the performance of autoregressive models in density estimation while producing meaningful encoders and latent representations. 'Posterior collapse' in our experiments happened only very early in the training due to local minima, but was easily solved using β annealing (warm-up) procedure (Bowman et al., 2015).

4.4 SDN APPLIED TO MEDICAL IMAGE SEGMENTATION

In Section 4.1, we introduced spatial dependency network as a general building block for any task that requires image generation. In Section 4.2 and Section 4.3, we showed how SDN can be used for constructing powerful VAE decoders. We next explore another application of SDN – medical image segmentation. We show that by augmenting widely used U-Net (Ronneberger et al., 2015) and U-Net++ (Zhou et al., 2019) architectures with SDN, we can get a significant boost in segmentation accuracy. While this section is not central to the main topic of this thesis, it is relevant to provide supplementary evidence that the proposed SDN architecture is indeed well-suited for describing image data.

4.4.1 *Introduction and motivation*

Medical Imaging (MI) enables the in-vivo and non-invasive visualization of structural, molecular, and functional information inside the human body (Xue et al., 2018). Imaging the internal tissues of a patient facilitates the diagnosis, prognosis, and treatment planning (Wang et al., 2016; Gonzalez and Kramer, 2015; Son et al., 2021). MI-driven patient assessment is normally based on the (subjective) opinion of a physician but can be automated e.g. using recent developments in artificial intelligence (Ronneberger et al., 2015) with the potential of accelerating patient management and removing the subjectivity from clinical decision-making (Giger, 2018). Whether the interpretation of MI data is done by doctors or automatic methods, a typical preceding step in the clinical pipeline common to both is to perform image segmentation to reduce the dimensionality of data and highlight regions of interest. Since manual segmentation is a time-consuming and tedious task for physicians, many algorithms have been developed to automate this process. Arguably most successful approaches are based on deep

learning (Goodfellow et al., 2016), utilizing fully convolutional networks (FCN) and encoder-decoder-based convolutional networks such as the U-net (Ronneberger et al., 2015). Extrapolating on their remarkable progress over the past few years (Zhou et al., 2019; Isensee et al., 2018), we speculate that neural network-based algorithms are very likely to match human-level performance in the near future. The hereby proposed approach is a step in that direction.

The performance of automatic medical image segmentation critically depends on the capability of machine learning algorithms to: *(i)* accurately identify intensity discontinuities or edges as object boundaries; and *(ii)* account for global, contextual information when assessing the relevance of different image regions e.g. understanding image-specific semantics and texture. Convolutional neural networks excel at *(i)* but often fail at *(ii)* as examplified in Figure 4.10. We argue that the main reason is the intrinsically limited receptive field of convolutions coupled with the 'shallow' architecture design typical in medical domain. This issue is common to practically all state-of-the-art architectures that are based on the U-Net-like design (Ronneberger et al., 2015). U-Net improves upon the vanilla encoder-decoder design by enabling fusion of textural features across distinct semantic levels; done by introducing skip connections between encoding and decoding paths at different levels and increasing the number of feature channels in the expansive path. However, none of the existing networks in the family of U-Nets is well-equipped to perform non-local similarity comparisons and provide spatially coherent, holistic image segmentation – one of the key characteristics of human-based medical image segmentation. To enable U-Net based architectures to produce globally coherent image segmentation, capturing both long and short-range dependencies in the pixel/voxel space, we propose to endow the original architecture with SDN. In particular, we utilize SDN to construct two new variants of U-Nets: *(i)* spatially dependent nested U-Net (SDNU-Net) – to acquire new state-of-the-art performance in medical image segmentation; and *(ii)* spatially dependent U-Net (SDU-Net) – a faster and less memory-consuming version of *(i)*.

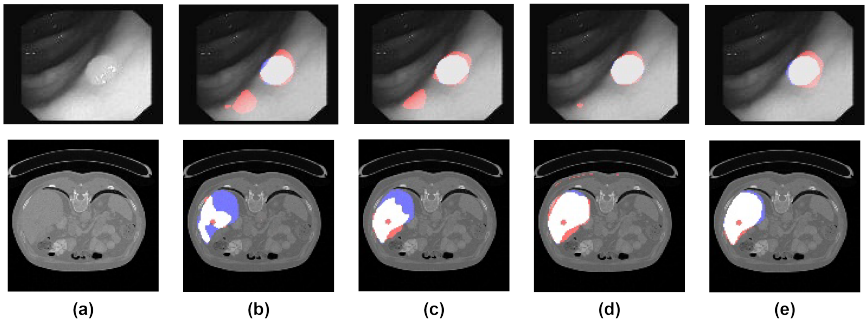


FIGURE 4.10: **Qualitative analysis of image segmentation for different competing methods (best viewed in color). (a)** original image; **(b)** U-Net (Ronneberger et al., 2015); **(c)** U-Net++ (Zhou et al., 2019); **(d)** SDU-Net (ours); **(e)** SDNU-Net (ours); **(top row)** *polyp segmentation*. The baseline U-Net architectures mistakenly identifies a region in the bottom left corner as relevant due to the local contrasts. However, when observing the image as a whole, it is clear that there is a single region of interest, as marked successfully by our networks. **(bottom row)** *liver segmentation*. In contrast to SDN-based networks, the baseline U-Net architectures are unable to coherently identify the well-shaped region of interest. **(red color)** false positives; **(blue color)** false negatives; **(white color)** correctly predicted pixels.

4.4.2 *Spatially dependent U-Nets*

The main contribution of this section is to integrate SDN into the U-Net (Ronneberger et al., 2015) and U-Net++ (Zhou et al., 2019) architectures. This integration is schematically depicted in Figure 4.11. Notice that in the figure we include spatial dependency layers only at lower scales of the U-Net-based architectures. This improves efficiency with little sacrifice in performance. Note that this is aligned to what was previously discussed in Section 4.1.

4.4.3 *Results*

Qualitative comparison is given in Figure 4.10. For quantitative analysis, we evaluated SDN variants of U-Nets on three segmentation tasks

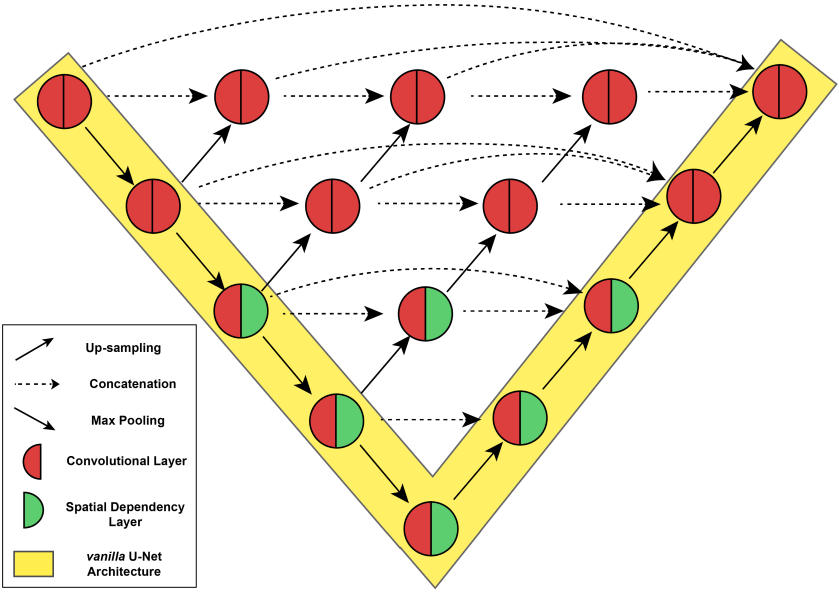


FIGURE 4.11: **SDNU-net.** Integrating spatial dependency layers into U-Net++.

(see Table 4.4). We compared the U-Net architectures with and without SDN layers with respect to Dice and Jaccard score (Table 4.5). All models were trained using a combination of dice and cross-entropy losses following (Isensee et al., 2018). The final model was selected using early-stopping based on the performance on the validation set. The architectural details for the baseline U-Net models were adopted from the original papers (Ronneberger et al., 2015; Zhou et al., 2019). All relevant details regarding SDN specification are documented in Appendix A.1.6.

4.5 CONCLUSIONS

In this chapter, we introduced spatial dependency network for building image generators, and then applied it to construct *(i)* a hierarchical VAE called SDN-VAE that achieves state-of-the-art performance in non-autoregressive modeling; and *(ii)* a non-hierarchical one that offers improved learning of disentangled representations. Crucially, we have shown that powerful deep generative latent-variable models for images can be learned without utilizing teacher forcing, consequently avoiding the repercussions studied

	Number of Images	Image Size (resampled size)	Modality	Challenge
Nuclei	670 (2D images)	96 × 96	Microscopy	2018 Data Science Bowl (Caicedo et al., 2019)
Polyps	612 (29 sequences)	384 × 288 (192 × 144)	Colonoscopy	Endoscopic Vision MICCAI 2015 (Bernal et al., 2017)
Liver	131 (3D volumes)	512 × 512 (128 × 128)	CT	LiTS ISBI 2016/MICCAI 2017 (Bilic et al., 2019)

TABLE 4.4: **Datasets used for medical image segmentation.**

	Nuclei		Polyps		Liver	
Model	**Dice**	**Jaccard**	**Dice**	**Jaccard**	**Dice**	**Jaccard**
U-Net	91.79±0.32	85.52±0.50	75.37±0.60	65.32±0.67	77.44±2.25	69.92±2.19
U-Net++	92.64±0.24	86.88±0.38	76.33±1.69	66.45±1.73	80.24±2.10	73.73±2.46
SDU-Net (ours)	93.25±0.35	87.33±0.58	80.62±0.79	71.04±0.90	82.43±2.24	75.02±2.51
SDNU-Net (ours)	**94.10**±0.36	**88.71**±0.52	**83.31**±1.30	**73.19**±1.55	**85.72**±2.45	**79.37**±2.17

TABLE 4.5: **Image segmentation results, averaged across 5 runs.**

in Chapter 3. Spatial dependency layer is a simple-to-use module that can be easily integrated into any deep neural network, and as demonstrated in the paper, it is favorable to a convolutional layer in multiple VAE settings and also in the context of medical image segmentation. As we will see in Chapter 5, the powerful SDN-VAE architecture can be applied to the semi-supervised learning and anomaly detection. The main downside of SDN remains the computation time. However, we suspect that a more optimized implementation could substantially improve the runtime performance of SDN.

5

IDENTIFYING SLEEP PATTERNS FROM BRAIN SIGNALS

The best thing about being a statistician is that you get to play in everyone's backyard.

— John Tukey

Understanding sleep and its perturbation by environment, mutation, or medication remains a central problem in biomedical research. Its examination in animal and human models rests on brain state analysis of electroencephalographic (EEG) and electromyographic (EMG) signatures. Traditionally, these states are classified by trained human experts via visual inspection of raw EEG/EMG signals. This is a laborious task prone to inter-individual variability. In this chapter, we first present a convolutional neural network (CNN) trained to perform *automatic sleep classification* from time-frequency images derived from EEG/EMG. Secondly, we integrate CNN with a hidden Markov model to *constrain state dynamics based upon known sleep physiology*. Thirdly, we utilize SDN-VAE, the state-of-the-art hierarchical variational autoencoder for images that was presented in Chapter 3, to construct a tool for *detecting anomalous patterns in sleep signals*. We leverage on the ability of SDN-VAE to disentangle high-level from low-level latent features, enabling anomaly detection based only on high-level semantics. The entire framework, which we named *SPINDLE (Sleep Phase Identification with Neural networks for Domain-invariant LEearning)*, was extensively validated on a new dataset and achieved average agreement rates of 99%, 98%, 93%, and 97%. Introduced dataset contains data of four animal cohorts from three independent sleep labs, with scorings from five human experts, rendering roughly 1M labels in total. Finally, we provide to the scientific community free usage of SPINDLE as a web service and benchmarking datasets at `https://sleeplearning.ethz.ch`. At the time of writing, SPINDLE has already processed more than 10.000 sleep recordings submitted by different sleep labs worldwide.

5.1 INTRODUCTION AND MOTIVATION

The importance of sleep in humans and animals is a widely studied and intriguing topic in medical research (Mignot, 2008). Across all organisms with neurons – from Aplysia 'sea slugs' to man – sleep-like states can be identified (Vorster et al., 2014), and in mammalian phyla these basic states share characteristic synchronous neuronal oscillations accompanied by partial or total cessation of motor activity. Until today, electroencephalogram and electromyogram (EEG and EMG) recordings still provide the most accurate data to describe and monitor sleep and wake. At least three major vigilance states can be identified: wake (with low-amplitude high-frequency beta and gamma EEG oscillation between 15-30 Hz and 30-100Hz, as well as extensive EMG activity), non-rapid eye movement sleep (NREM, characterized by large-amplitude delta EEG waves 0.1-4Hz and low or no EMG activity), and rapid eye movement sleep (REM, with predominant theta activity between 6-9Hz and low EMG activity). The relative abundance of these states is governed by both an endogenous 24-hour circadian clock consolidating sleep mostly to day or night, and a sleep homeostat that directs sleep in proportion to the intensity and duration of prior waking experience (Borbély et al., 2016).

Both for the understanding of sleep itself and to study the pathological significance of altered sleep, the identification of individual episodes of sleep and wake across the day based upon EEG/EMG recordings represents a crucial first step. For animals such as rodents EEG/EMG are normally first segmented into 1-8 sec epochs, and are then epoch-by-epoch categorized into the three cardinal vigilance states. Based on the derived scorings, sleep researchers are able to describe the dynamic regulation of sleep, its evolution over time, and the intensity of various EEG oscillations in each vigilance state (its 'spectral composition'). Traditionally, this initial classification has been done manually. However, visual inspection-based sleep scoring is a laborious and ambiguous process that requires constant focus of well-trained human experts. Manual sleep scoring is also highly prone to inter-individual variability, and 90-95% agreement in distinguishing vigilance states is usual across human experts (as our study suggests). This data annotation bias is a potential source of inconsistencies between experimental sleep studies.

In the past years, numerous approaches have been developed with the aim of automating sleep scoring procedures for human and non-human

species. Classical methods transform epochs into hand-designed feature vectors that enable simple discrimination between vigilance states. Manual feature extraction from EEG/EMG is usually based on the domain knowledge already applied in visual sleep scoring. Most popularly, feature vectors are formed from energies of standard frequency bands of EEG power spectrum, such as delta δ(0.14Hz), theta θ(69Hz), sigma σ(1015Hz) and beta β(1530Hz) (Bastianini et al., 2014; Rempe et al., 2015; Kohtoh et al., 2008; Yaghouby et al., 2016), or sometimes more fine-grained binning of the spectrum is applied (Sunagawa et al., 2013; Rytkönen et al., 2011; Dong et al., 2017). Alternatively, features are extracted directly from the temporal domain (Dong et al., 2017). Various machine learning methods are then employed to learn to map derived features to vigilance states i.e. to score sleep. Depending on how these methods utilize annotated data, learning procedures are performed in a supervised (Bastianini et al., 2014; Rytkönen et al., 2011; Rempe et al., 2015; Kohtoh et al., 2008) or unsupervised (Sunagawa et al., 2013; Yaghouby et al., 2016) fashion.

