A coding perspective on deep latent variable models

Ullrich, K.

Publication date
2020
Document Version
Final published version
License
Other

A CODING PERSPECTIVE ON DEEP LATENT VARIABLE MODELS BY KAREN ULLRICH

KAREN ULLRICH

A CODING PERSPECTIVE

-ON DEEP LATENT VARIABLE MODELS

KAREN ULLRICH

A Coding Perspective on Deep Latent Variable Models

A Coding Perspective on Deep Latent Variable Models

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor

aan de Universiteit van Amsterdam

op gezag van de Rector Magnicus

prof. dr. ir. K.I.J. Maex

ten overstaan van een door het College voor Promoties ingestelde commissie,

in het openbaar te verdedigen in de Aula der Universiteit op

vrijdag 11 september 2020, te 11 uur

door Karen Ullrich

geboren te Leisnig (Duitsland).

Promotiecommissie:

Promotor:	Prof. dr. M. Welling,	Universiteit van Amsterdam
Coromotor:	Dr. P. Forré,	Universiteit van Amsterdam
Overige leden:	Prof. dr. J.M. Mooij,	Universiteit van Amsterdam
	Prof.dr. P.D. Grünwald,	Universiteit Leiden
	Prof. dr. D. Vetrov,	National Research University Higher School of Economics
	Dr. R.E. Turner,	University of Cambridge
	Dr. M. Cheng,	Universiteit van Amsterdam
Faculteit:	Faculteit der Natuurwetenschappen, Wiskunde en Informatica	

Preface

In my thesis "A Coding Perspective on Deep Latent Variable Models", we discuss how statistical inference in Deep Latent Variable Models (DLVMs) relates to coding.

In particular, we examine the minimum deception length (MDL) principle as a guide for statistical inference. In this context, we explore its relation to Bayesian inference. We shall see that despite both leading to similar algorithms, the MDL principle allows us to make no assumption about the data generating process. We merely restrict ourselves to finding regularity in the observed data, where regularity is connected to the ability to compress. We thus find that learning DLVMs is equivalent to minimizing the cost for communicating (compressing) a set of observations. One common approach to communication is to send a hypothesis (or model), and subsequently the data misfit under the aforementioned model. This is known as the two-part code. In this thesis, we will mainly focus on the so-called Bayesian code – a theoretically more effective code than the two-part code.

Somewhat counter-intuitively, the Bayesian inference method will allow us to compute the code length without knowing the code nor the coding scheme that achieved this code length. The purpose of this thesis is to close this gap by developing respective coding schemes. We will, inspired and guided by the MDL principle, look for the codes that achieve the code length predicted by MDL. A special focus lies on differentiable functions, and more precisely, deep neural networks, learned by way of large quantities of high dimensional data. We will investigate model compression as well as source compression through the lens of the MDL principle.

Samenvatting

In mijn dissertatie "A Coding Perspective on Deep Latent Variable Models" bespreken we hoe statistische gevolgtrekking in Deep Latent Variable Models (DLVMs) zich verhoudt tot coderen.

We onderzoeken specifiek het principe van minimum deception length (MDL) als een gids voor statistische gevolgtrekking. In deze context verkennen we de relatie met Bayesiaanse gevolgtrekking. We zullen zien dat ondanks dat beide tot vergelijkbare algoritmes leiden, het MDL principe ons toestaat om geen aannames te maken over het proces dat de data genereert. We beperken onszelf slechts tot het vinden van regelmatigheden in de geobserveerde data, waar regelmatigheid verbonden is aan het vermogen tot comprimeren. We ontdekken dus dat het leren van DLVMs equivalent is aan het minimaliseren van de kosten voor het communiceren (comprimeren) van een set van observaties. Eén veelvoorkomende aanpak voor communicatie is het versturen van een hypothese (of model) en vervolgens de data "misfit" onder eerdergenoemd model. Dit staat bekend als de tweedelige code. In deze dissertatie zullen we voornamelijk focussen op de zogenoemde Bayesiaanse code – theoretisch een effectievere code dan de tweedelige code.

Enigszins tegenintuïtief laat de Bayesiaanse methode voor gevolgtrekking ons de lengte van de code uitrekenen zonder de code of de codering te kennen die deze lengte behaalde. Het doel van deze dissertatie is het sluiten van deze kloof door het ontwikkelen van respectievelijke coderingen. We zullen, geïnspireerd en geleid door het MDL principe, codes zoeken die de lengte behalen die door MDL voorspelt. Een speciale focus ligt op differentieerbare functies, en specifieker, diepe neurale netwerken, geleerd via grote hoeveelheden hoog dimensionale data. We zullen zowel model compressie als bron compressie onderzoeken door de lens van het MDL principe.

Contents

1

Introduction and Background

In this chapter, we shall introduce essentials to frame the remainder of this thesis and develop our research questions.

We shall start by recapping models for communication, especially those that have been picked up and interpreted by the machine learning community (Section 1.1). We shall continue by introducing the foundations of inference and learning in the communication models we consider in this thesis. Specifically, we will focus our view on the minimum description length (MDL) principle as a credo for sending information (Section 1.2). The importance of the MDL view for learning heavily parametrized communication systems will be exemplified on two areas of interest: function compression (Section 1.3) and deep latent variable models for communication (Section 1.4).

1.1 Information Theory

At the core of information theory, we study systems that convey information from one point to another point. Alternatively, we shall describe the systems as transmitting *messages* from a *sender* to a *receiver*. The fundamental *problem of communication* to be solved is to forward messages such that the received reconstructed message is either an approximate or exact copy of the original message. We shall

refer to these two goals as *lossy or lossless communication*, respectively. The communication problem is caused by the medium through which information is passed. The effect of an inherently noisy environment on the messages is described by the *channel*.

Systems of communication can be found anywhere in technology, society and nature. Some prominent example include;

(1) Most will promptly think of digital and analog communication in technology: two modems that communicate via a telephone line, two radios that communicate through the air, or two computers that are connected through the internet.

(2) Communication can be more than spacial, however. When writing information from the computer RAM to a hard disk drive, we can think of the process of storing the information as communication that unfolds in the time dimension.

(3) Further, by no means is the concept of communication restricted to human made systems. Conveying messages from one living being to another is a core problem in nature approached by different means. As seen in many mammalian species, one may send messages by the medium of urine to be received by the other party's sense of smell. Similar examples include visual markings found in insects, bird vocal communication, and of course human language.

(4) Communication is also present in less conscious processes. Evolution is a communication problem, transporting information from one generation to another. Reproduction and repair of any cell is a communication problem, where science has identified the message to be conveyed as DNA and RNA. Mistakes in communication of this information can lead to cancer. It is therefore fascinating that even this unconscious communication process appears to have found mechanisms to prevent errors in transmitting messages, thereby making long lived lives and complex individuals with billion of cells possible – each cell representing a successful communication.

Imagine we were to naively send a message through a channel as exemplified in Figure 1.1. We would find two kinds of errors in the reconstructed message: (i) the message may be cut short because it is too large to be carried through the channel; and (ii) the message would suffer from corruption due to the nature of the transfer medium. We find examples of errors of the second kind in examples (1) and (2).

Telephone lines may suffer from cross-talk, radio-waves interfere with each other and are corrupted by other signals traveling through air, hard-drives suffer from corruption when magnetization changes spontaneously. A common error of the first kind is often present in example (3). Smell, visual markings, and language can all suffer from capacity constrains. This can lead to ambiguous message reconstructions and thus misunderstandings.

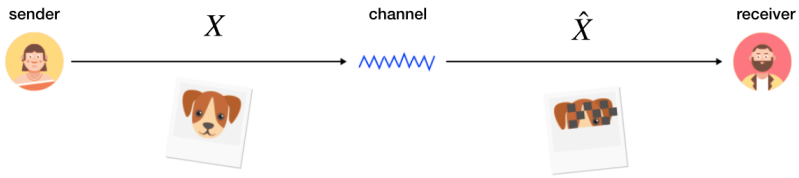

Figure 1.1: The sender is trying to transmit a message X to the receiver. However, due to the channel the received massage $\hat{X}$ is incomplete and corrupted.

In the next two sections, we will unfold principled approaches to tackling both errors and identify the limits of successful communication. In our discussion, we assume that we have no access to the channel itself. Hence, we ignore the option to physically improve the channel to perfection. It may be argued that this is obvious, because all systems undergo entropic decay eventually. However, often the decay rate is small enough for this to be ignored as an argument. For example, instead of using ferromagnets to store information digitally, moving to a storage medium such as glass with fewer spontaneous corruption events may yield great benefits [6]. Arguably the main reason as to why an algorithmic solution to the communication problem should receive strong consideration is coming from a cost point of view. Improving a channel physically will increase the communication cost for each message sent through the channel. Furthermore, algorithmic solutions may be transferred more easily to other communication domains, further reducing the cost per sent message. In short, we assume algorithmic solutions to be more scalable due to the negligible cost of replicating algorithmic solutions.

Source compression: removing redundancy

A channel may restrict the amount of information that can pass through per time unit and additionally corrupt the information that does pass. In this section, we address the former problem. For this, we shall compress the messages to be sent as much as possible.