In a wider context, following the revolution of deep learning, many groundbreaking results in automatic speech recognition (ASR) have been obtained (Hinton et al., 2012; Graves et al., 2013; Yu and Deng, 2016; Chan et al., 2016). ASR provides a particularly pertinent analogy to EEG sleep classification: Instead of mapping audio to a sequence of words, one maps EEG/EMG to a sequence of vigilance states. Both problems encounter similar challenges related to subject and environmental changes. While in sleep scoring different animals exhibit different oscillatory activity patterns, in ASR audio patterns differ across speakers due to different accents, noise level, recording device or other voice characteristics. Generalizing ASR and other related domains like image recognition (Krizhevsky et al., 2012a), primarily from harvesting the discriminative power of end-to-end neural network architectures, another class of sleep scoring methods has relatively recently emerged. Neural networks are either trained on top of manually extracted features (Hsu et al., 2013; Dong et al., 2017) or the discriminative features are via end-to-end training learned directly from raw EEG/EMG (Supratak et al., 2017; Längkvist et al., 2012; Sors et al., 2018) or their time-frequency transforms (Bashivan et al., 2015). Commonly used deep architectures include convolutional neural networks (CNNs) (Sors et al., 2018), recurrent neural networks (RNNs) (Hsu et al., 2013; Dong et al., 2017), or the combination of the two (Supratak et al., 2017; Zhao et al., 2017).

Albeit undoubtedly useful to sleep researchers for (semi-)automated sleep scoring, the prediction accuracy of existing methods is still not equal to that of human experts. Even more importantly, current animal sleep scoring solutions have major difficulties to generalize pattern recognition across animals from different experimental settings and labs. Cross-subject variations in EEG/EMG sleep patterns normally originate from experimental differences such as signal-to-noise ratio, EEG derivation and electrode placement, genetic background, drug application, disease models, or differing lab strains and animal species e.g. rats vs. mice (Franken et al., 1998). Due to these signal variabilities, designing robust features by hand is an extremely challenging task. Contrary to approaches attempting to manually construct consistent features, end-to-end architectures have potential to learn robust discriminative features de novo. However, to our knowledge, end-to-end learning frameworks were applied only in the context of human sleep and their applicability was not thoroughly investigated across different experimental domains (Supratak et al., 2017; Längkvist et al., 2012).

In this chapter, we introduce a novel framework for automatic sleep pattern recognition – SPINDLE (Sleep Phase Identification with Neural networks for Domain-invariant LEearning) – by drawing some inspiration from the classical hybrid DNN-HMM models which combine deep neural networks (DNN) and hidden Markov models (HMM) in automatic speech recognition (ASR) (Hinton et al., 2012). The main idea is to leverage on the power of DNNs to learn mapping between EEG/EMG and sleep stages, and the structure of HMMs to constrain implausible sleep transitions e.g. REM to NREM transition which is impossible except from in the exceptional cases – this is similar to DNN-HMM ASR systems that constrain the output space alleviating infeasible language constructions. SPINDLE operates on time-frequency domain treating time-series as images. It uses similar preprocessing procedures to the ones for extracting commonly used Mel-frequency cepstrum (MFCC) features (Yu and Deng, 2016), generalized to multiple heterogeneous channels. Our approach to solving time-frequency fluctuations using a CNN is also motivated by similar modeling ideas in ASR for dealing with speaker and environmental variability (Abdel-Hamid et al., 2014). We demonstrate that SPINDLE provides excellent performance in terms of generalization across different domains: The model was trained only on two wildtype mice, and was then evaluated on 12 mice and 8 rats from four animal cohorts of three independent labs, rendering accuracies of 99%, 98%, 93% and 97% in signal areas where human

experts agreed. When compared to the individual human experts, SPINDLE showed practically equal agreement rate to the one human experts had between themselves, both for artifacts and vigilance states. The second key aspect of SPINDLE is the utilization of SDN-VAE, introduced in Chapter 4, for detecting anomalous EEG/EMG signals. There are two features of SDN-VAE that make it suitable for this task: *(i)* explicitness in density estimation. This makes it possible to detect EEG/EMG that have low likelihood conditioned on the learned density model; *(ii)* hierarchical structure that separates high- from low-level features, such that anomaly detection is performed only based on high-level semantics. For a descriptive explanation of this phenomenon, we refer the reader to Figure A.3 in our Appendix; Furthermore, we disclose a diverse double scored data set (14 mice and 8 rats, annotated by five human experts, rendering 950.400 labels in total) assembled from EEG/EMG recordings produced in three separate sleep labs. The data may be used in the future for benchmarking purposes. Finally, we implement the entire framework in the form of a publicly available free web service for simple and quick classification of EEG/EMG animal recordings: `https://sleeplearning.ethz.ch/`.

An overview of the SPINDLE framework

The SPINDLE method presented here (sketched in Figure 5.1) is designed to achieve high predictive performance preserved across different experimental settings and labs. Its architecture is carefully crafted in an end-to-end fashion around a convolutional neural network (CNN) which operates on top of the preprocessed time-frequency channels of EEG/EMG. We exploit the ability of the CNN to learn highly discriminative and translation-invariant features, as this allows us to remain agnostic to changes in sleep patterns, in both time and frequency dimension. On top of the CNN, a hidden Markov model (HMM) describes vigilance state transition dynamics and suppresses physiologically infeasible vigilance state transitions when applicable. To account for artifacts, SDN-VAE is used to produce binary output, indicating whether certain signal is an artifact or not. The binary signal is combined with the predictions of vigilance states to determine the type of that artifact (steps (d) and (g) in Figure 5.1 respectively). A more detailed explanation is given in Section 5.3 and Section .

SPINDLE was tested on data we produced in three independent sleep labs: BrownLab (`www.sbrownlab.com`), TidisLab (`http://tidis-lab.org/`) and

Cohort	Lab	Wild	Mutant	Specie	S.Rate	EEGs	EMG	Experts	Duration	Artifacts
A	BrownLab	4	0	mice	128Hz	1 frontal, 1 parietal	neck	2	24h	15.2%
B	BrownLab	0	4	mice	128Hz	1 frontal, 1 parietal	neck	2	24h	19.2%
C	BaumannLab	8	0	rats	200Hz	2 parietals	neck	$1+1$	24h	21.3%
D	TidisLab	6	0	mice	512Hz	1 frontal, 1 parietal	neck	2	24h	$\approx 0\%$

TABLE 5.2: **Collected data overview.** Presented are the notable properties of EEG/EMG animal recordings produced in our study. All recordings were segmented into 4 sec time intervals (epochs) and then annotated rendering 21600 $\times$ 2 labels per animal. Table columns for each cohort and lab depict: (a) the number of wildtypes; (b) the number of mutants; (c) rodent specie; (d) the sampling rate of the recording device; (e) the derivation of 2 EEG signals with respect to the placement of corresponding EEG electrodes; (f) the derivation of EMG signal; (g) the number of human experts who scored the data; (h) the duration of each animal recording within given cohort; and lastly (i) the degree of signal corruption taken as the average percentage of artifacts computed from the scorings of the corresponding experts. The cohort C was scored by an expert from BaumannLab, as well as by an expert from BrownLab. All other cohorts were scored by experts from the same lab. Data acquisition is for each animal cohort explained in detail in Materials and Methods.

BaumannLab (http://www.sleep.uzh.ch/en/research-groups/group-baumann.html). The collected data consists of a number of rodent EEG/EMG recordings acquired during sleep studies performed with varying experimental paradigms. The recordings were clustered into four animal cohorts with similar characteristics as summarized in Table 5.2, and evaluated separately.

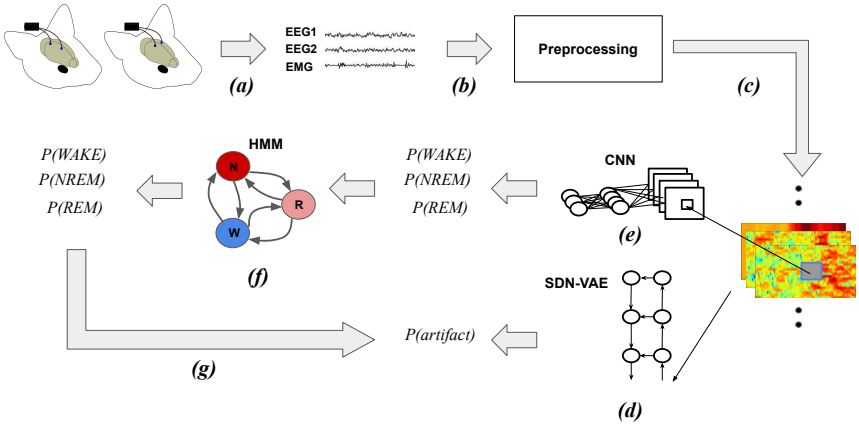


FIGURE 5.1: **Conceptual overview of the SPINDLE framework.** *(a)* Measured EEG activity may vary depending on where the electrodes are placed. Assumed input in our setting are two EEG channels and one EMG channel. EMG signal is recorded on the neck muscle (not depicted for simplicity). *(b)* Raw signals are processed by windowed Fourier transforms applied on overlapping frames, resulting in *(c)* time-frequency representations of EEG/EMG which are additionally preprocessed. Three two-dimensional signals are then sectioned into epochs which correspond to 4 sec intervals. *(d)* Each epoch is independently processed by the two CNNs, the first one estimates whether the evaluated epoch is an artifact *(e)* the second one estimates the probability of each vigilance state. *(f)* The sequence of estimated vigilance state probabilities is then corrected using the Viterbi decoding algorithm and predetermined transition matrix of HMM which encodes the transition rules. *(g)* If an epoch is not designated as an artifact, the most probable vigilance state is assigned. Otherwise, the most probable vigilance state determines the type of the artifact: WAKE-artifact, NREM-artifact or REM-artifact. NREM/N, non-rapid eye movement; REM/R, rapid eye movement; WAKE/W, wakefulness; CNN, convolutional neural network; HMM hidden Markov model.

5.2 RELATED WORK

In our experiments described further below, we compared SPINDLE to three previously reported approaches, using the same data identically split into training and testing sets as already described. Analysis was performed only on the epochs where human experts agreed on the label. Since other algorithms lack dedicated artifact detection and analysis subroutines, corrupted epochs were not taken into account. The following three methods were used as our baselines:

FASTER (SUNAGAWA ET AL., 2013) An unsupervised learning approach which uses nonparametric density estimation clustering on top of manually extracted features. The features in FASTER are derived from a comprehensively binned EEG/EMG power spectra, and are further compressed via principal component analysis. The training data were used for optimizing the hyperparameters according to the procedure described by the authors. Since FASTER was originally applied to 8 seconds long epochs, we down-sampled our scoring resolution from 4 to 8 seconds and then kept only the new larger epochs which contained two equal labels. This way we ensured that FASTER is not at any disadvantage due to the different data annotation setup, furthermore even giving it some advantage by discarding many of the state transition epochs which are hard to score.

SCOPRISM (BASTIANINI ET AL., 2014) SCOPRISM uses two features: (i) the ratio between EEG spectral power of theta θ(6-9Hz) and delta δ(0.5-4Hz) frequency ranges; (ii) the root mean squared error of the EMG signal. The two-dimensional feature space is separated according to threshold values which are learned from data. The sleep scoring of each epoch is further refined, following the results of the scoring draft in adjacent epochs. We optimized the thresholds with respect to the training set and kept them fixed during the evaluation on the testing set. SCOPRISM is originally designed for 4 seconds epochs thus no further parameter adaptation was required.

AUTOSCORE (RYTKÖNEN ET AL., 2011) This method extracts features from power spectra with respect to the logarithmic distribution. Classification in Autoscore is performed using naive Bayes classifier. The feature vectors are smoothed time-wise using a Gaussian convolution to reduce

noise. The method eliminates epochs using a 1-40Hz band-pass filter of the EMG signal. This artifact removal approach produced poor overlap with the artifacts from our data set so it was not further considered. The method was set to produce sleep scoring predictions for 4 second epochs.

Since the competing methods were designed to operate on 1EEG/1EMG setup (rather than 2EEG/1EMG), in our experiments, for each approach we explored the following variations: (a) using only the first EEG signal; (b) using only the second EEG signal; (c) using the average of two EEG signals; (d) averaging the features extracted from two EEG signals; (e) forming the feature vector by combining the features extracted from two EEG signals. In each case, the alternative performing best for the other algorithm was used in our analysis.

5.3 SPINDLE: A FRAMEWORK FOR SLEEP PATTERN ANALYSIS

5.3.1 *Data preprocessing*

The data preprocessing module (step (c) in Figure 5.1) serves primarily to form the input for the convolutional neural networks (CNNs). We subject EEG/EMG signals to a sequence of transformations which enhance the learning process and consequently improve classification performance. The transformations are performed per animal and are meant to diminish non-informative differences in subject specific spectral patterns. The preprocessing procedure is illustrated in Figure 5.2. Both EEG signals are first resampled to the frequency of 128Hz to neutralize the differences in sampling rates coming from different recording devices. Resampled EEGs are then transformed into the time-frequency domain by applying fast Fourier transform on overlapping frames of size 256 (corresponds to 2 seconds) with steps of size 16. Hamming windows were applied to reduce edge effects. Power spectral density (PSD) in time-frequency representation is estimated as squared magnitude of the Fourier transform. Each of the two dimensional spectrograms constructed from EEG signals is treated as a separate feature map on top of which the CNNs convolve. This is analogous to the well-known CNN image classification architectures where the input consists of 3 RGB channels (Krizhevsky et al., 2012a). EEG spectrograms are additionally band-pass filtered (0.5-24Hz), as we experimentally determined that classification performance remains unaffected. Both time-

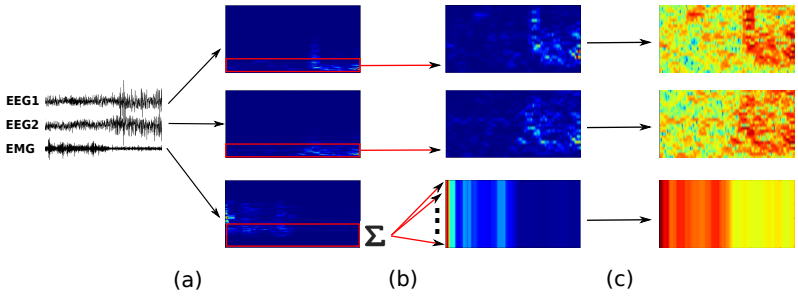


FIGURE 5.2: **Data preprocessing and input preparation.** The figure depicts the creation of the three-channel two-dimensional input for the CNNs. In step (a) raw time series of EEG/EMG signals are separately transformed into the corresponding time-frequency domain (power spectrum density is computed) via a sequence of short Fourier transforms applied to overlapping Hamming windows. In step (b) EEG signals are band-pass filtered $(0.5 - 24Hz)$ and EMG power is integrated over frequency range (0.5-30Hz) resulting in one-dimensional representation of muscle activity change over time. Furthermore, one-dimensional representation of EMG is converted into the two-dimensional one by a multiplication of the signal. Finally, in step (c) the data is log transformed and standardized per frequency component.

frequency channels are then transformed to log scale and finally each channel is per frequency component standardized (zero mean / unit variance).

The EMG signal on the other hand carries the information about muscle activity of evaluated subjects. The total energy of EMG indicates the activity of the corresponding muscle. To decrease the noise we compute signal energy by integrating PSD over a limited frequency band (0.5-30Hz). In other words, we sum up the rows in our time-frequency representation within the given frequency range (see Figure 5.2). This leaves us with one-dimensional signal which measures the change in muscle activity over time. However, in order to form a consistent input for CNN with respect to two-dimensional representations of EEG, we introduce an additional dimension by repeating the signal as illustrated in the figure. This way of forming input is beneficial because in each time instance the CNN filters can relate the total EMG signal power with spectral patterns in different regions of the frequency axis.

5.3.2 *CNN-based sleep classification*

A convolutional neural network (CNN) (Goodfellow et al., 2016) is an artificial neural network most commonly composed of a sequence of (i) convolutional layers which learn high level signal representations; (ii) pooling layers which increase the translational feature invariance; (iii) dense layers which learn high-level feature combinations in a discriminative manner; and (iv) a softmax layer which generates class probabilities. Details of these layers follow below.For a more thorough introduction to CNN we refer the reader to (Goodfellow et al., 2016).

Discriminative features in SPINDLE are learned in an end-to-end fashion from three-channel time-frequency signals (the output in Figure 5.2). The architectural details are given in Figure 5.3. To encapsulate the contextual information the CNN convolves over the target epoch, but also over the surrounding neighbor epochs, two from each side. The variability of spectral profiles (previously discussed in Results) is naturally solved through the discriminative learning of translation invariant features. Namely, by convolving and max-pooling over the frequency domain, the CNN becomes agnostic to small shifts in spectral energy distribution which appear when comparing different animals. On the other hand, by operating over time domain we become agnostic to where the spectral patterns appear within the evaluated region of the input signal.

As usual, weight learning was performed via back-propagation using the Adam optimizer. The weight decay rates of first and second moment were set to 0.9 and 0.999 as suggested in the original paper. Prior to back-propagation, weights were randomly initialized as described in (LeCun et al., 2012). To account for the class imbalance, the optimizer was governed by a class-weighted cross-entropy loss function. Given the observation $\boldsymbol{x}$ and the true class $c \in \{c_1 = WAKE, C_2 = NREM, c_3 = REM\}$ the corresponding loss was calculated as:

$$\text{loss}(c, \boldsymbol{x}) = -w(c) \cdot \log\left(\frac{e^{f(c,\boldsymbol{x})}}{\sum_{i=1}^{3} e^{f(c_i,\boldsymbol{x})}}\right) = -w(c) \cdot \left(f(c, \boldsymbol{x}) - \log\left(\sum_{i=1}^{3} e^{f(c_i,\boldsymbol{x})}\right)\right) \tag{5.1}$$

where $f(c, \boldsymbol{x})$ is the output of the last fully-connected layer corresponding to class c and $w(c)$ is the weight of that class. Class weights $w(c_i)$ were for each mini-batch independently set with respect to the class sample

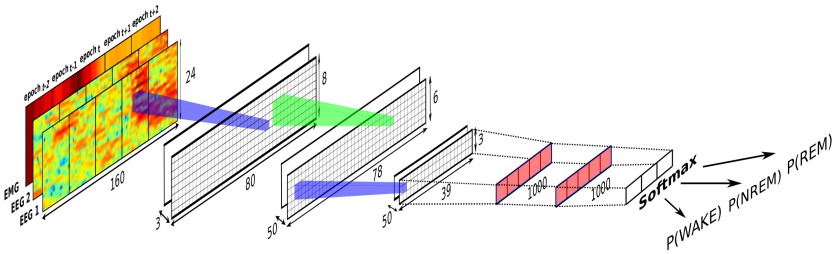


FIGURE 5.3: **Sleep scoring CNN architecture.** Presented are the architectural details of the CNN which estimates the probability distribution over vigilance states for the target epoch_t. Input to the CNN is formed as shown in Figure 5.2. The CNN operates over four neighboring epochs epoch_{t-2}, epoch_{t-1}, epoch_{t+1} and epoch_{t+2} to capture the contextual information. Illustrated CNN consists of two max-pooling layers (depicted in blue), one convolutional layer (depicted in green), and two fully-connected layers (depicted in red). At the very end, a softmax layer outputs class probabilities. The dimensions of the first max-pooling layer are (width, height) = $(2, 3)$ with the corresponding strides $(2, 3)$. The dimensions of the second max-pooling layer are $(2, 2)$ with the corresponding strides $(2, 2)$. The dimensions of the convolutional kernel are $(3, 3)$ with the corresponding strides $(1, 1)$.

ratio within that mini-batch. Final gradient was computed as the normalized sum of individual losses within a mini-batch. Learning rate was set to $5 \cdot 10^{-5}$ and each mini-batch contained M=100 samples. To further regularize the learning procedure we applied the dropout with the probability of 50% to the fully connected layers and allocated 10% of the data from the training set for early stopping. Overall, our CNN contained 6.8M parameters in total, and converged on the held-out data already after 5 full iterations over the training data.

5.3.3 *Constraining sleep transition dynamics with an HMM*

To obtain a finer-grained modeling control over the dynamics of vigilance state transitions, we utilize a hidden Markov model (HMM). Broadly speaking, HMMs are tools for representing probability distributions over se-

quences of indirectly observable states. A first-order HMMs is fully specified by the probability distribution of the initial state $P(s_1)$, the matrix of transition probabilities between neighboring states $P(s_t|s_{t-1})$ and the output model defined by the emission likelihoods $P(\boldsymbol{y_t}|s_t)$ where $\boldsymbol{y_t}$ is the indirect observation of variable s_t. In the context of our problem, the hidden state in some moment t is the vigilance state of the brain $s_t \in \{WAKE, REM, NREM\}$, while the observation $\boldsymbol{y_t}$ is the corresponding region of EEG/EMG signal (in Figure 5.3, that would be $epoch_{t-2}$ to $epoch_{t+2}$) from which hypothetically the state s_t can be inferred. The idea is that, whenever it is known in advance that certain transitions are not valid such as $REM \rightarrow NREM$ and $WAKE \rightarrow REM$ (Benington and Heller, 1994; Borb and Achermann, 1999), which is the case for all the recordings in our data set, we can zero out the corresponding entries in the probability transition matrix of the HMM. Since there is an additional constraint that the rows of the transition matrix must sum up to 1, this leaves us with effectively only four remaining free parameters for which simply specify uniform distribution. These can be set to be of equal value, or tuned additionally to improve the smoothing of the vigilance state sequence estimates. Having specified the model, new posterior probabilities over vigilance states are generated as follows: Using the specified HMM model we can apply Viterbi decoding to find the most probable sequence of vigilance states (the output of step (f) in Figure 5.1). CNN-HMM framework is depicted in Figure 5.4.

5.3.4 *SDN-VAE-based anomaly detection*

Following basically the same procedure described in Section 4.2.2, we trained SDN-VAE probabilistic model to estimate the density of 3-channel EEG/EMG images of dimensionality 160×24 images (shown in Figure 5.3). To make it consistent with images used in Chapter 4, we normalized pixel values to [-1,1] and discretized them into 256 bins. We trained SDN-VAE with non-artifact data from the training set. To perform test-time evaluation of artifactual data, we followed the approach proposed in (Maaløe et al., 2019). The idea is to make binary decisions whether certain EEG/EMG is an artifact or not, only based on top $L - k$ stochastic latent variables which we assume describe high-level statistics. k is regarded as a hyperparameter of the model. L is the number of hierarchical layers. To evaluate the high-level-feature-based density at test time, we utilize the following

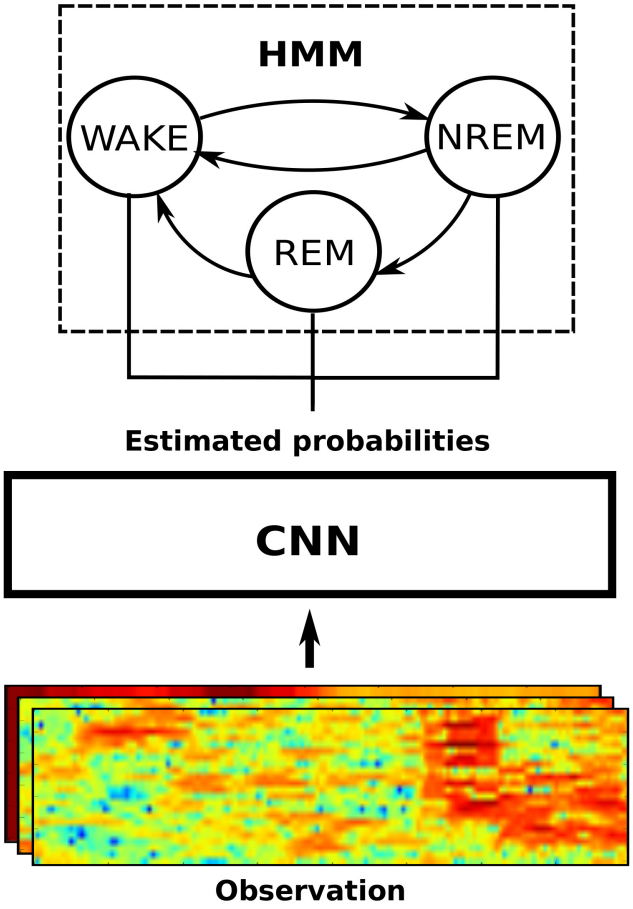


FIGURE 5.4: **CNN-HMM for constraining implausible sleep state transitions.** The figure illustrates how the HMM is used on top of the CNN to enforce prediction sequences which adhere to physiological constraints. In particular, we disallow $REM \rightarrow NREM$ and $WAKE \rightarrow REM$ vigilance state transitions. The constraints are encoded through the transition probability matrix of the HMM, and the observation likelihoods are implicitly calculated by the CNN.

modified ELBO (Maaløe et al., 2019) to approximate the log-likelihood of an EEG/EMG sample $\boldsymbol{x}$:

$$\mathcal{L}^{k}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}_{\leq k}|\boldsymbol{z}_{>k})q_{\boldsymbol{\phi}}(\boldsymbol{z}_{>k}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}_{>k})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{>k}|\boldsymbol{x})}\right] \tag{5.2}$$

where by setting $k = 0$ we obtain the vanilla ELBO from Equation 2.15. In our work, we performed anomaly detection based on top $L - k = 2$ layers. We found this variant of ELBO to be substantially more robust than the vanilla version. The artifact threshold was set based on the artifact labels in the training set, to make a balance between precision and recall. The idea is to set a variant that maximizes the separation between artifact and non-artifact data. Note also that in the SPINDLE web platform described in Section 5.4.6 we allow users to manually adapt this threshold based on their requirements.

5.4 RESULTS

5.4.1 *Exploratory data analysis*

One of the major issues of visual inspection is the intrinsic subjectivity of human experts in data annotation, especially in ambiguous cases when signal patterns do not clearly adhere to the predefined scoring rules. For example, during the transition between vigilance states, it is not always clear where the actual state change occurs. Taking this into consideration is particularly relevant for the validation of an automated sleep scoring method. To this end, we analyzed the inter-expert scoring agreement in evaluation of identical EEG/EMG data (see Figure 5.5). To estimate the coherence between human experts, we first measured their agreement in regions that no expert identified as 'artifacts' – EEG/EMG perturbations related to environmental interference rather than changes in brain state. We then computed the accuracy from the corresponding 3×3 vigilance state submatrices from Fig 5.5. When comparing human experts from the same lab, the estimated agreement rate of sleep scoring in non-artifact data was 95-96%, while the inter-lab agreement was about 90%. On the other hand, the disagreement between human experts in classification of artifacts was notably higher, as the figure indicates. To measure this, we calculated the ratio between the number of epochs marked as corrupted

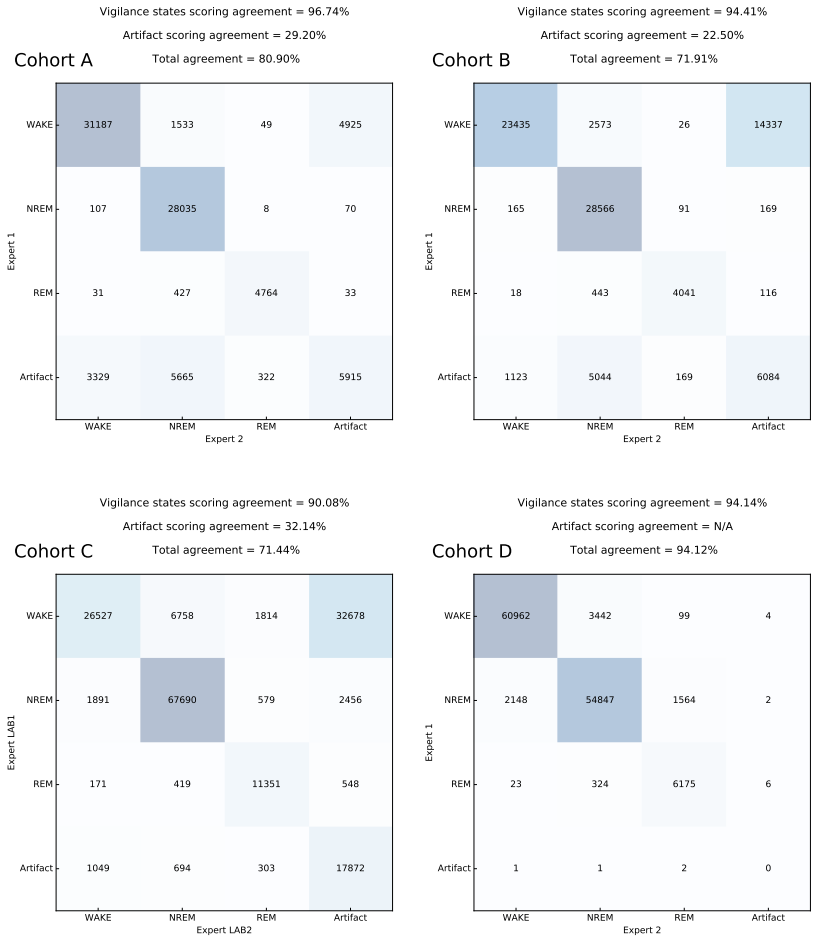


FIGURE 5.5: **Intra-lab and inter-lab human expert agreement:** Confusion matrices derived from the twofold annotation procedure of EEG/EMG data with number of common epochs shown at each intersection, and overall percentage agreement calculated above. We evaluated the agreement of human experts from the same lab (intra-lab agreement), but we also compared the scorings of a BrownLab human expert with the scorings of a BaumannLab human expert on the cohort C (inter-lab agreement). The agreements were computed per-cohort, for non-artifact and artifact data separately, and again when taking all epochs into account.

by both experts and the number of epochs marked as corrupted by at least one of the two experts:

$$artifact\_scoring\_agreement = \frac{|artifact\_intersection(expert_1, expert_2)|}{|artifact\_union(expert_1, expert_2)|} \tag{5.3}$$

These numbers provide rough estimates of the expected accuracy bounds of a hypothetical sleep scoring method comparable to human experts in terms of the predictive performance.