Let us formalize our ideas by thinking of the sender as emitting an unlimited stream of messages $\mathcal{D} = \{x_i\}_{i=1}^{\infty}$. We may think of a message as random variable X. Messages are sampled from the source $X \sim P_S(X)$, where we approximated the source as a distribution. We shall map each message to a finite length string of symbols from a finite length dictionary. For example, in Figure 1.2, we map the dog image to the binary string 110010. We call this map a source code $SC(\cdot)$, i.e., $SC(\text{'dog'}) = 110010$. We can now estimate how long we expect a source code to be on average

$$L(C) = \sum_{x_i \in \mathcal{D}} P(X = x_i)\text{len}(SC(x_i)), \tag{1.1}$$

where $\text{len}(\cdot)$ denotes the length of the source code. In the interest of sending short messages, we want to optimize the average length of a source code to be as short as possible. Intuitively, it should be clear that there is a lower bound to the achievable code length. This bound is determined by how much information the source emits. We can quantify the amount of information emitted by the source by its entropy $H(X)$ [1]. Shannon's source coding theorem states more concretely, that the source entropy bounds the expected code length $H(X) \leq L(C)$.

[1] $H(X) := \sum_x P(X = x) \log P(X = x)$

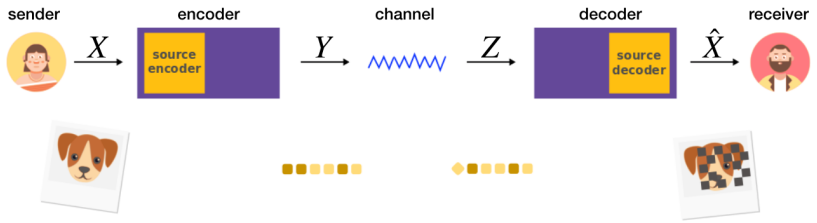

Figure 1.2: In contrast to Figure 1.1, we now compress the message before sending it through the channel. We use a source code to compress the message X to the encoding Y, $SC : X \to Y$. In our example, the message is a dog picture, and the respective code is '110010' (visualized by dark and bright pixels). The encoding Y gets corrupted by the channel such that some bits change their orientation, e.g. '110010' $\to$ '010010'. As a result of source coding the entire message passes the channel, but still suffers from corruption.

This bound applies if we seek to compress without loss of information. We can create even shorter codes if we allow for a certain level of message distortion after reconstruction. The following formulation will also allow for random variables with infinite source alphabet. More precisely, the distortion measure $d(x_i, x_j)$ between two messages x_i and x_j may be any deterministic, non-negative function. An encoding function maps messages to encodings $q : X \to Y$. A decoding or reconstruction function inverts the encoder approximately $g : Y \to X$. We can compute the expected distortion, given an encoder and a decoder, to be

$$D = \sum_{x_i} P(X = x_i)d(x_i, g\left(q\left(x_i\right)\right)). \tag{1.2}$$

A lower bound on the expected code length L given a distortion mea-

sure and distortion is given by the information rate distortion function

$$R(D) = \min_{p(\hat{x}|x):\ \sum_{(x,\hat{x})} p(x)p(\hat{x}|x)d(x,\hat{x}) \leq D} I(X;\hat{X}) \tag{1.3}$$

where the minimization is over all conditional distributions $p(\hat{x}|x)$ for which the joint distribution $p(\hat{x}, x)$ is limited by the distortion D. We can relate the notions of rate and distortion to the source entropy with the Rate-Distortion theorem. It states that the rate-distortion function lower bounds achievable codes,

$$H(X) - D \leq R(D) \leq L(q, g, D). \tag{1.4}$$

Channel coding: error-correcting codes

In the previous section, we have discussed how to reduce a message to its essential information units. When sending the compressed message through the channel without further alterations, we would be left with random corruptions as exemplified in Figure 1.2. In this section, we will discuss how we can send messages without error despite noisy channels.

Possible errors cannot be detected by observing the received source sequence itself, since when we compress our data optimally, all source bits are independent. We thus need to add redundancy to our message. As a consequence the sent messages will become longer. We shall discuss by how much longer they become at the end of this section. A natural solution to the corruption problem is to send repetition codes such as illustrated in Figure 1.3. For this, we simply repeat the source signal N times. We decode the received code by averaging. For the binary channel, the probability of error p_e is related to the code length by $p_e \sim (p_b(1 - p_b))^N$.

However, we can do better than repetition. The central idea behind other codes is to correlate redundant bits with more than one variable of the source code. In Hamming codes this is known as parity checks.

This leads us to a central question: How good can such a code possibly be. To find out, we first need to establish a property that evaluates the channel – the channel capacity. We define the channel capacity to be

$$C(P(Z|Y)) = \max_{P(Y)} I(Y;Z). \tag{1.5}$$

The distribution $P(Y)$ that achieves the maximum is the optimal input distribution. Note that, the mutual information between channel in-

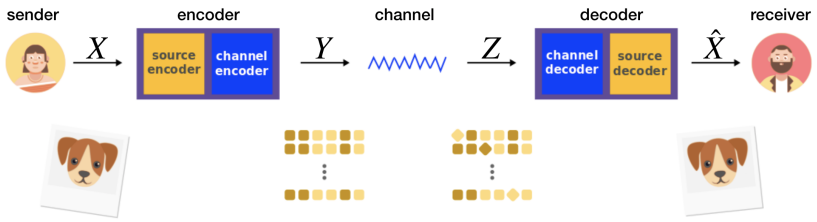

Figure 1.3: By source compression and error correction we can send the message without corruption as long as the source entropy $H(X)$ is smaller than the channel capacity $SC(P(Z|Y))$.

and output is crucial for reconstructing the output. Given that we can change the input distribution to the channel, we may look for the input distribution on source codes that makes best sense for the channel. The noisy channel coding theorem shows that the capacity does indeed measure the maximum amount of error free communication that can happen over the channel per time unit. Specifically, for every discrete memory-less channel, when we seek to send source messages with a rate smaller than the capacity $R < C$, an arbitrarily small error rate can be achieved.

Joint source channel coding theorem

Next, we will demonstrate how we can combine both of the results discussed in the previous two sections. That is, in the error free case of transmitting finite alphabet random variables: (1) we can compress messages X such that the average code length R is at least the source entropy $R < H(X)$, and (2) we can transmit our code Y only through the channel with arbitrarily small error if the average code length is smaller than the channel capacity $R < C$. We suspect the source entropy smaller than the channel capacity $H < C$ to be a necessary condition for successful transmission. This is not immediately clear, because we have applied a two stage (data compression and transmission) communication process. However, our intuition will prove to be correct. The joint source channel coding theorem (SCCT) yields the following results: (1) There exist encoders and decoders that allow for error free message transmission, but only if the source entropy is smaller than the channel capacity $H < C$. Conversely, if the channel capacity is smaller than the source entropy $C < H$, error free transmission is not possible. (2) There exists an optimal solution, that uses a two stage method.

The result can be extended for the case that we allow for some distortion. It can be shown that for i.i.d. random variables and memoryless

channels, there exists a solution to the communication problem if and only if the rate given a distortion is smaller than the channel capacity, $R(D) < C$.

Let us think about the implications of the SCCT. We can think of a communication problem to be defined by its source and its channel. For a separate solution, we aim to find a good source coding scheme, depending only on the source, and a good channel coding scheme, depending only on the channel. Any good solution for a given source, may be reused for any other channel and vise versa. That means the total number of communication problems in the world can be reduced from $M \cdot N$ joined problems to $M + N$ separate problems, where M is the total number of sources in the world and N the total number of channels.

The SCCT can be misleading or lead to wrong conclusions. First, it might be hard to identify the source entropy or channel capacity especially for empirical sources and channels. In practical situations, we may often only have access to an imperfect solution to either source or channel coding. It is hard to predict how much an imperfect solution to one problem affects the other solution. Furthermore for the theorem to hold, we must ignore common properties of channels such as memory and feedback. Moreover, we think of the entire process as an infinite stream of information. However, in reality we are often interested in sending a package of information. For finite length messages, we need to trade compression and transmission bits against each other. Finally, knowing that there exists a solution does not necessarily constitute one finding it.

1.2 Minimum description length principle

In this section we address two limitations of Shannon's studies that stem from his description of the message source. As discussed in the previous section, in reality we seek to send finite messages instead of infinite ones. Thus it seems to be important to account for shared knowledge between sender and receiver, such as the message decoder. If one were not to do so, the messages to be sent can be stored within the decoder entirely, reducing the cost of sending to zero. To be fair, we have to account for the information that is stored in the decoder too. Further we shall not make any assumption about the true data

generating process because any model is a simplified abstraction of a real source and thus no model can contain the true source.

Naturally, the idea arises to compress the message as concisely as possible including the information that is placed in the decoder. How would one formalize this concept practically? Solomonoff [195] suggests a criterion based on universal computer languages[2].Every data sequence S may be encoded by a computer program P that prints S and than halts. The description length of the data sequence is thus defined by the length of the program (in bits) that prints S and than halts. We can generalize the idea of a programming language with the notion of a description method. A description method is a one-to-many relation from the sample space to the set of binary strings of arbitrary length.

[2] Universal languages are those that can implement universal Turing machines. Commonly used computer languages are usually universal.

The Kolmogorov complexity measures the length of the shortest program to describe the data.[3] The Kolmogorov complexity measures the ideal message length after compression accounting for the coding hypothesis. In general, the Kolmogorov complexity cannot be computed because a description method as discussed here cannot exist. Assuming that for every possible input such a method exists leads to contradiction [121]. Further, for short sequences S the invariance theorem is not relevant and the Kolmogorov complexity is dominated by a large, description method dependent, constant.

[3] Solomonoff [195], Kolmogorov [110] and Chaitin [27] show that asymptotically, the complexity is independent of the programming language used, known as the invariance theorem. In fact, the Kolmogorov complexity is language independent up to a (possibly large) constant.

There are two schools of thought for dealing with these problems: The algorithmic or idealized minimum description length (MDL) school that assumes the two problems can be discounted [4]; and the practical MDL school that tackles uncomputability and large constants (due to universal programming languages) explicitly by using a less than universal description method. Any such description method must be a compromise in that it should be general enough to describe many similar sequences but also restrictive enough such that we can compute the Kolmogorov complexity for any input S. For many description methods, we can reduce the description length constant considerably and express their Kolmogorov complexity explicitly. However, we must acknowledge that the choice of method is guided by some prior knowledge about S. Thus there will be sequences for which the description method fails to produce a description.

[4] This school can be subdivided again into authors that focus more on Solomonoff [195] original work and those that focus on Kolmogorov [110] studies.

As an example, consider sending data $\mathcal{D}$ by means of a class of parametric models $\mathcal{H}$. We describe each model $H \in \mathcal{H}$ by describing its parameters. The number of parameters we need and their respective precision will influence the description length of the hypothesis $L(H)$.

We send S in a two-part code: we first send the model H and then the data misfit[5]. Identifying the best hypothesis H becomes a matter of balancing the complexity of the hypothesis $L(H)$ and the error made by H in describing the data $L(\mathcal{D}|H)$,

$$L(\mathcal{D}) := \min_H L(H) + L(\mathcal{D}|H). \tag{1.6}$$

[5] For any probabilistic observation model $P(\mathcal{D}|H)$, the Shannon-Fano code relates the observation model to the description length $L(\mathcal{D}|H) = -\log P(\mathcal{D}|H)$.

This is the most common interpretation of MDL and, as such, has been introduced by Rissanen [173, 175]. Note that in contrast to idealized MDL, the winning hypothesis may strongly depend on its description method. To tackle the arbitrary length issue, it has been advocated to refine the MDL principle [175, 15] such that one should chose the best hypotheses under worst-case assumptions for the code (minimax code). However, this is not a large concern for this work. In the context of this work, the description length of H is implicitly given by the task at hand. It can be the number of flops to make a prediction, or the number of bits to store H. We acknowledge that our approach may fail the test of time, for a given hypothesis, when implementation details or hardware of an underlying system change over time. A reasonable solution is provided by repeating experiments presented here.

The simple two-part method described in this section can be related to maximum a posteriori (MAP) inference. Specifically in the two-part code we first encode a hypothesis H or its representative parameters θ using a code L_{MAP}. The hypothesis specifies a prior over the parameters as well. The two-part code length is thus given by

$$L_{\text{MAP}}(\mathcal{D}) = \min_{H \in \mathcal{H}} -\log P(\mathcal{D}|H) - \log P(H) \tag{1.7}$$

$$= \min_{\theta \in \Theta} -\log P(\mathcal{D}|\theta) - \log P(\theta), \tag{1.8}$$

where $-\log P(\theta)$ depends on the specific code L_0 that is used.

The MDL principle may be extended to include probabilistic hypothesis as well. To achieve this we optimize $L(\mathcal{D}|\mathcal{H})$ such that, if there exists a hypothesis H in the model class $\mathcal{H}$ with small description length $L(\mathcal{D}|H)$, $L(\mathcal{D}|\mathcal{H})$ will be small too. As a specific example we shall introduce the Baysian code,

$$L_{\text{Bayes}}(\mathcal{D}) = -\log \sum_{H \in \mathcal{H}} P(\mathcal{D}|H)P(H) \tag{1.9}$$

$$= -\log \sum_{\theta \in \Theta} P(\mathcal{D}|\theta)P(\theta). \tag{1.10}$$

We can show that from an MDL perspective Bayesian inference is to be preferred over the two-part (MAP) inference method. The Bayesian

code is a lower bound to the two-part code,

$$L_{\text{Bayes}}(\mathcal{D}) = -\log \sum_{\theta \in \Theta} P(\mathcal{D}|\theta)P(\theta) \tag{1.11}$$

$$\leq \min_{\theta \in \Theta} -\log P(\mathcal{D}|\theta) - \log P(\theta) = L_{\text{MAP}}(\mathcal{D}). \tag{1.12}$$

Generally, the Bayesian code will provide shorter code-length than the two-part code. From this analysis it is not clear however, which code can achieve this code-length. In the next section, we shall discuss which coding scheme can achieve the this code-length approximately.

To conclude, in this section, we have introduced the MDL principle as a basis for learning. We have seen that we can formulate MAP and full Bayesian inference as instances of the principle; and that, of the two, Bayesian inference is to be preferred. However, note that the MDL principle motivates other forms of inference that have no Bayesian counter part such as the Shtarkov normalized maximum likelihood code or the prequential plug-in code that we will not discuss in this thesis (see Grünwald [63], Chapter 11).

Exemplified by the Bayesian code, we have observed another obscurity in coding with the MDL principle: To construct the code length of the Bayesian code, we did not require to construct an actual code. This is an interesting property when using the MDL for complexity analysis of functions, but of little value for us that we seek the codes to achieve the MDL. We will tackle this problem in the next sections.

Approximate Bayesian inference and MDL principle

In the previous section, we have seen that the MDL point of view provides a motivation for Bayesian inference. Thus, both frameworks lead to the same learning methodology. We may thus regard them as equivalent, despite their variance in motivation[6]. In this section, we strive to develop Bayesian codes. Derived from the previous discussion, we expect them to outperform two-part codes. Our previous analysis only showed the advantage in expected code length, however, it did not provide the code that can achieve such code length. We now turn to exploring possible coding schemes. We will see that we need to approximate Bayesian codes due to infeasible integration. Different approximation techniques will yield different codes. In the cases that we deem relevant, it turns out that variational inference provides a practical framework for both feasible approximate Bayesian inference and a code.

[6] Bayesian statistics are usually motivated through Cox's axioms. Cox's axioms state that a subject shall act rational with respect to the information it is given [36]. A subjects beliefs are represented by probabilities. Observations in the world are weighted by a given utility function to provide for optimal decision making based on previous observations.

To see the need for approximation, lets formalize our ideas more precisely. Consider observations $\mathcal{D} = \{x_i(t)|i,t\}$, that shall be fit to a parametric model governed by $\boldsymbol{\theta} = \{\theta_j|j\}$. The two-part code in this setting would correspond to a maximum a posteriori (MAP) approach to finding the single set of variables that maximize $p(\boldsymbol{\theta}|\mathcal{D})$. Yet, we already know that if we can include more point hypotheses, we should be able to get a shorter code. For this we need for the parameter values to be distributed according to the posterior probability distribution of the parameters given the data, and all the assumptions related to using the given model $p(\boldsymbol{\theta}|\mathcal{D},\mathcal{H})$. We can evaluate the posterior by means of the Bayes rule

$$p(\boldsymbol{\theta}|\mathcal{D},\mathcal{H}) = p(\mathcal{D}|\boldsymbol{\theta},\mathcal{H})\frac{p(\boldsymbol{\theta}|\mathcal{H})}{p(\mathcal{D}|\mathcal{H})}. \tag{1.13}$$

The evaluation process includes the marginalization over the parameter space. Therefore, even if we were to have a clear way of realizing the Bayesian code, in many realistic scenarios, it is difficult to even evaluate the full posterior due to the burden of marginalization. We thus rely on approximation techniques including variational approximations and stochastic approximations provided by different Markov chain Monte Carlo (MCMC) methods. In the following, we will demonstrate two coding schemes approximating the Bayesian code.

Recall that according to Shannon's coding theorem the code length $L(x)$ of a continuous or discrete variable is bounded by its entropy. The optimal code encodes each variable with $-\log_2 p(x)$ bits or $-\log p(x)$ nats for binary or continuous variables, respectively. For discrete latent variables, there are optimal coding schemes (prefix codes) [35]. The models that we are interested in, in the scope of this thesis, however, are parameterized primarily by continuous variables. Because we cannot send continuous variables with infinite precision, we require some form of vector quantization (VQ) for the sending scheme. [7]

[7] VQ is a classical technique from signal processing that allows the modeling of probability density functions by the distribution of prototype vectors. Examples include k-means quantization, product quantization, residual quantization, optimal fixed point and hashing quantization.

1.3 Coding schemes based on approximate Bayesian inference

In the following, we will motivate two coding schemes that achieve the Bayesian code length approximately. Frequently, the situation is such that we aim to send continuous symbols from a similarly continuous distribution. In this scenario, sending even one symbol at arbi-

trary precision would lead to an infinitely long message (see Equation (1.16), for $\epsilon \to 0$). We thus instead only restrict ourselves to coding up

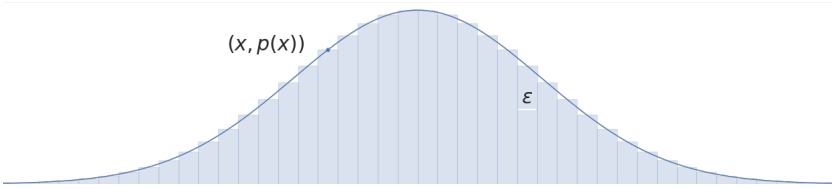

Figure 1.4: The figure shows the probability mass associated with a quantized probability distribution. The quantization width ϵ determines the how precisely the approximation matches the continuous distribution. The log-probability mass of the continuous distribution is approximated by $\log p(x) + \log \epsilon$. As is intuitive, the narrower we choose ϵ, the more information we need to transmit.