To identify the key obstacles towards robust cross-subject sleep classification, we analyzed the fluctuations of EEG in epochs classified as belonging to the same vigilance state. In the context of our problem, ideally, for each vigilance state we would have signal patterns which are consistent (i) across epochs within the same subject; (ii) across subjects within the same animal cohort; (iii) across subjects from different animal cohorts analyzed under different experimental conditions. To explore the variability across these categories, for each animal and for each vigilance state we separately averaged EEG frequency spectra over all epochs, and then compared these measures within and across cohorts (Fig 5.6). Whether we applied coarse-grained histogram binning according to the commonly used frequency bands (Figure 5.6, middle column) or finer binning (Figure 5.6, right column), spectral energy was differently distributed among frequency bands for different animal cohorts. For example, even though the figure indicates the existence of certain patterns in sleep state signatures i.e. a prominent peak at around 7Hz characteristic of REM sleep, this cannot be simply interpreted as a rule due to high cross-epoch, cross-animal and cross-cohort variabilities (Franken et al., 1998; Tafti et al., 2003). This is arguably the main reason why the classical methods which base their features on energies of different frequency bands of power spectrum do not generalize well. The feature vectors of equal vigilance states are highly inconsistent across subjects, especially if animals have significantly different backgrounds.

To overcome these variations, SPINDLE employs a preprocessing procedure to increase the consistency of spectral patterns within the samples of the same vigilance class. The effects of preprocessing are illustrated in Figure 5.7. On one hand, the log transformation attenuates the discrepancies in magnitudes, and on the other the typical zero mean/unit variance standardization emphasizes the differences between vigilance states. The core characteristic of SPINDLE, however, is its ability to adapt to the variations of sleep state pattern variations in the frequency axis. This flexibility is

achieved through translational invariance, an intrinsic property of CNNs. Whenever the frequency spectrum of evaluated data sample deviates from a hypothetically expected spectral pattern in terms of small shifts of relevant peaks, the CNN absorbs these shifts through the convolutional and max-pooling layers.

5.4.2 *Qualitative analysis*

Before providing a rigorous statistical performance evaluation of SPINDLE, we illustrate its general applicability in Figure 5.8, where we visually compare the scorings of two human experts from different sleep labs with the predictions of our method on identical portion of an EEG/EMG recording from the cohort C. The figure shows that the agreement between the predictions of SPINDLE and the corresponding experts is visually appealing and also sheds light on some common sources of disagreements in the sleep scoring procedure. When vigilance state is constant, the predictions are mainly in agreement, but during the transitions between vigilance states, disagreements are frequent both between human expert scorers and between human and automatically generated scorings. The figure also shows that artifacts are another common source of disagreements. Finally, it is illustrated why time-frequency representation is useful for understanding sleep dynamics i.e. it is easy to notice the correlation between the spectral patterns in the spectrogram and the corresponding vigilance states.

5.4.3 *Quantitative analysis*

In a comprehensive quantitative study we evaluated different performance aspects of SPINDLE. For this purpose, the data set (previously summarized in Table 5.2) was separated into training and testing subsets. The training set consisted of 2 wildtype mice taken from the cohort A, while the validation was performed on the rest of the data i.e. 20 remaining rodents from different labs, strains, and species. Splitting the data set in this way enabled us to test the main premise of this paper: the robustness of our method holds for different experimental settings and labs without any additional model adaptation. By training SPINDLE only on wildtypes we were able to investigate how well it generalizes across the subjects of the same kind (two other wildtypes from the same cohort A), genetically mu-

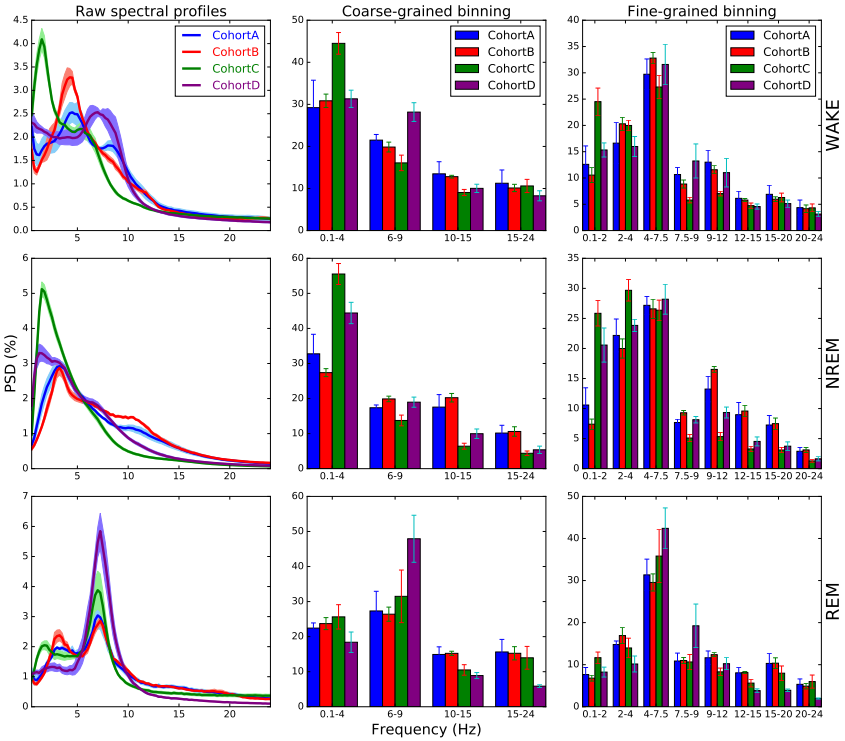


FIGURE 5.6: **Spectral profiles.** For each animal, the averaged frequency spectrum of the EEG recording (its spectral profile) was computed per vigilance class. Each plot in the left column is related to one of the 4 animal cohorts and consists of the mean spectral profile curve and the corresponding standard deviation (a half of it). All curves are normalized relative to the total power of the signal. The middle and the right column respectively represent coarse-grained (following the classical delta, theta, sigma and beta bands) and fine-grained histogram binning applied to the raw spectral profiles, with bars representing summed spectral power in each bin.

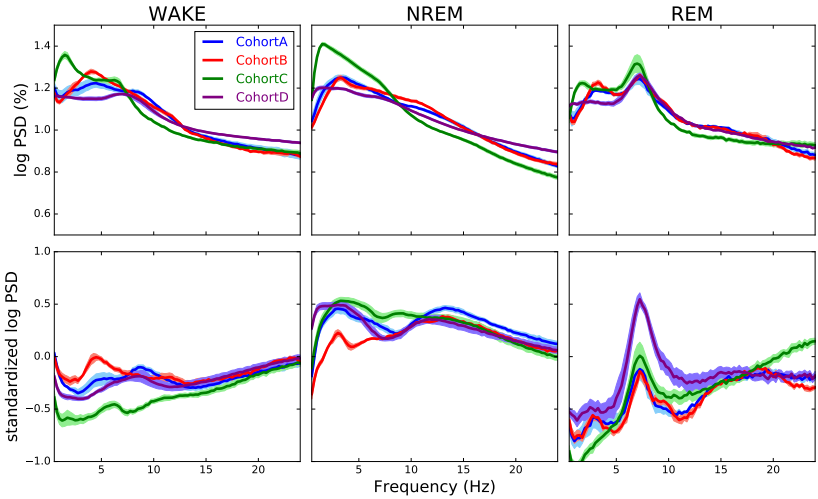


FIGURE 5.7: **Effects of preprocessing to spectral profiles.** Top row shows per vigilance state spectral profiles normalized relative to the total signal power, but only after the log transformation was applied. Relative differences in amplitudes between cohorts are attenuated (compared to Figure 5.6). Bottom row shows log transformed curves after being per frequency component standardized to emphasize the differences between vigilance states.

tated animals (4 mice from the cohort B), different animal species (the rats from the cohort C), and different sleep labs (comparing across cohorts A/B, C, and D).

First, to diminish the effect of subjectivity in manual sleep scoring, we evaluated the predictions against human expert scoring intersection – epochs in which two human experts agree on the label, since all animal recordings were double scored by two individuals. Here, the epochs in which two human experts disagreed did not have any influence upon the performance evaluation. Secondly, to avoid discarding hard-to-score signal regions, we additionally analyzed the performance with respect to the human experts individually. Finally, as we mention above, artifacts represent a major source of difficulty for human expert scorers (recall Fig 5.5). To evaluate SPINDLE more accurately in this respect, we then considered 'clean' regions and artifacts separately.

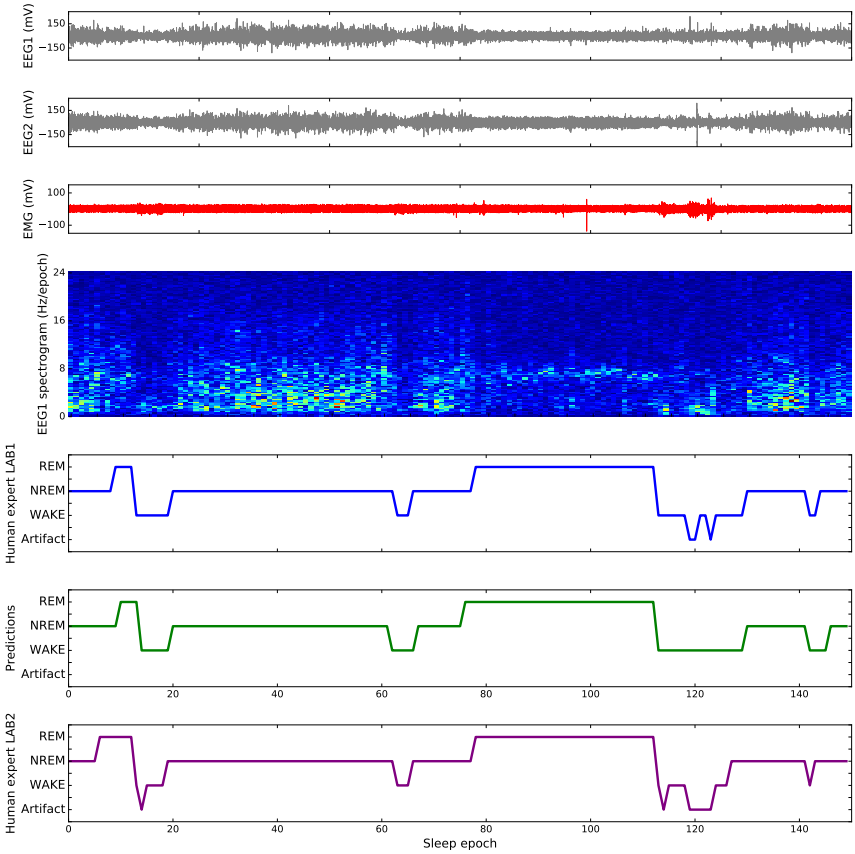


FIGURE 5.8: **Qualitative analysis.** 150 epochs were extracted from an animal from the cohort C, where we found the automated classification to be more challenging, to qualitatively compare the predictions of SPINDLE to the scorings of two experts from different labs. The first three signals from top represent the input, two EEG and an EMG. The spectrogram in the middle is a time-frequency representation of one of the EEG signals. The bottom three plots are the hypnograms, the first one derived from the scorings of one human expert, the second one derived from the predictions of SPINDLE and the third one derived from the scorings of the other human expert. This graph also depicts how artifacts are at times 'smoothed' by our method.

More concretely, we first measured the agreement of vigilance state predictions (given by the output of step (f) in Figure 5.1) with the corresponding human expert scorings only for the epochs not marked as artifacts (by any of the two human experts). To evaluate the quality of artifact detection, the 4-category scorings were re-labeled into non-artifacts and artifacts, and then compared to the corresponding predictions (given by the output of step (d) in Figure 5.1). In addition, we investigated the benefits of applying hidden Markov model (HMM) based post-processing, comparing the predictions of HMM (the output of step (f) in Figure 5.1) against the predictions of the convolutional neural network (CNN) (the output of step (e) in Figure 5.1).In the first set of experiments we quantified the prediction power and the generalization capabilities of SPINDLE by computing the usual classification metrics. In addition, we investigated the benefits of applying the hidden Markov model (HMM) based post-processing. Table 5.5 compares the predictions of HMM (the output of step (f) in Fig 5.1) against the predictions of the convolutional neural network (CNN) (the output of step (e) in Fig 5.1) with respect to the global agreement i.e. the accuracy (AC) and with respect to the precision $PR^{(c)}$, recall $RC^{(c)}$ and F1-score $F1^{(c)}$ for each vigilance class c, in percentages: Evaluations were conducted according to the usual classification metrics, first of overall Accuracy AC, and then for each vigilance state in terms of Precision $PR^{(c)}$, Recall $RC^{(c)}$ and F1-score $F1^{(c)}$ for each vigilance state C, in percentages:

$$\begin{aligned} AC &= \frac{TP + TN}{\#samples} \cdot 100 & PR^{(c)} &= \frac{TP^{(c)}}{TP^{(c)} + FP^{(c)}} \cdot 100 \\ RC^{(c)} &= \frac{TP^{(c)}}{TP^{(c)} + FN^{(c)}} \cdot 100 & F1^{(c)} &= \frac{2 \cdot PR^{(c)} \cdot RC^{(c)}}{PR^{(c)} + RC^{(c)}} \cdot 100 \end{aligned} \tag{5.4}$$

where $TP^{(c)}$, $FP^{(c)}$ and $FN^{(c)}$ are for vigilance class c the numbers of true positives, false positives and false negatives respectively. TP and TN represent the total number of true positives and true negatives.