to a tolerance of error t. As illustrated in Figure 1.4, the quantization width ϵ determines that error. The error tolerance and the quantization width are linked as follows,

$$t = \max_{x \in supp(p(x))} |p(x) - p(x \pm \tfrac{1}{2}\epsilon)|, \tag{1.14}$$

where $supp(p(x))$ is the support of $p(x)$. The probability mass of the continuous distribution is approximated by $p(x)\epsilon$. Thus, we expect an optimal code of length $-\log(p(x)\epsilon)$ nats per message. Equivalent to the two-part code, we may postulate the following coding scheme for the Bayesian code. First, we send the parameterized model $(\mathcal{H}, \boldsymbol{\theta})$, which models the data, through the channel. Assuming the model parameters are continuous, we assign the quatization width to be $\epsilon_\theta^{|\theta|}$, and we similarly assign the quatization width of our continuous observation model to be $\epsilon_x^{|\mathcal{D}|}$. Subsequently, the model-data misfit is communicated, resulting in

$$\begin{aligned} L(\mathcal{D}) &= L(\boldsymbol{\theta}) + L(\mathcal{D}|\boldsymbol{\theta}) && (1.15)\\ &= -\log\left(p(\boldsymbol{\theta}|\mathcal{H})\epsilon_\theta^{|\theta|}\right) - \log\left(p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{H})\epsilon_x^{|\mathcal{D}|}\right) && (1.16)\end{aligned}$$

where $L(\boldsymbol{\theta}) = -\log\left(p(\boldsymbol{\theta}|\mathcal{H})\epsilon_\theta^{|\boldsymbol{\theta}|}\right)$ is the description length of the model parameters using a given prior distribution and precision ϵ_θ. The description length to convey the model-data misfit up to precision ϵ_x is given by $L(\mathcal{D}|\boldsymbol{\theta}) = -\log\left(p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{H})\epsilon_x^{[\mathcal{D}]}\right)$, where $|\mathcal{D}|$ and $|\boldsymbol{\theta}|$ denote the number of elements in the respective sets. Ignoring the error and parameter precisions for a moment, we note that optimizing this objective is equivalent to identifying the MAP estimate. Therefore, to achieve a short description length, we can decrease the parameter precision by increasing ϵ_θ. Our hope is that despite giving up information and effectively collapsing many hypothesis into equivalence classes, the model-data misfit is maintained. We may speculate that this approach works by exploiting regularity in the model class.

The strategy outlined in the previous paragraph, however, leads to some challenges. When performing an exact Bayesian evaluation, each

parameter setting contributes zero probability mass to the continuous density. When we quantize, many parameter settings are collapsed, thus collapse the parameter probability density functions (PDFs). Thus, some areas of the original PDFs will carry much density while others may carry none. The more we quantize, the more we average over the original PDF. This may result in poor model performance. Even worse, one can imagine that, on top of quantization, we use a statistical approximation such as MC sampling. Encountering a high density model may be so unlikely that we never sample one. This leads to poor approximations of the averaged regions. Besides the effect of quantization on the model itself, in the current formulation (Equation (1.16)), there is no feedback from the change we invoke to the model, to how this translates to the model-data misfit.

As a solution to both aforementioned problems, we make use of importance sampling. For this, let us introduce a quantized approximation to the parameter distribution

$$q(\boldsymbol{\theta}|\mathcal{H}, \epsilon_\theta^{|\theta|}) \approx \int\limits_{\left[\theta-\frac{1}{2}\epsilon_\theta^{|\theta|}, \theta+\frac{1}{2}\epsilon_\theta^{|\theta|}\right]} p(\hat{\theta}|\mathcal{H}) d\hat{\theta}. \tag{1.17}$$

We now rewrite Equation (1.16) to

$$\begin{aligned} L(\mathcal{D}) &= -\log\left(\int p(\boldsymbol{\theta}|\mathcal{H}) p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{H}) \epsilon_x^{|\mathcal{D}|} d\boldsymbol{\theta}\right) && (1.18)\\ &= -\log\left(\mathbb{E}_{q(\boldsymbol{\theta}|\mathcal{H}, \epsilon_\theta^{|\theta|})}\left[\frac{p(\boldsymbol{\theta}|\mathcal{H})}{q(\boldsymbol{\theta}|\mathcal{H}, \epsilon_\theta^{|\theta|})} p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{H}) \epsilon_x^{|\mathcal{D}|}\right]\right) && (1.19)\\ &\geq -\mathbb{E}_{q(\boldsymbol{\theta}|\mathcal{H}, \epsilon_\theta^{|\theta|})}\left[\log\left(\frac{p(\boldsymbol{\theta}|\mathcal{H})}{q(\boldsymbol{\theta}|\mathcal{H}, \epsilon_\theta^{|\theta|})} p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{H}) \epsilon_x^{|\mathcal{D}|}\right)\right], && (1.20) \end{aligned}$$

where we have made use of Jensen's inequality. To conclude, we have made several approximations to the Bayesian code. First, we quantized the model. Second, to circumvent the required model marginalization and combat problems introduced by the quantization process, we required the introduction of a parameter distribution approximation. This led to a likelihood ratio of $q(\boldsymbol{\theta}|\mathcal{H}, \epsilon_\theta^{|\theta|})$ and $p(\boldsymbol{\theta}|\mathcal{H})$. When combined with sampling, this is also known as importance sampling. The above consideration lead us to the first research question for this thesis.

Research Question 1

Research Question 1: Can we identify a coding scheme that achieves the code length in Equation (1.20) ? Can we learn how much quantization $\epsilon_\theta^{|\theta|}$ a model can sustain while maintaining its performance? Can

our approach be of practical use for compressing deep neural network models?

In answering this research question we refer the reader to Chapter 2 of this thesis.

Research Question 1 aims at performing a strategy for non-uniform precision quantization of continuous variable models. Next, we shall discuss an alternative approach that is based on an approximation to the Baysian mixture code, where we use a mixture model to turn continuous variables models into discrete ones. Specifically, we use a mixture model to describe the true parameter distribution $p(\boldsymbol{\theta}|\mathcal{H}) \approx \sum_i p(i)p(\theta|\mathcal{H}, i)$. Next, each parameter is approximated with one of the components in the mixture model $q(\boldsymbol{\theta}|\mathcal{H}) = \max_i p(\boldsymbol{\theta}|\mathcal{H}, i) = p(\boldsymbol{\theta}|\mathcal{H}, i^*)$. The corresponding objective is

$$L(\mathcal{D}) = -\log\left(\int p(\boldsymbol{\theta}|\mathcal{H})p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{H})\epsilon_x^{|\mathcal{D}|}d\boldsymbol{\theta}\right) \tag{1.21}$$

$$\geq -\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{H},i^*)}\left[\log\left(\frac{\sum_i p(i)p(\boldsymbol{\theta}|\mathcal{H}, i)}{p(\boldsymbol{\theta}|\mathcal{H}, i^*)}p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{H})\epsilon_x^{|\mathcal{D}|}\right)\right]. \tag{1.22}$$

Under the constraint that only one component claims responsibility for the approximation $q(\boldsymbol{\theta}||\mathcal{H})$,

$$\frac{p(i^*)p(\boldsymbol{\theta}|\mathcal{H}, i^*) + \sum_{i\neq i^*} p(i)p(\boldsymbol{\theta}|\mathcal{H}, i^*)}{p(\boldsymbol{\theta}|\mathcal{H}, i^*)} \approx p(i^*) + 0, \tag{1.23}$$

the cost of sending the model now reduces to $\log_2 p(i^*)$ bits per parameter. In other words, we only send the description for the mixture component that best describes a model parameter. This allows us to model with continuous distributions while sending discrete symbols. Our examinations raise the second set of research questions of this thesis.

Research Question 2

Research Question 2: Can we identify a coding scheme that achieves the code length in Equation (1.22) under the constraint from Equation (1.23)? Can we learn how much quantization a model can sustain while maintaining its performance? Can our approach be of practical use for compressing deep neural network models?

In answering this research question we refer the reader to Chapter 3 of this thesis.

Conclusion

In this section, we have introduced two coding schemes, both require us to use variational inference. We have hinted at, but not yet discussed, how to optimize such models. This will be laid out in the respective chapters.

The reader may have noticed that we seem to give up a lot of information in the model by the quantization we perform. Even though giving up information improves the coding length, this surely will effect the model performance ("No pain no gain"). However, as we will see, the overparameterized neural network models that we are dealing with are able to lose a lot of information while maintaining model performance.

The reader may also wonder why we have not discussed another coding scheme – the so called bits-back coding. In the next section, we shall discuss this alternative coding idea. Numerous authors have experimented with the idea of sending distributions rather than parameter settings [80, 84, 134, 52, 51]. Practically, one would send a code using redundant code words. The code words are selected based on auxiliary information. The receiver can identify the redundant bits in the code by rerunning the encoding procedure and thus get its bits-back [8]. One question then remains: How to use the received bits? It turns out that they can be used when the variables being sent are source encodings.

[8] The term bits-back has been coined by [80].

1.4 Latent variable models as communication models

In the previous section, we discussed model communication from an MDL point of view. In the following, we will show how we can widen our perception and transfer the insights and ideas from model to source coding.

Latent variable models for source coding

Let us consider a source that emits partially redundant messages $\mathcal{D} = \{x_i\}_{i=1}^N$. We wish to find a code $SC : X \to Y$ that maps the original

messages to a code such that the overall description length is as short as possible according to Equation (1.1). The coding scheme at use is similar to the one we used when communicating a model. We first send the encodings $\{SC(x_i)\}_{i=1}^N = \{y_i\}_{i=1}^N$, thereafter we communicate the model misfit [9]. As in the previous section, we prefer the Bayesian code over the two-part code. This leads to a code with description length,

$$L_{\text{Bayes}}(X;\mathcal{H}) = -\log \int_{\Omega} p(X|Y,\mathcal{H})p(Y,\mathcal{H})dY, \tag{1.24}$$

per message. We face precisely the same situation as before: even if we were able to connect the code length to a coding scheme, we face the problem of intractability first. We hence need to make approximations. From the approximation techniques at hand, we shall choose variational inference, because, as we will discuss in the next section, we can derive a general coding scheme based on this particular approximation.

[9] To simplify the discussion, in this section, we will disregard the effect of the channel. We motivate this decision in Section 1.1. We will also ignore the cost of communicating the model. This task is orthogonal to our discussion here.

Assume we choose a conditional approximation $q(Y|X,\mathcal{H})$ to the posterior. Equivalent to Equation (1.20), the bound to $L_{\text{Bayes}}(X;\mathcal{H})$ becomes

$$L_{\text{Bayes}}(X;\mathcal{H}) \geq -\mathbb{E}_{q(Y|X,\mathcal{H})}\left[\log\left(\frac{p(Y|\mathcal{H})}{q(Y|X,\mathcal{H})}p(X|Y,\mathcal{H})\epsilon_x^{|X|}\right)\right]. \tag{1.25}$$

Note that $q(Y|X,\mathcal{H})$ acts as the code SC. We have talked about two coding schemes involving variational inference that request a particular form of the posterior approximation to the parameters. In the next section, we see that when compressing many messages we can efficiently code any distribution $q(Y|X,\mathcal{H})$ independent of the source model. Knowing this, we shall spend the remainder of this section discussing how we can learn rich function classes for $p(X,Y|\mathcal{H})$ and $q(Y|X,\mathcal{H})$.

When source coding with latent variable models was first proposed by Frey and Hinton [52], the authors assumed a mixture of Gaussians source model. This choice allowed for finding the maximum likelihood solution by application of the Expectation Maximization algorithm. The algorithm iterates between optimizing $p(X,Y|\mathcal{H})$ and $q(Y|X,\mathcal{H})$. In each step, the optimal solution is determined given the other distribution is fixed. The algorithm requires its user to compute $\frac{p(X,Y|\mathcal{H})}{\int p(X,Y|\mathcal{H})dY}$. This is tractable only for a small number of distributions unfortunately.

Since this proposal for learning latent variable models, there have been many advances in the field of inference and estimation using

variational methods that allow for more complex posteriors and joint models [92, 156, 86, 165, 166, 171, 106, 56, 182]. In the 2010s, deep neural networks (DNNs) increased in popularity due to successful application for discriminative tasks. DNNs are a class of parameterized models that can easily be trained by stochastic gradient decent. Hoping the performance gain would carry to generative tasks as well, it has been proposed to use DNNs to learn complex conditional distributions [167, 216, 106]. For this, the DNN takes as input the conditional and returns the parameters for the distribution. For instance $q(y|x,\theta) = q(y|x; DNN(x;\hat{\theta}))$, where $\hat{\theta}$ are the parameters of the DNN. This technique allows for a powerful class of conditional distributions. The problem of finding the optimal distribution has now been shifted to finding the optimal DNN. Luckily, we may use stochastic-gradient decent to identify the best solution. To accomplish this, we separate the stochastic computational node, i.e. our distribution, into a stochastic node and deterministic node, where the latter carries all the parameters, i.e. $q(x;\theta) = f(q(\epsilon);\theta)$. This allows us to compute any expectation $\mathbb{E}_q(x;\theta)[g(x)] = \mathbb{E}_q(\epsilon)[g(f(\epsilon;\theta))]$ with MC sampling. This is also known as the reparameterization trick [106, 171]. By applying this trick, we attain unbiased stochastic gradients of our objective with respect to the variational parameters θ. Thus, this results in a standard optimization problem that may be optimized with stochastic gradient ascent as desired.

Latent variable models that use distributions parameterized by deep neural networks are also known as deep latent variable models (DLVMs). One important advantage of DLVMs is that they can encode complex marginal distributions $p(X|\mathcal{H})$ even with relatively simple conditional distributions $q(Y|X,\mathcal{H})$. Hence, this expressivity makes DLVMs attractive for approximating complex source distributions.

Since their introduction, DLVMs have seen many approaches to fit even more complex latent distributions. Since we are interested in having short codes Y, we want to point out two directions in particular. First, one can improve the decoding process through conditional recurrence. This allows for progressive refinement and spatial attention [155, 212, 181, 59, 154]. Exploiting these correlations will eventually allow for shorter codes. Second, we may learn to transform simple latent approximations to more complex ones by learning invertible representations, also known as flows [170, 211, 139, 104]. Matching the true distribution more closely will allow for a shorter code as well.

So far we have seen how we may use one encoder to encode many messages. However, we can also understand communication in the

opposite direction. We may find a system that communicates one message by sending many partially redundant codes. For example, this might be the case if we have no access to the encoder. Reconstruction algorithms represent an example of such communication [169, 114]. Here we seek to reconstruct a 3D object or scene, i.e. the message, by observing 2D projections, i.e. many noisy encodings. When DNNs learn to reconstruct, we cannot be sure to reconstruct the message correctly. Modelling bias may influence the reconstruction. Reconstructing protein structures from electron microscopes correctly is an important challenge where we need to make sure that the reconstructed 3D structure is precise.This leads us to our third research question:

Research Question 3

Research Question 3: Can the projections caused by a molecule in an electron microscope be interpreted as source compression. Can we reconstruct a protein with local uncertainty bounds?

In answering this research question we refer the reader to Chapter 4 of this thesis.

Bits-back coding for source coding

In section 1.3, we approximated the Bayesian code length for communicating both a model and data by a variational code length. We saw that in contrast to the Bayesian code, we could identify a corresponding coding scheme for some choices of prior and posterior. In this section we will describe a variational coding scheme to achieve the variational code length that can use any choice of posterior and source assumption. The new scheme presented here, generally should allow for tighter bounds than the two schemes presented earlier. For this, we will encode redundant code words by using auxiliary information that we will be able to retrieve.

So, let us consider the variational code in Equation (1.25). We will encode each message X according to the approximate posterior distribution $q(Y|X,\mathcal{H})$,

$$L_{VI}(X;\mathcal{H}) = \mathbb{E}_{q(Y|X,\mathcal{H})}\left[-\log p(Y|\mathcal{H}) - \log p(X|Y,\mathcal{H})\epsilon_x^{|X|}\right]. \quad (1.26)$$

Computationally this is still intractable, we thus use MC samples to retrieve encodings $Y \sim q(Y|X)$. We send encodings Y in this fashion (step 1). Once we are done with that, we also send the data misfit, as before, with a method such as Shannon-Fano codes (step 2).

Note that parts of the code we sent contained no information about the data. The entropy of the approximate posterior distribution quantifies the amount of extra information that we transmit,

$$H_{q(Y|X,\mathcal{H})}(Y|X) = -\mathbb{E}_{q(Y|X,\mathcal{H})}\left[\log q(Y|X,\mathcal{H})\right]. \tag{1.27}$$

This opens two questions: How can the receiver identify this information? How can the receiver use these auxiliary bits?

The receiver is now in possession of messages and encoding, X and Y respectively. From this information, the receiver can construct $q(Y|X,\mathcal{H})$ given it knows the algorithm used to compute the posterior or has shared resources (step 3). It can now identify the auxiliary information bits by sampling the message $Y \sim q(Y|X)$.

This justifies the effective coding length of the variational code to be the same as Equation (1.25), $L_{VI}(X;\mathcal{H}) - H_{q(Y|X,\mathcal{H})}(Y|X)$. From a practical point of view though, it is not clear how the receiver can make use of the auxiliary bits that have already been sent after all. It turns out that when sending many encodings in sequence, the auxiliary bits of one encoding can be used to code the next one. For the first data point, we do need to send the auxiliary bits. However, starting at the second data point, we can use the previously encoded data point's bits as auxiliary bits (assuming they are random). This scheme is also known as 'bits-back with feedback'. This practical bits-back coding was first worked out by Frey and Hinton [52], and has recently been refined [207, 107, 85].

Beyond source coding

In this section, we would like to point out some shortcomings of the modelling choices we have presented so far. First of all, we have looked at sending codes and models separately. However, it should be clear that in a realistic scenario we have to consider both. This results in the following coding length,

$$\begin{aligned} L_{\text{Bayes}}(X|\mathcal{H}) \geq &-\mathbb{E}_{q(\theta|\mathcal{H},\epsilon_\theta^{|\theta|})}\left[\log\left(\frac{p(\theta|\mathcal{H})}{q(\theta|\mathcal{H},\epsilon_\theta^{|\theta|})}\right)\right. \\ &\left.-\mathbb{E}_{q(Y|X,\mathcal{H})}\left[\log\left(\frac{p(Y|\mathcal{H})}{q(Y|X,\mathcal{H})}p(X|Y,\mathcal{H})\epsilon_x^{|X|}\right)\right]\right]. \end{aligned} \tag{1.28}$$

This is especially important when learning with large over parameterized models. Imagine we would want to do source compression without accounting for the model transfer cost. For a finite amount of

messages sent, we could hide all messages in the encoder and decoder. Thus, we would not need to send any code. The communication cost would be zero. In fact, this is a common problem in generative modelling where the latent code is being ignored (observed in e.g. [155] and discussed in [5]).

Secondly, so far we have only payed attention to the source coding part of the communication process instead of considering channel and source coding. We have justified that we can treat those two problems separately by the SCCT. However, it turns out that the theorem is only applicable when a certain set of assumptions is met (see Section 1.1). If this is not the case, we can not assume separate solutions to be optimal. On top of this, even if they were to exist, we may not be able to identify the best model in the model class. This is why pioneering work looks at channel coding and modelling channel and source jointly [147, 62, 23, 44, 100, 101].

One of the most important characteristics of a channel to take into consideration when modelling is the bandwidth-limited channel. This channel describes the situation in which the amount of transmitted information varies, e.g. wifi or radio waves.