Table 5.5 demonstrates the predictive performance of SPINDLE with respect to these metrics (excluding artifacts as described above). We compared the predictions of SPINDLE against the scoring intersection of corresponding human experts. The evaluation was performed with and without HMM-based postprocessing which additionally enforced physiological constraints on vigilance state transitions. Although CNN generates impressive performance alone (top rows), HMM leads to additional improvements (bottom rows), most notably in identification of the REM phase

CNN predictions										
		WAKE			NREM			REM		
Cohort	Accuracy	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
A	99 ± 0.5%	100 ± 0.3%	99 ± 0.4%	99 ± 0.3%	99 ± 0.8%	99 ± 0.4%	99 ± 0.6%	98 ± 1.6%	97 ± 2.6%	98 ± 1.2%
B	98 ± 0.3%	98 ± 2.1%	97 ± 1.1%	98 ± 0.7%	97 ± 0.9%	98 ± 1.1%	98 ± 0.1%	96 ± 2.2%	95 ± 1.9%	96 ± 0.9%
C	92 ± 3.2%	80 ± 10.0%	97 ± 1.5%	87 ± 6.0%	99 ± 1.0%	94 ± 3.4%	96 ± 1.8%	86 ± 5.2%	70 ± 16.8%	76 ± 12.2%
D	97 ± 1.3%	98 ± 1.0%	99 ± 0.7%	98 ± 0.6%	97 ± 2.7%	98 ± 1.1%	97 ± 1.2%	96 ± 3.5%	79 ± 23.0%	85 ± 17.7%

HMM predictions										
		WAKE			NREM			REM		
Cohort	Accuracy	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
A	99 ± 0.4%	100 ± 0.3%	99 ± 0.4%	99 ± 0.3%	99 ± 0.8%	99 ± 0.4%	99 ± 0.6%	98 ± 1.5%	97 ± 2.5%	98 ± 1.1%
B	98 ± 0.4%	98 ± 2.1%	97 ± 1.0%	98 ± 0.7%	97 ± 0.9%	98 ± 1.1%	98 ± 0.1%	97 ± 2.4%	96 ± 2.0%	96 ± 0.8%
C	93 ± 3.4%	81 ± 10.6%	98 ± 1.5%	88 ± 6.4%	99 ± 1.0%	94 ± 3.6%	96 ± 1.8%	94 ± 3.0%	75 ± 17.7%	82 ± 13.0%
D	97 ± 1.3%	98 ± 1.0%	99 ± 0.8%	98 ± 0.6%	97 ± 2.8%	98 ± 1.0%	97 ± 1.2%	97 ± 3.3%	81 ± 23.0%	86 ± 17.0%

TABLE 5.5: **Predicting vigilance states - agreement analysis.** The evaluation was performed with and without the application of HMM based post-processing (the outputs of steps (f) and (e) in Figure 5.1 respectively). The predictive power is quantified with respect to the global accuracy, and for each vigilance state separately with respect to the precision, recall and F1-score according to Equation 5.4.

which is usually more challenging to score (Sunagawa et al., 2013). SPINDLE hence showed that injecting sleep domain knowledge into the model may induce very positive effects on the predictive performance. Across all wildtype and mutant mouse species and across labs, overall accuracies exceeded 97%. Across species, overall accuracy remained at 93%, demonstrating the generalization capabilities of SPINDLE.

We also compared the predictions of SPINDLE against the scorings of individual human experts. Doing so enabled us to (i) include into our analysis the epochs in which two experts disagreed; (ii) investigate the potential of SPINDLE to generate predictions which are indistinguishable from the scorings produced by human experts: ideally, the agreement between a human expert and SPINDLE would be close to the agreement between two human experts. The results of the analysis given in Table 5.7 are more than encouraging and show that in terms of the global scoring agreement i.e. the accuracy, SPINDLE is perfectly comparable to human experts. The agreement rate between the predictions of SPINDLE and each expert is close to the agreement rate between two corresponding experts.

	Cohort			
	A	B	C	D
Expert 1 vs Expert 2	96.9%	94.5%	90.0%	94.1%
Expert 1 vs SPINDLE	97.7%	96.0%	89.5%	94.9%
Expert 2 vs SPINDLE	96.7%	93.7%	88.7%	94.5%
Expert intersection vs SPINDLE	99.3%	98.1%	92.8%	97.4%

TABLE 5.7: **Predicting vigilance states - comparison against individual human experts.** The table shows global agreement rate measured by comparing (a) individual experts between themselves; (b) individual experts with SPINDLE; (c) the scoring intersection of two experts with the predictions of SPINDLE. The evaluation was performed on each cohort separately and only non-artifactual epochs were taken into account.

Finally, in a separate set of experiments we evaluated the predictive performance of SPINDLE in identifying artifacts. The subjectivity in distinguishing artifacts from clean epochs is generally known to be overwhelming, and similar conclusions may be derived from Figure 5.5. To ensure a fair performance estimation we computed the agreement rate with each human expert separately, and then compared it to the agreement rate between the two human experts. Table 5.9 suggests that SPINDLE's predictions are, in terms of global agreement as defined by Equation 5.3, de facto equal to that of human experts.

5.4.4 *Comparative analysis*

We now present the comparative analysis in Fig 5.10, where we compare SPINDLE against the methods discussed in Section 5.2. The experiments showed remarkable reduction in both – the error rates and the corresponding standard deviation, which indicates that SPINDLE is much more accurate, and furthermore is more robust than the previous solutions. Particularly appealing is its superiority in detecting REM sleep which is generally considered more difficult to identify. It is also worthwhile considering the computational intensity of the competing methods (bottom right side of Figure 5.10). SPINDLE provides impressive predictive performance in a reasonable time frame.

	Cohort		
	A	B	C
Expert 1 vs Expert 2	29.2%	22.5%	32.1%
Expert 1 vs SPINDLE	28.5%	27.3%	45.3%
Expert 2 vs SPINDLE	39.1%	54.8%	35.9%
Expert intersection vs SPINDLE	45.3%	45.5%	67.1%

TABLE 5.9: **Predicting artifacts - comparison against individual human experts.** The table shows global agreement rate in artifact detection (evaluated with respect to Equation 5.3) by comparing (a) individual experts between themselves; (b) individual experts with SPINDLE; (c) only the epochs marked as artifacts by both experts to the artifact predictions of SPINDLE. Note that the cohort D was omitted since it contained practically no epochs labeled as artifacts.

Furthermore, we analyzed the prediction overlap between SPINDLE and FASTER, to understand their agreement with respect to the ground truth annotations. The analysis is given in Figure 5.9. Note again that in order to ensure a consistent comparison, we down-sampled ground truth and SPINDLE's predictions to 8 second epochs, and discarded epochs which contained: (i) artefacts; (ii) inconsistent scoring between human experts; (iii) label confusion as a product of down-sampling.

5.4.5 *Downstream analysis*

After epoch-by-epoch classification, sleep data is typically pooled across each vigilance state, and then quantitatively evaluated with respect to various parameters of sleep architecture, sleep timing and EEG spectral power. Therefore, we next compared such quantitative outputs when calculated using classifications from SPINDLE or from human scorers. To that end, we analyzed the performance of the downstream analysis applied to the predictions of SPINDLE to investigate its capability to (i) predict major parameters of the sleep architecture; (ii) detect sleep alteration induced by a genetic mutation performed on the cohort B, with respect to the cohort A.

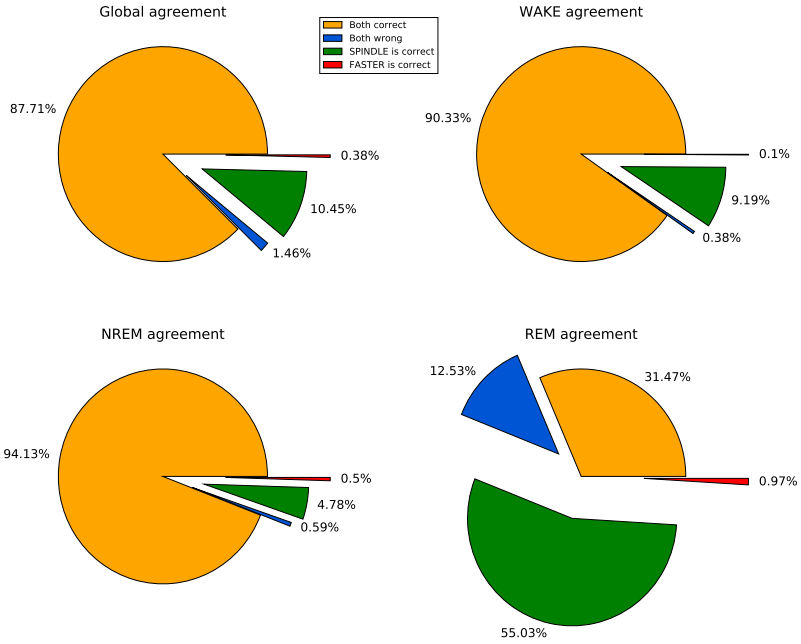


FIGURE 5.9: **Scoring agreement comparison of FASTER and SPINDLE on 8 second intervals.** The overlap was measured with respect to all epochs from all animal cohorts, and additionally with respect to each vigilance state individually.

SLEEP ARCHITECTURE ANALYSIS We computed and then compared the parameters of sleep architecture from all the three sources of vigilance state scorings, for each cohort separately. The goal was to evaluate the predictive power of SPINDLE in terms of several products of the output analysis: the fraction of vigilance states, bout duration, number of bouts and number of transitions. The resulting plots are given in Figure 5.11. Visually, it is clear that in most of the cases SPINDLE produced appealing predictions which represent a good balance between the two corresponding experts, thus potentially enabling less biased analysis. To statistically quantify the discrepancy between the predictions of SPINDLE and the human experts, we performed unpaired Student T-test between each pair of

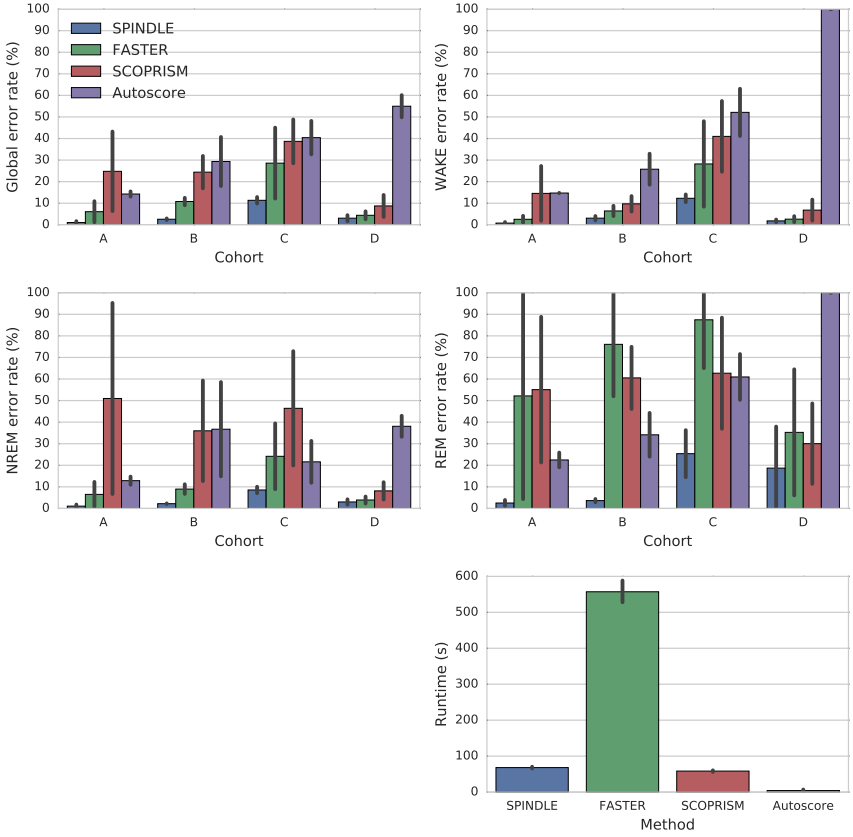


FIGURE 5.10: **Comparative analysis of SPINDLE.** SPINDLE was compared against three other state-of-the-art solutions (FASTER (Sunagawa et al., 2013), SCOPRISM (Bastianini et al., 2014) and Autoscore (Rytkönen et al., 2011)). Evaluations were performed for each animal separately and the results were grouped per cohort (top four figures). The global error rate was measured as $ER = 100 - AC$, and for each vigilance state separately the class specific error rate was measured as $CER^{(C)} = 100 - F1^{(C)}$, where AC and $F1^{(C)}$ are defined in Equation 5.4. The evaluation of errors was performed on the scoring intersection of two human raters and did not take corrupted epochs into account. Execution times for scoring of 24 hour EEG/EMG animal recordings are given at the right bottom figure.

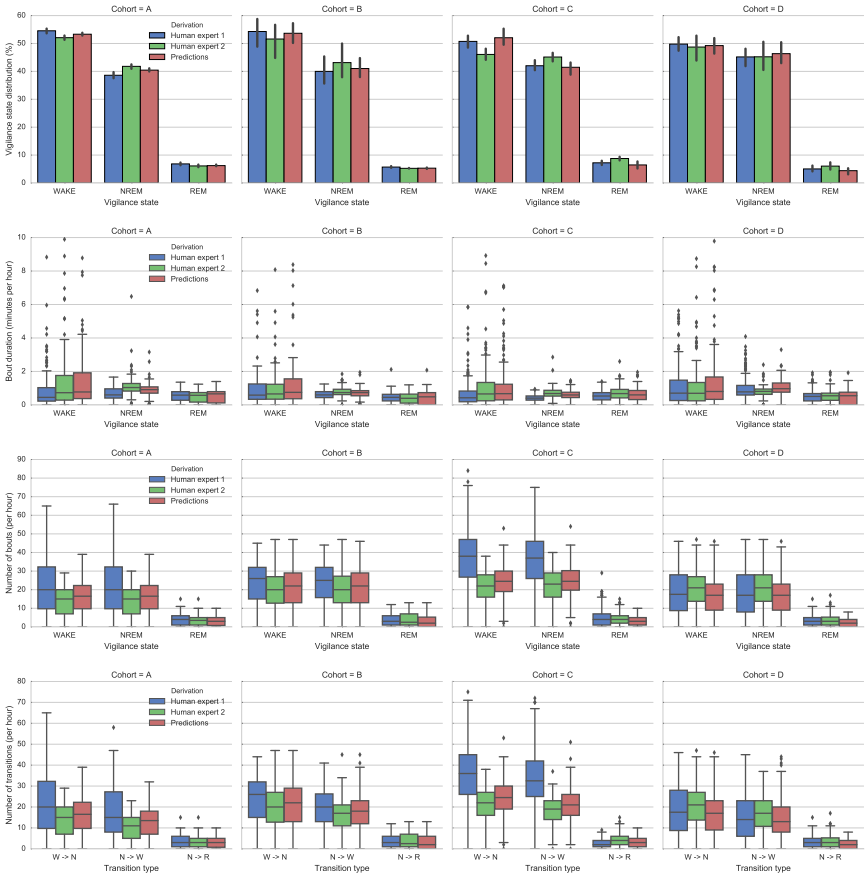


FIGURE 5.11: **Predicting parameters of sleep architecture.** Fraction of sleep (top row), bout duration (second row), number of bouts (third row), and number of sleep transitions (bottom row). Given are the box plots computed from the evaluation of these parameters per hour for each cohort, using data from each human scorer and from SPINDLE output. W→N, transition from WAKE to NREM; N→W, transition from NREM to WAKE; N→R, transitions from NREM to REM.

the corresponding distributions from Figure 5.11, comparing human experts and predictions. Statistical significance level was set at $p < 0.05$. Out of 48 independent output analysis shown in the figure, only in two cases the significance was reported between the predictions and both human experts, while the difference between two experts was non-significant: REM bout duration in cohort D ($p = 0.016$ and $p = 0.004$), and the number of REM bouts in the cohort C (both $p << 0.01$). In all other cases, our predictions were significantly close to at least one of the human experts. Furthermore, there were cases when the output of two human experts was significantly different, while SPINDLE's predictions were not significantly different from any of the two: wake and NREM sleep fractions in the cohort A ($p = 0.011$ and $p = 0.009$), wake bout duration in the cohort C ($p = 0.019$), wake and NREM number of bouts in the cohort B ($p = 0.011$ and $p = 0.032$), and wake to NREM transition and vice versa in the cohort B ($p = 0.013$ and $p = 0.009$). These results clearly demonstrate that SPINDLE has great potential to improve the cross-expert/lab consistency in sleep classification.