Research Question 4

Research Question 4: What are relevant modelling choices when modelling communication systems with the bandwidth-limited channel when the encoder and decoder are large parametric models?

In answering this research question we refer the reader to Chapter 5 of this thesis.

2

List of Publications

As required, I provide a list of publications whose content contributed to this thesis, I will further provide the contributions of each co-author.

Ullrich, Karen, Edward Meeds, and Max Welling. "Soft weight-sharing for neural network compression." *ICLR, 2017*.

Louizos, Christos, Karen Ullrich, and Max Welling. "Bayesian compression for deep learning." *NeurIPS, 2017*.

Federici, Marco, Karen Ullrich, and Max Welling. "Improved Bayesian compression." *Bayesian Deep Learning Workshop NeurIPS, 2017*.

Ullrich, Karen, Rianne van den Berg, Marcus Brubaker, David Fleet, and Max Welling. "Differentiable probabilistic models of scientific imaging with the Fourier slice theorem." *UAI, 2019*.

Ullrich, Karen, Fabio Viola, and Danilo Jimenez Rezende. "Neural Communication Systems with Bandwidth-limited Channel." *under submission, 2020*.

Me, Karen Ullrich, has contributed to all aspects of the above publications. Max Welling has provided insight and advice for all publications. Edward Meeds provided guidance and advice in the publication "Soft weight-sharing for neural network compression.". In the publication "Bayesian compression for deep learning", Christos Louizos provided the majority of the contributions and in particular the imple-

mentation and variations of the model, while I provided insights with respect to compression such as the need for grouping and quantisation, thus I designed the post-processing of the models and evaluation methods. Sections of the original paper written by the first author have been rewritten for the purpose of this thesis. The authors gave permission to mutually use all figures, tables and algorithms of the original work. In the workshop paper "Improved Bayesian compression.", I took an advisory role, Marco Federici contributed in all other aspects. Rianne van den Berg, Marcus Brubaker and David Fleet provided supervision and technical advice in the publication about "Differentiable probabilistic models of scientific imaging with the Fourier slice theorem.". In the publication "Neural Communication Systems with Bandwidth-limited Channel.", Danilo Rezende provided supervision and insights, Fabio Viola provided technical advice.

Part I

MDL principle for model compression

In the first part of this thesis, we focus our efforts on learning simple models as motivated by the MDL principle. We will focus our efforts on simplification through parameter reduction in deep neural networks. This is, of course, not the only way to think about function complexity. We shall elaborate on alternative methods in the final chapter. In the following, we relate the next two chapters to related literature.

Neural network compression

Due to the ever increasing computational demands of recent neural network models, pruning neural network parameters has become a major line of research. In some common architectures, more than 99% of parameters can be removed without significant loss of accuracy. There are a few notable approaches to pruning when given a trained neural network. Some work operates under the assumption that weights with small magnitude, or small first or second derivatives may be removed [119, 71, Han et al., 64, 69, 97, 42]. This approach is usually applied iteratively: weights are removed, then the smaller network is trained again, and optionally, some previously pruned weight may be revived. This cycle repeats until a required pruning rate is achieved. Other pruning criteria include heuristics based on activation [89] or redundancy [137, 197, 225].

The Bayesian community introduced schemes that explicitly optimize the description length of a model alongside the discrimination accuracy, thus leading to sparser models [135, 149, 116, 20, 105, 148, 99, 209]. Notable work, additionally relevant in this context, is work that applies learned noise to learned weights [56, 53, 146, 199, 145, 129, 131, 54, 48, 199, 150]. The impact of this noise on the discrimination ability of the model is used as a learning signal for the noise and weights. As a result, weights that can sustain high levels of noise may be removed.

Beyond pruning single parameters, the community realized that in order to drop computational needs of a neural network it can be more beneficial to prune entire structures such as filters, channels or features [221, 224, 120, 131, 132, 42, 77, 230, 227, 226]. Even though structure pruning might reduce the total number of parameters less than aforementioned methods, it often leads to more computationally efficient architectures. On the other hand, there is work that trains efficiency aware architectures from scratch [90, 43, 88, 9, 189]. The aforementioned work was influenced by earlier work that showed small net-

works can be trained from scratch when learning from samples drawn from an overparameterized network, often referred to as distillation [10, 178, 81].

Vector quantization

It is worth mentioning that good neural network compression results are usually achieved by pruning and a form of vector quantization (VQ). VQ is a classical technique from signal processing that allows for the modeling of probability density functions by the distribution of prototype vectors. Various studies explore k-means quantization [Han et al.], product quantization, residual quantization [55], optimal fixed point [124] and hashing quantization [28]. Most successful in this field is precision quantification where the prototype vectors lay on a regular grid, i.e. the grid of points that can be represented by a 16-bit floating-point number [141, 65, 33, 214, 26].

Binary neural networks

Binary neural networks can be seen as another special case of VQ. A common approach to learning binary (or tertiary) weights is to utilize floating-point gradients during training. Through gradient accumulation, it can be determined when to switch to the other state [32, 140, 168, 233, 31, 158]. Even though binary architectures are promising due to their low computational demands per operation during inference, it is hard to optimize them. Further, they often rely on many more parameters than their continuous counterparts.

Low rank matrix decomposition

A final approach to compressing information is to apply low rank matrix decomposition. This was first introduced by [41] and [94], and elaborated on by using low rank filters [91], low rank regularization [202] or combining low rank decomposition with sparsity [125].

3

Approximate Bayesian Compression for Deep Learning

In this chapter of the thesis, we seek to approximate the Bayesian code of communicating deep neural network models. We do this by removing prediction irrelevant parameters from the model and by lowering the bit-precision of parameters. These procedures are performed subject to the constraint that they do not affect the predictive power of the model. Our scheme thus executes the communication idea that was laid out in the introduction (see Equation (1.20)). Note that reducing precision as a form of vector quantization can be understood, from a Bayesian point of view, to be an ensemble of many high precision models (see Equation (1.17)).

Practically, we determine the parameter precision by investigating the uncertainty learned in the posterior distribution. To prune unnecessary parameters, we rely on a scheme proposed by Kingma et al. [105] and developed by Molchanov et al. [145]. In contrast to their work, we use hierarchical sparsity inducing priors to eliminate *groups of weights* instead individual ones. This bears large computational benefits not for communicating the model, but for executing it with good performance. Both of our inventions significantly improve model compression rates. They also show excellent energy efficiency and speed compared to other methods that have been explicitly designed for this purpose.

3.1 Introduction

While deep neural networks have become extremely successful in in a wide range of applications, often exceeding human performance, they remain difficult to apply in many real world scenarios. For instance, making billions of predictions per day comes with substantial energy costs given the energy consumption of common Graphical Processing Units (GPUs). Also, real-time predictions are often about a factor 100 away in terms of speed from what deep NNs can deliver, and sending NNs with millions of parameters through band limited channels is still impractical. As a result, running them on hardware limited devices such as smart phones, robots or cars requires substantial improvements on all of these issues. For all those reasons, compression and efficiency have become a topic of interest in the deep learning community.

While all of these issues are certainly related, compression and performance optimizing procedures might not always be aligned. As an illustration, consider the convolutional layers of Alexnet, which account for only 4% of the parameters but 91% of the computation [201]. Compressing these layers will not contribute much to the overall memory footprint.

There is a variety of approaches to address these problem settings. However, most methods have the common strategy of reducing both the neural network structure and the effective fixed point precision for each weight. A justification for the former is the finding that NNs suffer from significant parameter redundancy [40]. Methods in this line of thought are network pruning, where unnecessary connections are being removed [119, 68, 64], or student-teacher learning where a large network is used to train a significantly smaller network [10, 81].

In this work, we show that parameter sparsity and vector quantization (aka reducing bit precision) alongside achieving high predictive accuracy is motivated by a Bayesian perspective on learning. Specifically, we will use the MDL principle [63] to justify our actions. The principle has been linked to Bayesian learning through the bits-back argument [80]. As suggested by the authors of the latter, we will use the variational Bayesian approximation to the Bayesian inference method.

More precisely, our method codes with sparsity inducing priors. This induces posteriors that are rewarded for uncertainty, which can be

translated into reduced bit precision. To design our priors, we also keep computational benefits in mind. This means, we employ sparsity inducing priors for hidden units (instead of individual weights). We can thus prune neurons. When this scheme is applied for each layer, it has an effect not just to the outgoing weights of a layer but also to the incoming ones. We use the posterior uncertainty, to identify the required precision of a given neuron. This means that our Bayesian network can be re-interpreted as a low precision network. The result of our pruning method does not require any sampling. It is in fact applicable and practical on chip with current GPUs. We demonstrate this in our experiment section.

3.2 Variational Bayes and Minimum Description Length

A fundamental theorem in information theory is the minimum description length (MDL) principle [63]. It relates to compression directly in that it defines the best hypothesis to be the one that communicates the sum of the model (complexity cost $\mathcal{L}^C$) and the data misfit (error cost $\mathcal{L}^E$) with the minimum number of bits [174, 176]. It is well understood that variational inference can be reinterpreted from an MDL point of view [159, 220, 80, 87, 56]. More specifically, assume that we are presented with a dataset $\mathcal{D}$ that consists from N input-output pairs $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. Let $p(\mathcal{D}|\mathbf{w}) = \prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \mathbf{w})$ be a parametric model, e.g. a deep neural network, that maps inputs $\mathbf{x}$ to their corresponding outputs y using parameters $\mathbf{w}$ governed by a prior distribution $p(\mathbf{w})$. In this scenario, we wish to approximate the intractable posterior distribution $p(\mathbf{w}|\mathcal{D}) = p(\mathcal{D}|\mathbf{w})p(\mathbf{w})/p(\mathcal{D})$ with a fixed form approximate posterior $q_\phi(\mathbf{w})$ by optimizing the variational parameters ϕ according to:

$$\mathcal{L}(\phi) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{w})}[\log p(\mathcal{D}|\mathbf{w})]}_{\mathcal{L}^E} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{w})}[\log p(\mathbf{w})] + \mathcal{H}(q_\phi(\mathbf{w}))}_{\mathcal{L}^C} \tag{3.1}$$

where $\mathcal{H}(\cdot)$ denotes the entropy and $\mathcal{L}(\phi)$ is known as the evidence-lower-bound (ELBO) or negative variational free energy. As indicated in eq. 6.8, $\mathcal{L}(\phi)$ naturally decomposes into a minimum cost for communicating the targets $\{y_n\}_{n=1}^N$ under the assumption that the sender and receiver agreed on a prior $p(\mathbf{w})$ and that the receiver knows the inputs $\{\mathbf{x}_n\}_{n=1}^N$ and form of the parametric model.

By using sparsity inducing priors for groups of weights that feed into

a neuron the Bayesian mechanism will start pruning hidden units that are not strictly necessary for prediction and thus achieving compression. But there is also a second mechanism by which Bayes can help us compress. By explicitly entertaining noisy weight encodings through $q_\phi(\mathbf{w})$ we can benefit from the bits-back argument [80, 87] due to the entropy term; this is in contrast to infinitely precise weights that lead to $\mathcal{H}(\delta(\mathbf{w})) = -\infty$[1]. Nevertheless in practice, the data misfit term $\mathcal{L}^E$ is intractable for neural network models under a noisy weight encoding, so as a solution Monte Carlo integration is usually employed. Continuous $q_\phi(\mathbf{w})$ allow for the reparametrization trick [106, 171]. Here, we replace sampling from $q_\phi(\mathbf{w})$ by sampling from a deterministic function of the variational parameters ϕ and noise variables ϵ:

[1] In practice this term is a large constant determined by the weight precision.

$$\mathcal{L}(\phi) = \mathbb{E}_{p(\epsilon)}[\log p(\mathcal{D}|f(\phi,\epsilon))] + \mathbb{E}_{q_\phi(\mathbf{w})}[\log p(\mathbf{w})] + \mathcal{H}(q_\phi(\mathbf{w})) \quad (3.2)$$

where $\mathbf{w} = f(\phi,\epsilon)$. By applying this trick, we obtain unbiased stochastic gradients of the ELBO with respect to the variational parameters ϕ, thus resulting in a standard optimization problem that is fit for stochastic gradient ascent. The efficiency of the gradient estimator resulting from eq. 3.2 can be further improved for neural networks by utilizing local reparametrizations [105] (which we will use in our experiments); they provide variance reduction in an efficient way by locally marginalizing the weights at each layer and instead sampling the distribution of the pre-activations.

3.3 Bayesian compression with scale mixtures of normals

We consider a network parameter to be a random variable w. Its scale z is characterized by a distribution $p(z)$:

$$z \sim p(z); \qquad w \sim \mathcal{N}(w; 0, z^2) \quad (3.3)$$

where z^2 serves as the variance of the zero-mean normal distribution over w. Because we treat the scale z of the parameter as random variables too, under this parameterization, we may recover marginal prior distributions over the parameters. We bias the distribution over w to be sparse by choosing the appropriate distribution on its scale z to allow for heavier tails and more mass at zero. A corresponding family of distributions are the scale-mixtures of normals [17, 7]. We will present some members of this family in this work. However, the group is general and many sparsity inducing priors can be considered special cases.

One of the most well known members of this family is the spike-and-slab distribution [143], also refereed to as the golden standard for sparse Bayesian inference. Assume the Bernoulli distribution on the parameter scale. The resulting marginal distribution $p(w)$ has a delta "spike" at zero when $z = 0$ and a continuous "slab" over the real line when $z = 1$. Sadly, using this prior leads to computationally expensive inference. We would have to search 2^M models, with M being the number of model parameters. In search for a more effective inference scheme, Dropout [82, 200] can be interpreted as effective spike-and-slab. Specifically, the variance of the "slab" is zero [53, 128]. This comes handy because Dropout is already a popular regularization techniques for neural networks.

A second example of the scale-mixture family is the Laplace distribution. For this the scale is parameterized by an exponential distribution $p(z^2) = \text{Exp}(\lambda)$. The mode of the posterior distribution given a Laplace prior is called the Lasso [205] estimator. Previously, Lasso has been used for sparsifying neural networks [221, 183]. An advantage of the Lasso estimator is that it can be applied with few computational burdens. However, the effect of the Lasso estimator may be to "shrink" large signals [25] and only provide for point estimates over the parameters. The lack of providing uncertainty bounds may lead to over-fitting. The need for potentially precise point estimates leads on top to large networks even after compression.

Because we strive for computationally feasible inference and sparse results, we tackle the problem of group sparsity with an approximate Bayesian approach. We will consider two choices for the scale prior $p(z)$: the hyperparameter free log-uniform prior [105, 145] and the half-Cauchy prior. Latter results into a horseshoe [25] prior distribution. Both of these distributions correspond to a continuous relaxation of the spike-and-slab prior and we provide a brief discussion on their shrinkage properties at the Appendix C.

Reparametrizing variational dropout for group sparsity

We seek to identify a scale prior that allows for group sparsity. A viable candidate is the normal-Jeffreys prior, that has been introduced by [50]. For this we choose $p(z)$ to be the improper log-uniform prior, $p(z) \propto |z|^{-1}$. When we marginalize z to recover the corresponding

weight prior we again obtain a log-uniform prior:

$$p(w) \propto \int \frac{1}{|z|} \mathcal{N}(w|0, z^2) dz = \frac{1}{|w|} \tag{3.4}$$

Equipped with our prior choices we can simply "couple" the scales of weights belonging to the same group (e.g. neuron or feature map). We do so by sharing the corresponding scale variable z in the joint prior:

$$p(\mathbf{W}, \mathbf{z}) \propto \prod_{i}^{A} \frac{1}{|z_i|} \prod_{ij}^{A,B} \mathcal{N}(w_{ij}|0, z_i^2). \tag{3.5}$$

Note that $\mathbf{W}$ is the weight matrix of a fully connected neural network layer, where A is the dimensionality of the inputs and B is the dimensionality of the outputs. Next we specify the form of the approximate posterior distribution, we require for our variational inference scheme. The joint posterior may be parametrized in the following way:

$$q_\phi(\mathbf{W}, \mathbf{z}) = \prod_{i=1}^{A} \mathcal{N}(z_i|\mu_{z_i}, \mu_{z_i}^2 \alpha_i) \prod_{i,j}^{A,B} \mathcal{N}(w_{ij}|z_i \mu_{ij}, z_i^2 \sigma_{ij}^2) \tag{3.6}$$

with α_i being the dropout rate [200, 105, 145] of a given group indexed by i. [105, 145] report that the multiplicative parametrization of the approximate posterior as we chose it suffers from high variance gradients. [145] offer a solution by re-parameterizing the scale such that $\sigma_{z_i}^2 = \mu_{z_i}^2 \alpha_i$. We follow them and thus instead of optimizing the scale we shall optimize w.r.t. $\sigma_{z_i}^2$. The resulting lower bound given our choice of prior and posterior becomes:

$$\begin{aligned} \mathcal{L}(\phi) = &\mathbb{E}_{q_\phi(\mathbf{z}) q_\phi(\mathbf{W}|\mathbf{z})}[\log p(\mathcal{D}|\mathbf{W})] \\ &- \mathbb{E}_{q_\phi(\mathbf{z})}[KL(q_\phi(\mathbf{W}|\mathbf{z})||p(\mathbf{W}|\mathbf{z}))] - KL(q_\phi(\mathbf{z})||p(\mathbf{z})) \end{aligned} \tag{3.7}$$

Surprisingly, under our choices, the KL-divergence between the conditional prior $p(\mathbf{W}|\mathbf{z})$ and posterior $q_\phi(\mathbf{W}|\mathbf{z})$ is independent of $\mathbf{z}$:

$$KL(q_\phi(\mathbf{W}|\mathbf{z})||p(\mathbf{W}|\mathbf{z})) = \frac{1}{2} \sum_{i,j}^{A,B} \left(\log \frac{\cancel{z_i^2}}{\cancel{z_i^2} \sigma_{ij}^2} + \frac{\cancel{z_i^2} \sigma_{ij}^2}{\cancel{z_i^2}} + \frac{\cancel{z_i^2} \mu_{ij}^2}{\cancel{z_i^2}} - 1 \right) \tag{3.8}$$

Let us understand why this is the case. For this let us consider the non-centered parametrization of the prior [157] over the weights; $\tilde{w}_{ij} = \frac{w_{ij}}{z_i}$. The corresponding joint distribution is $p(\mathbf{W}|\mathbf{z})p(\mathbf{z}) = p(\tilde{\mathbf{W}})p(\mathbf{z})$ where $p(\tilde{\mathbf{W}}) = \prod_{i,j} \mathcal{N}(w_{ij}|0, 1)$ and $\mathbf{W} = \text{diag}(\mathbf{z})\tilde{\mathbf{W}}$. Given this trick, we see that we arrive at the same KL-divergence term when performing variational inference under the $p(\tilde{\mathbf{W}})p(\mathbf{z})$ prior and a posterior of the form $q_\phi(\tilde{\mathbf{W}}, \mathbf{z}) = q_\phi(\tilde{\mathbf{W}}) q_\phi(\mathbf{z})$, where $q_\phi(\tilde{\mathbf{W}}) = \mathcal{N}(\tilde{w}_{ij}|\mu_{ij}, \sigma_{ij}^2)$.