DETECTING MUTATION-INDUCED DIFFERENCES BETWEEN COHORTS

We investigated the capability of SPINDLE to detect significant differences in animal sleep patterns induced by experimental factors, thus emulating a real-life study. To that end, we post-processed the scorings of human experts and the predictions of SPINDLE to compare the cohorts A and B. The two cohorts were chosen because the corresponding animals had the same background, the only difference being a mutation of a gene in the mutant cohort B. We calculated the cohort differences with respect to sleep timing (Figure 5.12) and EEG power spectra (Figure 5.13). Both figures suggest that statistically significant discrepancies between cohorts were successfully identified by SPINDLE, when comparing to the ones identified by two human experts. In addition, the figures illustrate the relevance of identifying artifactual data, especially when detecting significant statistical differences in EEG spectral power density cohort profiles. In particular, we performed equal output analysis for two different cut-off values applied on the probabilistic predictions generated by the CNN in step (d) in Figure 5.1. We showed that the strictness of artifact rejection criteria may have influence on designating the areas where the differences between compared cohorts appear. This is a relevant aspect to be considered when aiming for standardized and coherent systematic sleep studies. Since the artifacts are a major source of sleep scoring disagreement among human experts, it is

often hard to find a common ground. However, the probabilistic nature of the SPINDLE framework could be useful in this regard, as it could offer some flexibility in adapting to the experiment-specific artifact rejection criteria.

5.4.6 *SPINDLE web service*

Entire framework described in our study was integrated into a web service which can be found at `https://sleeplearning.ethz.ch`. This service ensures an easy access to the sleep researchers around the world and offers a possibility of accurate evaluation of EEG/EMG recordings in no time. Furthermore, we are continuously improving our framework aiming to create a self-sufficient environment for large scale animal analysis e.g. which would include output analysis in addition. Lastly, we would like to emphasize the two following aspects (i) the adaptable artifact threshold functionality; (ii) further technical considerations.

ADAPTABLE ARTIFACT THRESHOLD Due to the particularly high variations and subject-specific bias in designating epochs corrupted by artifacts, we decided to provide users with a certain flexibility in identifying artifactual epochs. To that end, SPINDLE offers an additional optional functionality which allows users to tune the artifact sensitivity by modifying the threshold from the output of the step (d) in Figure 5.1. A practical example of how this functionality may be utilized was mentioned earlier and is illustrated in Figure 5.12 and Figure 5.13. Note that SPINDLE remains parameter-free, and that this flexibility is introduced to ensure a smoother road towards standardization with respect to the sensitive and not well defined rules for artifact identification.

TECHNICAL CONSIDERATIONS To ensure the successful utilization of the proposed framework we note down several technical aspects. First of all, even though our model was trained on a commonly used 2EEG/1EMG setup (employed in the generation of all the data presented here), we also enable users to use SPINDLE with 1EEG/1EMG experimental setting. We do that by multiplying the corresponding EEG channel. This ensures that the sleep labs with a different recording setup are still able to use SPINDLE.

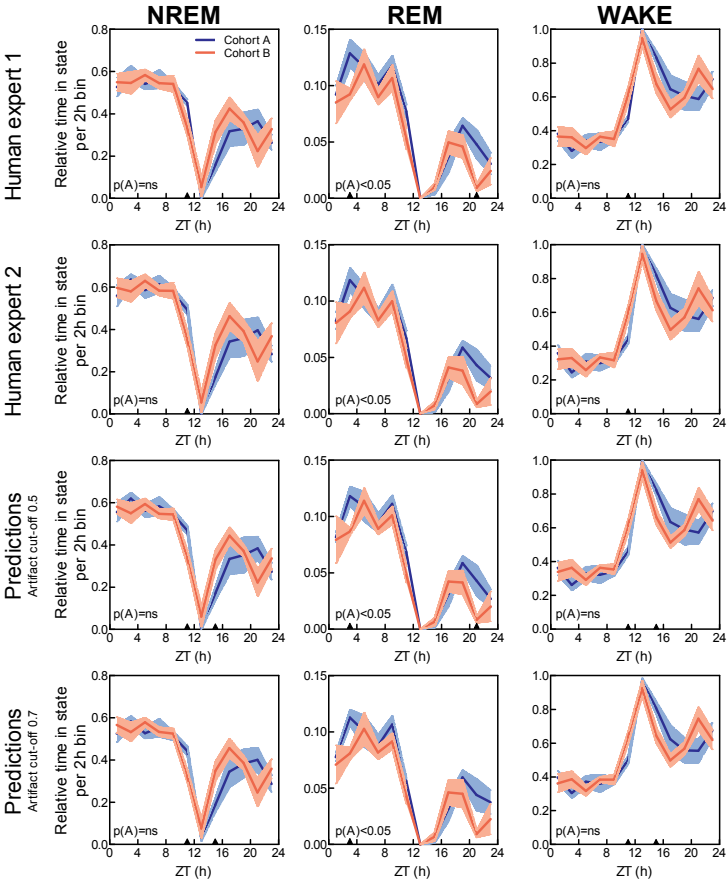


FIGURE 5.12: **Detecting mutation-induced cohort differences in sleep timing.** Fraction of NREM sleep, REM sleep and wakefulness per 2 hour intervals across 24 hours, in the cohorts A (wildtypes) and B (mutants). Results were evaluated from the scorings of the corresponding two experts and SPINDLE. The prediction curves were calculated for two different values of artifact threshold. 0.5 is the default threshold, and indicates that only epochs with $> 50\%$ confidence of being non-corrupted were kept in the analysis. Similar procedure was applied for 0.7 artifact threshold. For measuring of the overall statistical significance, two-way ANOVA (marked as A) was used. $p < 0.05$ regions explain statistical differences of the corresponding 0.5Hz frequency bins measured using a two-tailed T-test with equal variances. The curves represent mean $\pm$ SEM. ZT, zeitgeber time.

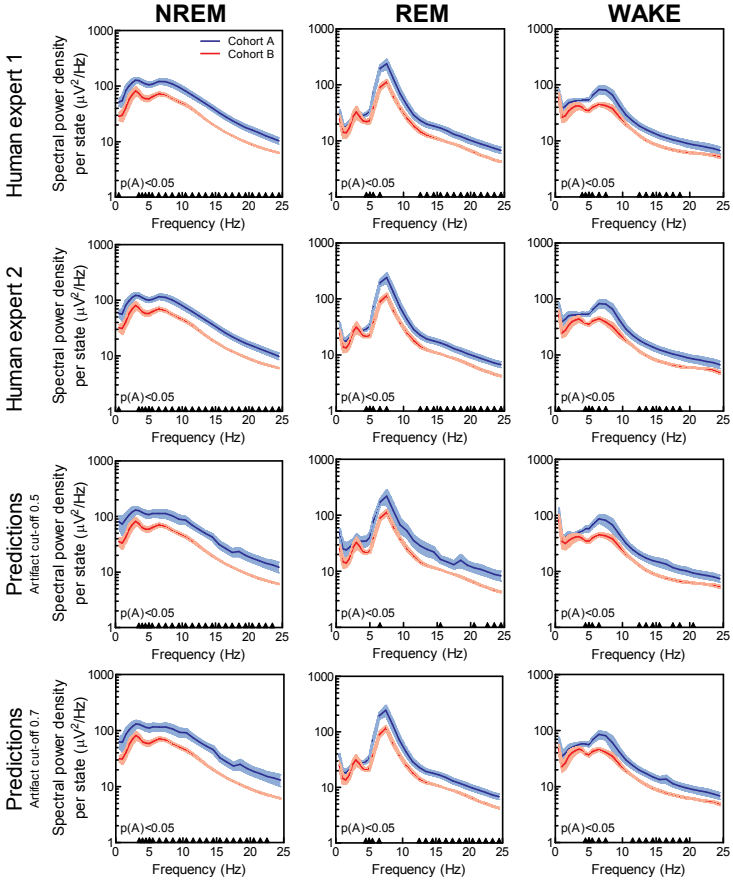


FIGURE 5.13: **Detecting mutation-induced cohort differences in EEG spectra.** EEG spectral power density plots of the cohort A (wildtypes) and B (mutants) during NREM, REM and wakefulness. Results were evaluated from the scorings of the corresponding two experts and SPINDLE. The prediction curves were calculated for two different values of artifact threshold. 0.5 is the default threshold, and indicates that only epochs with $> 50\%$ confidence of being non-corrupted were kept in the analysis. Similar procedure was applied for 0.7 artifact threshold. For measuring of the overall statistical significance, two-way ANOVA (marked as A) was used. $p < 0.05$ regions explain statistical differences of the corresponding 0.5Hz frequency bins measured using a two-tailed T-test with equal variances. The curves represent mean $\pm$ SEM.

It is also worth noting, that data format must adhere precisely to the one described in detail at the above mentioned web platform.

COMPUTATION The programming code underlying the sleep scoring server was written in PyTorch platform and is available through the server. To run the same instance of it locally no special requirements are necessary. We used the server framework for all our experiments. To speed up the learning and evaluation we used vectorized computation on a *GeForce GTX TITAN X* GPU card.

5.5 CONCLUSIONS

In this chapter, we have analyzed a concrete application of machine learning in the domain of sleep biology. We introduced SPINDLE – a framework for automatic analysis of sleep patterns from EEG and EMG signals. The first part of SPINDLE is based on a convolutional neural network and a hidden Markov model, and it is used for automatic classification of animal sleep stages. The second part utilizes SDN-VAE, the deep generative latent-variable model developed in Chapter 4, as a robust tool for anomaly detection in EEG/EMG recordings. The second part is relevant for the main topic of this thesis as it demonstrates how one can leverage on a hierarchy of latent variables and explicitness in density estimation in deep generative models with latent variables. Developed tools are immensely useful to sleep practitioners that need to label massive amounts of data on a daily basis. Furthermore, we have introduced a new dataset with almost 1M labels and integrated everything into a web framework that nowadays serves researchers worldwide, on a daily basis.

6

CONCLUDING REMARKS

In this thesis, we addressed the methods and applications of deep generative modeling with latent variables. The models from this class represent a very appealing approach to unsupervised learning as they provide a unified framework for generative modeling and representation learning. This makes them applicable across a broad spectrum of tasks. Some tasks covered in this thesis include language modeling, image synthesis, image representation learning, and anomaly detection from time-series signals (in our case, brain recordings of sleep). Because we provided detailed remarks at the end of the corresponding chapters, we now provide the reader with some rather general thoughts about the current situation in the very popular field of generative modeling and representation learning.

Our overall concluding impression is that the generality of deep generative models with latent variables is both their advantage and their disadvantage. On one hand, the benefits are obvious: Any model improvement will likely result in improvement on all related downstream tasks e.g. better density estimation will most probably improve representation learning and the plausibility of generated samples. On the other hand, for each of these individual tasks, at the moment of writing, there exist another approach that is less general but performs better than any deep generative model with latent variables. For instance, generative adversarial networks (Goodfellow et al., 2014) are still state-of-the-art approach to realistic image synthesis. Methods based on contrastive learning (Chen et al., 2020) are at the moment of writing top-performing approach to image representation learning. Autoregressive models (Brown et al., 2020) are by far the most widely used solution for language modeling, providing top-notch performance in terms of density estimation.

Even so, some recent work such as the one presented here, give promise that there remains lots of unexploited opportunities for advancing deep generative models with latent variables further. This thesis can certainly be seen as a step in that direction.

A

APPENDIX

A.1 APPENDIX TO CHAPTER 4

A.1.1 *Experimental details for SDN-VAE*

Full experimental details are given in Table A.1. Due to high computational demands, we only modestly optimized hyperparameters. In principle, our strategy was to scale up: (a) the number of deterministic and SDN filters; (b) batch size; (c) learning rate; (d) the number of spatial directions and (e) the number of DML components; as long as we found no signs of overfitting, had no memory or stability issues, and the training was not considerably slower. EMA coefficient values were taken from the previous related works (Kingma et al., 2016; Chen et al., 2018) – we tested 0.999 and 0.9995. We also swept through the values $\{2, 4, 8, 15\}$ for the number of latent stochastic filters, but saw no significant difference in the performance. Most extensively we explored the number of layers per scale, and found this to have relevant impact on runtime and over/underfitting, for both baseline and our architecture. We found that more downsampling improved runtime and reduced memory consumption, but made the tested models prone to overfitting.

A.1.2 *Computational considerations*

NUMBER OF PARAMETERS For a filter size of 200 (a value used in the SDN-VAE experiments), we compare the number of parameters between CNN and SDN layers. Note that the input scale does not have any effect on the resulting numbers. The numbers are given in Table A.2. Here 'dir' denotes the directions in SDN. 'Project phase' denotes the size of project-in and project-out SDN sub-layers which are in this experiment of the same size, since no upsampling is performed. We can observe that the 2-

	CIFAR10	ImageNet32	CelebAHQ256
	$32^2 = 32\text{x}32$	32^2	256^2
# training samples	50K	1.281M	27K
# test samples	10K	50K	3K
Quantization	8 bits	8 bits	5 bits
Encoder/decoder depth	15	16	17
Deterministic #channels per layer	200	200	200
Stochastic #channels per layer	4	4	8
Scales	$8^2, 16^2, 32^2$	$4^2, 8^2, 16^2, 32^2$	$4^2, 8^2, 16^2, 32^2, 64^2, 128^2, 256^2$
SDN #channels per scale	120, 240, 260	424 for all	360, 360, 360, 360, 360, ~~SDN~~, ~~SDN~~
Layers per scale	5 for all	4 for all	2,2,2,2,2,3,4
Number of directions per SDN	2	3	2
Optimizer (Kingma and Ba, 2014)	Adamax	Adamax	Adamax
Learning rate	0.002	0.002	0.001
Learning rate annealing	Exponential	Exponential	Exponential
Batch size per GPU	32	32	4
Number of GPUs	8	8	8
GPU type	TeslaV100	TeslaV100	TeslaV100
GPU memory	32GB	32GB	32GB
Prior model	Gaussian	Gaussian	Gaussian
Posterior model	Gaussian	Gaussian	Gaussian
Posterior flow	1 IAF	1 IAF	1 IAF
Observation model (Salimans et al., 2017)	DML	DML	DML
DML Mixture components	5	10	30
Exponential Moving Average (EMA)	0.999	0.9995	0.999
Free bits	0.01	0.01	0.01
Importance samples (Burda et al., 2015)	1024	1024	1024
Mixed-precision (Micikevicius et al., 2018)	Yes	Yes	Yes
Weight-norm (Salimans and Kingma, 2016)	Yes	Yes	Yes
Horizontal flip data augmentation	Yes	No	Yes
Training time	45h	200h	90h

TABLE A.1: **Experimental configurations for the density estimation tests.** ~~SDN~~ means that SDN was not applied at the corresponding scale. IAF contained 2 layers of masked 3x3 convolutional networks, with the context and number of CNN filters both of size 100.

directional SDN is approximately of the same size as 5x5CNN in terms of number of free parameters.

	3x3CNN	5x5CNN	Project phase	SDN cell	1dir-SDN	2dir-SDN
# parameters	360200	1000200	40200	481200	561600	1042800

TABLE A.2: **Number of parameters of different neural layers.**

RUNTIME UNIT TESTS We measured the execution times of forward propagation of CNN and SDN layers, for different input scales. To obtain standard deviation estimates, each forward propagation was repeated 100 times. The batch size was set to 128. The results are given in Table A.3. The SDN layer is considerably slower than the CNN layer, but in the reasonable limits. Note that more efficient implementation will likely improve SDN runtime performance.

Input scale	3x3CNN	5x5CNN	1dir-SDN	2dir-SDN
4	0.32±0.01	1.26±0.01	1.93±0.06	3.52±0.07
8	0.63±0.01	3.25±0.13	4.09±0.06	7.61±0.09
16	2.33±0.08	11.2±0.08	11.4±0.03	21.0±0.05
32	9.18±0.16	20.4±0.18	45.4±0.09	83.8±0.15

TABLE A.3: **Runtime unit tests.** The execution time of forward propagation in ms.

A.1.3 *More results for image synthesis and manipulation*

FIGURE A.1: **MSE-based nearest neighbors.** On the left, shown are images from Figure 4.8. On the right, shown are nearest neighbors in terms of mean squarred error (MSE), found in the training data set. There are no signs that generated samples are replicas of the training ones.

FIGURE A.2: **Additional samples synthesized at different temperatures.** As we decrease the temperature, getting closer to the mean of the prior, the photographs become smoother i.e. more 'generic'.

FIGURE A.3: **Sampling around a test image, for varying temperatures and number of fixed layers.** Presented is a 3x3 grid. In each grid there is a 3x3 sub-grid in which an image in the center is encoded, and the surrounding images are sampled conditioned on the specified number of layers from the encoded image and at the specified temperature. The text on the left hand side denotes the number of layers taken from an encoded image to condition sampling on. The text on the bottom denotes the temperature at which samples were drawn. Two key observations can be made: *(a)* lowering the temperature decreases the level of details, making photographs smoother; *(b)* Most of the information in SDN-VAE is encoded in the topmost layer. As we go further down the chain of the generator network, there is a gradual decrease in information content in latent stochastic variables.

A.1.4 *Experimental details for learning disentangled representations*

The hyperparameter configuration is given in Table A.4. In principle, we followed (Higgins et al., 2016; Locatello et al., 2019) in configuring parameters. Notable changes include using of dicretized logistics (Kingma et al., 2016) as the observation model and the Adamax optimizer. We also applied β-VAE annealing to avoid posterior collapse early in the training, an issue for both considered architectures. The base architecture was again taken from the same previous works. The exact description of architectures is given in Table A.5 for the CNN-based VAE, and in Table A.6 for the SDN-based VAE.

	3D Shapes $64^2 = 64 \times 64$
# data samples	448K
SDN #channels	200
Number of directions per SDN	1
Optimizer	Adamax
Learning rate	0.001
Batch size	128
Total number of training iterations	200k
Prior and posterior models	Gaussian
Observation model	Discretized Logistics

TABLE A.4: **Configuration of disentangled representation learning experiments.**

A.1.5 *More results for learning disentangled representations*

For the sake of completeness, we also provide learning curves for varying values of β and for different individual terms of β-VAE objective function. The plots are shown in Figure A.4.

Encoder	Decoder
4×4 conv, 32 ReLU, stride 2	FC, 256 ReLU
4×4 conv, 32 ReLU, stride 2	FC, $4 \times 4 \times 256$ ReLU
4×4 conv, 64 ReLU, stride 2	4×4 upconv, 64 ReLU, stride 2
4×4 conv, 64 ReLU, stride 2	4×4 upconv, 32 ReLU, stride 2
FC 256	4×4 upconv, 32 ReLU, stride 2
FC 2×10	4×4 upconv, 3 (number of channels) , stride 2

TABLE A.5: **CNN-based vanilla VAE architecture.**

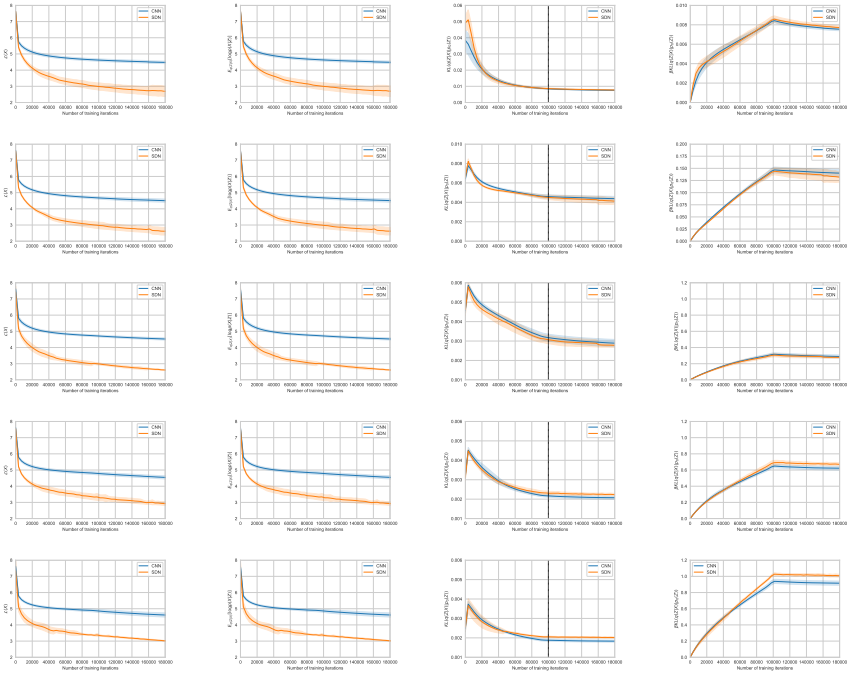
Encoder	Decoder
4×4 conv, 32 ReLU, stride 2	FC, 256 ReLU
4×4 conv, 32 ReLU, stride 2	FC, $4 \times 4 \times 256$ ReLU
4×4 conv, 64 ReLU, stride 2	4×4 upconv, 64 ReLU, stride 2
4×4 conv, 64 ReLU, stride 2	4×4 upconv, 32 ReLU, stride 2
FC 256	1dir-SDN with 200 channels
FC 2×10	4×4 upconv, 3 (number of channels) , stride 2

TABLE A.6: **SDN-based vanilla VAE architecture.**

A.1.6 *Experimental details for medical image segmentation*

	SDU-Net			SDNU-Net		
Segmentation task	**nuclei**	**polyp**	**liver**	**nuclei**	**polyp**	**liver**
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Learning rate	1,00E-03	1,00E-04	1,00E-04	1,00E-03	1,00E-04	1,00E-04
Convolutional layer kernel size	3x3	3x3	3x3	3x3	3x3	3x3
Activation function	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU
Batch Normalization	Yes	Yes	Yes	Yes	Yes	Yes
Number of layers per scale	2	2	2	2	2	2
Residual Connections	No	Yes	Yes	No	Yes	Yes
SDN kernel size	3x3	3x3	3x3	3x3	3x3	3x3
SDN #channels per scale	150	150	150	150	150	150
SDN layers per encoder/decoder	2	1	1	2	1	1
Number of directions per SDN	2	2	2	2	2	2
Batch sizer per GPU	20	20	16	20	20	16
Number of GPUs	4	4	4	4	4	4
GPU memory (GB)	11	11	11	11	11	11

TABLE A.7: **Experimental configuration for medical image segmentation.**



(a) (b) (c) (d)

FIGURE A.4: **β-VAE learning curves for CNN and SDN-based VAE architectures.** Presented are the following quantities measured over the course of training: *(a)* evidence lower bound (ELBO); *(b)* conditional log-likelihood (reconstruction) term; *(c)* KL divergence term; *(d)* β-scaled KL divergence term; Top-to-bottom, the rows are related to $\beta = \{1, 32, 100, 300, 500\}$. Each pair ($\beta$ value, VAE architecture) is trained for 10 different random seeds. Linear β-annealing procedure was performed, where β was increased from 0 to its final value across the span of 100K training iterations (the end of the annealing procedure is denoted by the vertical line in the plots of the column (c)).

A.2 APPENDIX TO CHAPTER 5

A.2.1 *Experimental data acquisition*

We detail the steps in collecting the data previously summarized in Table 5.2. In total, 4 animal cohorts containing 22 animals were acquired from 3 independent sleep labs. Animal studies were performed by authorized researchers according to all applicable laws and regulations of the cantons of Zürich (BrownLab, BaumannLab) and Bern (TidisLab), and were each approved by the relevant cantonal authorities. Each sleep recording consists of a pair of EEG signals and an EMG signal simultaneously recorded. Manual labeling of wake-sleep states based on EEG/EMG signals was performed by trained experts from the corresponding sleep labs on all consecutive 4second epochs. Raw EEG traces were visually inspected offline and scored in three vigilance states, wakefulness, NREM sleep and REM sleep. Wakefulness was defined based on increased EMG activity for more than 50% of the epoch duration. NREM sleep was defined by reduced EMG activity and increased EEG power in $<$ 4Hz frequency ranges. REM sleep was characterized based on low EEG power in $>$ 4Hz oscillations and high EEG power in 6-9Hz frequency bands and intermediate muscle tone. Unclear stages or technical artifacts were excluded and subsequently labeled as artifacts.

BROWNLAB DATA SET It contains sleep recordings of two cohorts: the cohort A containing 4 wildtype mice and the cohort B containing 4 mice with a genetic mutation. All mice were kept in Macrolon cages (36x20x35cm) with food and water ad libitum, maintained at a 12 hour light-dark cycle (light onset 07.00 AM) in normal cages prior to surgery, and then in open-top cages with counterbalanced swivel-attached cables during and between sleep recordings. For the EEG recordings, mice were implanted epidurally with gold-plated miniature screws (0.9mm diameter) under constant isofluran inhalation anesthesia. Analgesia was given i.p. at 0.1mg/kg during the surgery. The coordinates of EEG electrodes were as follows; frontal derivation was placed 1mm anterior to bregma, 1.5mm lateral to mid-line, the parietal derivation was placed 3mm posterior to bregma and 2mm lateral to mid-line. The reference was placed over the cerebellum (2mm posterior to lambda on the midline). Two gold wires (0.2mm diameter) were inserted bilaterally in the neck muscle for EMG recordings. The

screws were connected to stainless steel wires and fixed on the skull with acrylic dental concrete. The mice were tethered to a swivel throughout the entire experiment and were allowed to recover for 4 to 7 days before any further handling. The EEG and EMG signals were amplified (amplification factor, 2000), filtered (highpass filter: –3 dB at 0.016 Hz; low-pass filter: –3 dB at 40 Hz) sampled with 512 Hz, digitally filtered [EEG: low-pass finite impulse response (FIR) filter, 25 Hz; EMG: bandpass FIR filter, 20–50 Hz or 10-30Hz], and stored with a resolution of 128 Hz. Before each recording, the EEG and EMG channels were calibrated with a 10 Hz, 300 V peak-to-peak sine wave.

BAUMANNLAB DATA SET It contains sleep recordings of the cohort C which consists of 8 rats. The rats were implanted with EEG/EMG electrodes for recording of vigilance states using a protocol slightly modified from (Baumann et al., 2006). Briefly, four stainless steel miniature screws (Hasler, Switzerland), one pair for each hemisphere, were inserted bilaterally into the rats' skull following specific stereotactic coordinates: the anterior electrodes were implanted 3mm posterior to bregma and 2mm lateral to the midline, and the posterior electrodes 6mm posterior to bregma and 2mm lateral to the midline. For monitoring of muscle tone, a pair of gold wires served as EMG electrodes and was inserted into the rats' neck muscle. All electrodes were connected to stainless steel wires, further connected to a head piece (Farnell, #M80-8540842, Switzerland) and fixed to the skull with dental cement. Bilateral 24 hour EEG/EMG signals were recorded in freely-moving rats. For this purpose, the animals were transferred to special recording cages with food and water available ad libitum, where they had an adaptation period of two days before recordings took place. EEG and EMG were sampled at 200 Hz, signals were amplified, filtered and converted into analog-to-digital signals. Hardware EMBLA and Somnologica-3 software (Medcare Flaga, Iceland) were used. Activity in the 50-Hz band was discarded from the analysis because of power line artifacts.

TIDISLAB DATA SET It contains sleep recordings of the cohort D. *C57Bl6 mice* were used, 11-14 weeks of age, from Charles Rivers Laboratories, Germany. Animals were housed in individual custom-designed polycarbonate cages at constant temperature (22 ± 1 °C), humidity (30–40%), and circadian cycle (12 hour light-dark cycle, lights on at 08:00). Food and water

were available ad libitum. Animals were treated according to protocols and guidelines approved by the Veterinary office of the Canton of Bern, Switzerland (License number BE 113/13). Animals were housed in IVC cages in groups of 2–5 before instrumentation. After implantation, all mice were housed individually. Animals were habituated to the recording cable in their open-top home cages (300 × 170 mm). Animals were anaesthetized in isoflurane in oxygen and mounted in a stereotaxic frame. Saline 10ml/kg and meloxicam 5mg/kg were given subcutaneously. The skin on the head was shaved and aseptically prepared, and lidocaine 2mg/kg infused subcutaneously at the incision site. A single longitudinal midline incision was made from the level of the lateral canthus of the eyes to the lambda skull suture. Two stainless steel screws were placed in the skull over the parietal cortex to measure EEGs and two bare-ended wires sutured to the trapezius muscle of the neck to record EMG. The implant was stabilized using a methyl methacrylate cement and the animal allowed to recover in the home cage on top of a heating mat. Animals were allowed a minimum of 5 days to recover before starting recordings. Habituation to the cables was performed up to 8 hours per day until the animals had nested and resumed a normal sleep-wake cycle. For recordings, mice were connected to the AM system and data sampled at 512 Hz. For each mouse, 24 hour baseline sleep was recorded, while animal was allowed to move freely in the cage. Recorded EEG/EMG signals were down-sampled to 128 Hz, after applying a low pass filter (Chebyshev Type I, order 8, low pass edge frequency of 50 Hz, passband ripple of 0.05 dB) to prevent aliasing.

BIBLIOGRAPHY

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22 (10):1533–1545, 2014.

Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2897–2905, 2018.

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. In *International Conference on Learning Representations*, 2018.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

Pouya Bashivan, Irina Rish, Mohammed Yeasin, and Noel Codella. Learning representations from eeg with deep recurrent-convolutional neural networks. *arXiv preprint arXiv:1511.06448*, 2015.

Stefano Bastianini, Chiara Berteotti, Alessandro Gabrielli, Flavia Del Vecchio, Roberto Amici, Chloe Alexandre, Thomas E Scammell, Mary Gazea, Mayumi Kimura, Viviana Lo Martire, et al. Scoprism: a new algorithm for automatic sleep scoring in mice. *Journal of neuroscience methods*, 235:277–284, 2014.

Christian R Baumann, Ertugrul Kilic, Brice Petit, Esther Werth, Dirk M Hermann, Mehdi Tafti, and Claudio L Bassetti. Sleep eeg changes after middle cerebral artery infarcts in mice: different effects of striatal and cortical lesions. *Sleep*, 29(10):1339–1344, 2006.

Suzanna Becker and Geoffrey E Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356): 161–163, 1992.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

JOEL H Benington and H Craig Heller. Rem-sleep timing is controlled homeostatically by accumulation of rem-sleep propensity in non-rem sleep. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 266(6):R1992–R2000, 1994.

Jorge Bernal, Nima Tajkbaksh, Francisco Javier Sánchez, Bogdan J Matuszewski, Hao Chen, Lequan Yu, Quentin Angermann, Olivier Romain, Bjørn Rustad, Ilangko Balasingham, et al. Comparative validation of polyp detection methods in video colonoscopy: results from the miccai 2015 endoscopic vision challenge. *IEEE transactions on medical imaging*, 36(6):1231–1249, 2017.

Patrick Bilic, Patrick Ferdinand Christ, Eugene Vorontsov, Grzegorz Chlebus, Hao Chen, Qi Dou, Chi-Wing Fu, Xiao Han, Pheng-Ann Heng, Jürgen Hesser, Samuel Kadoury, Tomasz K. Konopczynski, Miao Le, Chunming Li, Xiaomeng Li, Jana Lipková, John S. Lowengrub, Hans Meine, Jan Hendrik Moltz, Chris Pal, Marie Piraud, Xiaojuan Qi, Jin Qi, Markus Rempfler, Karsten Roth, Andrea Schenk, Anjany Sekuboyina, Ping Zhou, Christian Hülsemeyer, Marcel Beetz, Florian Ettlinger, Felix Grün, Georgios Kaissis, Fabian Lohöfer, Rickmer Braren, Julian Holch, Felix Hofmann, Wieland H. Sommer, Volker Heinemann, Colin Jacobs, Gabriel Efrain Humpire Mamani, Bram van Ginneken, Gabriel Chartrand, An Tang, Michal Drozdzal, Avi Ben-Cohen, Eyal Klang, Michal Marianne Amitai, Eli Konen, Hayit Greenspan, Johan Moreau, Alexandre Hostettler, Luc Soler, Refael Vivanti, Adi Szeskin, Naama Lev-Cohain, Jacob Sosna, Leo Joskowicz, and Bjoern H. Menze. The liver tumor segmentation benchmark (lits). *CoRR*, abs/1901.04056, 2019. URL http://arxiv.org/abs/1901.04056.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Alexander A Borb and Peter Achermann. Sleep homeostasis and models of sleep regulation. *Journal of biological rhythms*, 14(6):559–570, 1999.

Alexander A Borbély, Serge Daan, Anna Wirz-Justice, and Tom Deboer. The two-process model of sleep regulation: a reappraisal. *Journal of sleep research*, 25(2):131–143, 2016.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1002.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Chris Burgess and Hyunjik Kim. 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

Juan C Caicedo, Allen Goodman, Kyle W Karhohs, Beth A Cimini, Jeanelle Ackerman, Marzieh Haghighi, CherKeng Heng, Tim Becker, Minh Doan, Claire McQuin, et al. Nucleus segmentation across imaging experiments: the 2018 data science bowl. *Nature methods*, 16(12):1247–1253, 2019.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech

recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *International Conference on Machine Learning*, pages 864–872. PMLR, 2018.

Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Jan Chorowski, Ron J Weiss, Samy Bengio, and Aäron van den Oord. Unsupervised speech representation learning using wavenet autoencoders. *IEEE/ACM transactions on audio, speech, and language processing*, 27(12): 2041–2053, 2019.

Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

Thomas M Cover and Joy A Thomas. Elements of information theory (wiley series in telecommunications and signal processing), 2006.

Bin Dai, Ziyu Wang, and David Wipf. The usual suspects? reassessing blame for vae posterior collapse. In *International Conference on Machine Learning*, pages 2313–2322. PMLR, 2020.

Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. Avoiding latent variable collapse with generative skip models. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2397–2405. PMLR, 2019.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Hao Dong, Akara Supratak, Wei Pan, Chao Wu, Paul M Matthews, and Yike Guo. Mixed neural network approach for temporal sleep stage classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(2):324–333, 2017.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *arXiv preprint arXiv:1605.07157*, 2016.

Ronald A Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222(594-604):309–368, 1922.

Paul Franken, Alain Malafosse, and Mehdi Tafti. Genetic variation in eeg activity during sleep in inbred mice. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 275(4):R1127–R1137, 1998.

Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

Y Gal and Z Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, pages 1019–1027, 2016.

Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

Maryellen L. Giger. Machine learning in medical imaging. *Journal of the American College of Radiology*, 15(3, Part B):512–520, 2018. ISSN 1546-1440. doi: https://doi.org/10.1016/j.jacr.2017.12.028. Data Science: Big Data Machine Learning and Artificial Intelligence.

Jorge A Gonzalez and Christopher M Kramer. Role of imaging techniques for diagnosis, prognosis and management of heart failure patients: Cardiac magnetic resonance. *Current heart failure reports*, 12(4):276–283, 2015.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Multi-dimensional recurrent neural networks. In *International conference on artificial neural networks*, pages 549–558. Springer, 2007.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rylDfnCqF7.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick

Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *arXiv preprint arXiv:1902.00275*, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Yu-Liang Hsu, Ya-Ting Yang, Jeen-Shing Wang, and Chung-Yao Hsu. Automatic sleep stage recurrent neural classifier using energy features of eeg signals. *Neurocomputing*, 104:105–114, 2013.

Chin-Wei Huang, Laurent Dinh, and Aaron Courville. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. *arXiv preprint arXiv:2002.07101*, 2020.

David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

Fabian Isensee, Philipp Kickingereder, Wolfgang Wick, Martin Bendszus, and Klaus H Maier-Hein. No new-net. In *International MICCAI Brainlesion Workshop*, pages 234–244. Springer, 2018.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 1681–1691, 2015.

Alan Julian Izenman. Review papers: Recent developments in nonparametric density estimation. *Journal of the american statistical association*, 86 (413):205–224, 1991.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Markus Kiefer and Friedemann Pulvermüller. Conceptual representations in mind and brain: theoretical developments, current evidence and future directions. *cortex*, 48(7):805–825, 2012.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. volume 80 of *Proceedings of Machine Learning Research*, pages 2649–2658, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/kim18b.html.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv:1506.02557*, 2015.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

Sayaka Kohtoh, Yujiro Taguchi, Naomi Matsumoto, Masashi Wada, Zhi-Li Huang, and Yoshihiro Urade. Algorithm for sleep scoring in experimental animals based on fast fourier transform power spectrum analysis of the electroencephalogram. *Sleep and Biological Rhythms*, 6(3):163–171, 2008.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012a.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012b.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

Martin Längkvist, Lars Karlsson, and Amy Loutfi. Sleep stage classification using unsupervised feature learning. *Advances in Artificial Neural Systems*, 2012, 2012.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4): 541–551, 1989.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop bt-neural networks: Tricks of the trade. *Neural Networks: Tricks of the Trade*, 2012.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124, 2019.

James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. In *Advances in Neural Information Processing Systems*, pages 9408–9418, 2019.

Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. Biva: A very deep hierarchy of latent variables for generative modeling. In *Advances in neural information processing systems*, pages 6551–6562, 2019.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. *arXiv preprint arXiv:1812.01608*, 2018.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *International Conference on Learning Representations*, 2018.

Emmanuel Mignot. Why we sleep: the temporal organization of recovery. *PLoS Biol*, 6(4):e106, 2008.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5528–5531. IEEE, 2011.

Đorđe Miladinović, Muhammad Waleed Gondal, Bernhard Schölkopf, Joachim M Buhmann, and Stefan Bauer. Disentangled state space representations. *arXiv preprint arXiv:1906.03255*, 2019.

Đorđe Miladinović, Aleksandar Stanić, Stefan Bauer, Jürgen Schmidhuber, and Joachim M. Buhmann. Spatial dependency networks: Neural layers for improved generative image modeling. In *9th International Conference on Learning Representations (ICLR 2021)*, May 2021.

Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. Survae flows: Surjections to bridge the gap between vaes and flows. *arXiv preprint arXiv:2007.02731*, 2020.

Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pages 3918–3926. PMLR, 2018.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Matt Post. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

Ali Razavi, Aaron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. In *International Conference on Learning Representations*, 2018.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14866–14876, 2019.

Michael J Rempe, William C Clegern, and Jonathan P Wisor. An automated sleep-state classification algorithm for quantifying sleep timing and sleep-dependent dynamics of electroencephalographic and cerebral metabolic parameters. *Nature and science of sleep*, 7:85, 2015.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837, 1956. doi: 10.1214/aoms/1177728190. URL https://doi.org/10.1214/aoms/1177728190.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

Kirsi-Marja Rytkönen, Jukka Zitting, and Tarja Porkka-Heiskanen. Automated sleep scoring in rats and mice using the naive bayes classifier. *Journal of neuroscience methods*, 202(1):60–64, 2011.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in neural information processing systems*, pages 901–909, 2016.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Jürgen Schmidhuber. Learning factorial codes by predictability minimization. *Neural computation*, 4(6):863–879, 1992.

Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36. IEEE, 2004.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 627–637, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

Mai Hong Son, Mai Hong Bang, Sungwoo Bae, Dinh Truong Giang, Nguyen Tien Thinh, Jin Chul Paeng, et al. Diagnostic and prognostic value of ^{99m}tc-maa spect/ct for treatment planning of ^{90}y-resin microsphere radioembolization for hepatocellular carcinoma: comparison with planar image. *Scientific reports*, 11(1):1–9, 2021.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746, 2016.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Arnaud Sors, Stéphane Bonnet, Sébastien Mirek, Laurent Vercueil, and Jean-François Payen. A convolutional neural network for sleep stage

scoring from raw single-channel eeg. *Biomedical Signal Processing and Control*, 42:107–114, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Marijn F Stollenga, Wonmin Byeon, Marcus Liwicki, and Juergen Schmidhuber. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. In *Advances in neural information processing systems*, pages 2998–3006, 2015.

Genshiro A Sunagawa, Hiroyoshi Séi, Shigeki Shimba, Yoshihiro Urade, and Hiroki R Ueda. Faster: an unsupervised fully automated sleep staging method for mice. *Genes to Cells*, 18(6):502–518, 2013.

Akara Supratak, Hao Dong, Chao Wu, and Yike Guo. Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25 (11):1998–2008, 2017.

Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

Esteban G Tabak, Eric Vanden-Eijnden, et al. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8 (1):217–233, 2010.

Mehdi Tafti, Brice Petit, Didier Chollet, Elisabeth Neidhart, Fabienne De Bilbao, Jozsef Z Kiss, Philip A Wood, and Paul Franken. Deficiency in short-chain fatty acid β-oxidation affects theta oscillations during sleep. *Nature genetics*, 34(3):320–325, 2003.

Lucas Theis and Matthias Bethge. Generative image modeling using spatial lstms. In *Advances in Neural Information Processing Systems*, pages 1927–1935, 2015.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

Harri Valpola. From neural pca to deep unsupervised learning. In *Advances in independent component analysis and learning machines*, pages 143–171. Elsevier, 2015.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798, 2016.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Francesco Visin, Kyle Kastner, Kyunghyun Cho, Matteo Matteucci, Aaron Courville, and Yoshua Bengio. Renet: A recurrent neural network based alternative to convolutional networks. *arXiv preprint arXiv:1505.00393*, 2015.

Albrecht PA Vorster, Harini C Krishnan, Chiara Cirelli, and Lisa C Lyons. Characterization of sleep in aplysia californica. *Sleep*, 37(9):1453–1463, 2014.

Alex Waibel. Phoneme recognition using time-delay neural networks. *Meeting of IEICE, Tokyo, Japan*, 1987.

Zhiye Wang, Meizhu Zheng, and Shuang Xia. The contribution of ct and mri in staging, treatment planning and prognosis prediction of malignant tumors of external auditory canal. *Clinical Imaging*, 40(6):1262–1268,

2016. ISSN 0899-7071. doi: https://doi.org/10.1016/j.clinimag.2016.08.020.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Yuan Xue, Tao Xu, Han Zhang, L Rodney Long, and Xiaolei Huang. Segan: Adversarial network with multi-scale l_1 loss for medical image segmentation. *Neuroinformatics*, 16(3):383–392, 2018.

Farid Yaghouby, Bruce F O'Hara, and Sridhar Sunderam. Unsupervised estimation of mouse sleep scores and dynamics using a graphical model of electrophysiological measurements. *International journal of neural systems*, 26(04):1650017, 2016.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *International Conference on Machine Learning*, pages 3881–3890, 2017.

Dong Yu and Li Deng. *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019a.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343, Florence, Italy, July 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1426. URL https://www.aclweb.org/anthology/P19-1426.

Mingmin Zhao, Shichao Yue, Dina Katabi, Tommi S Jaakkola, and Matt T Bianchi. Learning sleep stages from radio signals: A conditional adversarial architecture. In *International Conference on Machine Learning*, pages 4100–4109. PMLR, 2017.

Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: Redesigning skip connections to exploit multi-scale features in image segmentation. *IEEE transactions on medical imaging*, 39(6):1856–1867, 2019.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.