Finally, we compute the KL-divergence between the normal-Jeffreys scale prior $p(\mathbf{z})$ and Gaussian posterior $q_\phi(\mathbf{z})$. We follow [145] in

approximating its precise form by;

$$KL(q_\phi(\mathbf{z})||p(\mathbf{z})) \approx \sum_i^A \big(k_1\sigma(k_2 + k_3 \log \alpha_i) - 0.5m(-\log \alpha_i) - k_1\big) \tag{3.9}$$

where $\sigma(\cdot)$, $m(\cdot)$ are the sigmoid and softplus functions respectively[2]. The k-parameters are set as follows; $k_1 = 0.63576$, $k_2 = 1.87320$, $k_3 = 1.48695$. Note that the divergence depends on the "implied" dropout rate, $\alpha_i = \sigma^2_{z_i}/\mu^2_{z_i}$, only.

[2] $\sigma(x) = (1 + \exp(-x))^{-1}$, $m(x) = \log(1 + \exp(x))$

We will prune a group of parameters that have high dropout rates. For this we specify a threshold at which we remove the corresponding groups of parameters, i.e., $\log \alpha_i = (\log \sigma^2_{z_i} - \log \mu^2_{z_i}) \geq t$. Note that our prior parametrization allows generally for more flexible marginal posteriors over the weights because we have a compound distribution, $q_\phi(\mathbf{W}) = \int q_\phi(\mathbf{W}|\mathbf{z})q_\phi(\mathbf{z})d\mathbf{z}$. This contrasts previous work by by [105, 145].

Another measure we take is to mask the posterior mean at test time. This guarantees that we have a single feed-forward pass, where at each layer we replace the distribution over $\mathbf{W}$ with a single weight matrix: The masked posterior

$$\hat{\mathbf{W}} = \text{diag}(\mathbf{m}) \odot \mathbb{E}_{q(\mathbf{z})q(\tilde{\mathbf{W}})}[\text{diag}(\mathbf{z})\tilde{\mathbf{W}}] = \text{diag}\left(\mathbf{m} \odot \boldsymbol{\mu}_z\right)\mathbf{M}_W \tag{3.10}$$

carries a binary mask $\mathbf{m}$ that os determined according to the group variational dropout rate and $\mathbf{M}_W$, the means of $q_\phi(\tilde{\mathbf{W}})$.

Group horseshoe with half-Cauchy scale priors

In the previous section, we have investigated the effect of the normal-Jeffreys prior to our variational inference compression scheme. In this section we shall investigate an alternative choice for $p(z)$: the proper half-Cauchy distribution; $\mathcal{C}^+(0, s) = 2(s\pi(1 + (z/s)^2))^{-1}$. This leads to the following the prior hierarchy over the weights (in non-centered parametrization);

$$\begin{aligned} s \sim \mathcal{C}^+(0, \tau_0); \qquad \tilde{z}_i \sim \mathcal{C}^+(0, 1); \\ \tilde{w}_{ij} \sim \mathcal{N}(0, 1); \qquad w_{ij} = \tilde{w}_{ij}\tilde{z}_i s \end{aligned} \tag{3.11}$$

with τ_0 being the free parameter which may be tuned for a specific desiderata. The half-Cauchy scale prior induces a horseshoe prior [25] over the weights. Again this is an already well known sparsity inducing prior referenced in the statistics literature. The intuition for the

horseshoe prior is the one of "global-local" shrinkage. More precisely, the (global) scale variable s rewards all of the variables for striving towards zero. On the other hand, the heavy tailed (local) variables z_i compensates this behaviour thus allowing for some weights to escape.

If we were to apply the half-Cauchy priors directly, we could not compute the KL-divergence from the scale prior $p(\mathbf{z})$ to a log-normal scale posterior $q_\phi(\mathbf{z})$ in closed form. We thus chose a re-parameterization by decomposing the half-Cauchy with (inverse) gamma distributions according to [151]. We detail the derivation in in Appendix D. This results in the half-Cauchy prior to be expressed in a non-centered parametrization:

$$p(\tilde{\beta}) = \mathcal{IG}(0.5, 1); \qquad p(\tilde{\alpha}) = \mathcal{G}(0.5, k^2); \qquad z^2 = \tilde{\alpha}\tilde{\beta} \tag{3.12}$$

where $\mathcal{IG}()$ and $\mathcal{G}()$ denote the inverse Gamma and Gamma distributions, respectively. z shall follow a half-Cauchy with scale parameter k. Let us express the prior hierarchy (see Equation (3.12)) again:

$$\begin{aligned} &s_b \sim \mathcal{IG}(0.5, 1); \quad s_a \sim \mathcal{G}(0.5, \tau_0^2); \quad \tilde{\beta}_i \sim \mathcal{IG}(0.5, 1); \\ &\tilde{\alpha}_i \sim \mathcal{G}(0.5, 1); \quad \tilde{w}_{ij} \sim \mathcal{N}(0, 1); \qquad w_{ij} = \tilde{w}_{ij}\sqrt{s_a s_b \tilde{\alpha}_i \tilde{\beta}_i} \end{aligned} \tag{3.13}$$

Note that, we can relate the improper log-uniform prior from the previous section to this prior. When the shapes of the (inverse) Gamma hyper-priors on $\tilde{\alpha}_i, \tilde{\beta}_i$ goes to zero the horseshoe prior becomes the log-uniform prior [25]. More generally, we can express many shrinkage priors as horseshoe when altering shapes of the (inverse) Gamma hyper-priors [8].

Next, we choose a corresponding approximate posterior that allows for computable KL-divergences:

$$\begin{aligned} q_\phi(s_b, s_a, \tilde{\boldsymbol{\beta}}) =& \mathcal{LN}(s_b|\mu_{s_b}, \sigma^2_{s_b})\mathcal{LN}(s_a|\mu_{s_a}, \sigma^2_{s_a}) \\ & \prod_i^A \mathcal{LN}(\tilde{\beta}_i|\mu_{\tilde{\beta}_i}, \sigma^2_{\tilde{\beta}_i}) \end{aligned} \tag{3.14}$$

$$q_\phi(\tilde{\boldsymbol{\alpha}}, \tilde{\mathbf{W}}) = \prod_i^A \mathcal{LN}(\tilde{\alpha}_i|\mu_{\tilde{\alpha}_i}, \sigma^2_{\tilde{\alpha}_i}) \prod_{i,j}^{A,B} \mathcal{N}(\tilde{w}_{ij}|\mu_{\tilde{w}_{ij}}, \sigma^2_{\tilde{w}_{ij}}) \tag{3.15}$$

where $\mathcal{LN}(\cdot, \cdot)$ is a log-normal distribution. Furthermore, it is important to mention that we apply the local reparametrization trick [105] as it reduces variance. In particular, we sample $\sqrt{\tilde{\alpha}_i\tilde{\beta}_i}$ and $\sqrt{s_a s_b}$ [3], such

[3] This exploits the fact that the product of log-normal r.v.s is again a log-normal, furhter the power of a log-normal r.v. is also a log-normal.

that;

$$\tilde{z}_i = \sqrt{\tilde{\alpha}_i \tilde{\beta}_i} \sim \mathcal{LN}(\mu_{\tilde{z}_i}, \sigma^2_{\tilde{z}_i}); \qquad s = \sqrt{s_a s_b} \sim \mathcal{LN}(\mu_s, \sigma^2_s) \quad (3.16)$$

$$\mu_{\tilde{z}_i} = \frac{1}{2}(\mu_{\tilde{\alpha}_i} + \mu_{\tilde{\beta}_i}); \qquad \sigma^2_{\tilde{z}_i} = \frac{1}{4}(\sigma^2_{\tilde{\alpha}_i} + \sigma^2_{\tilde{\beta}_i}); \quad (3.17)$$

$$\mu_s = \frac{1}{2}(\mu_{s_a} + \mu_{s_b}); \qquad \sigma^2_s = \frac{1}{4}(\sigma^2_{s_a} + \sigma^2_{s_b}) \quad (3.18)$$

We will again use a thresholding method to achieve group pruning. Specifically, we will use the negative log-mode[4] of the local log-normal r.v. $z_i = s\tilde{z}_i$;

$$p(z_i) = \mathcal{LN}(z_i | \mu_{z_i}, \sigma^2_{z_i}); \qquad \mu_{z_i} = \mu_{\tilde{z}_i} + \mu_s; \qquad \sigma^2_{z_i} = \sigma^2_{\tilde{z}_i} + \sigma^2_s. \quad (3.19)$$

In other words, we prune when $(\sigma^2_{z_i} - \mu_{z_i}) \geq t$. Even though, our proposal ignores dependencies induced by the common scale s among the z_i elements, we nonetheless found convincing results in practice.

[4] In prior experiments, we found that the mode has similar behavior to the negative log-mean, $-(\mu_{z_i} + \frac{1}{2}\sigma^2_{z_i})$, but it is slightly better in separating the scales.

Equivalently to the group normal-Jeffreys prior, at test time, we replace the distribution over $\mathbf{W}$ with the masked posterior mean:

$$\begin{aligned} \hat{\mathbf{W}} &= \text{diag}(\mathbf{m}) \odot \mathbb{E}_{q(\mathbf{z})q(\tilde{\mathbf{W}})}[\text{diag}(\mathbf{z})\tilde{\mathbf{W}}] \\ &= \text{diag}\left(\mathbf{m} \odot \exp(\boldsymbol{\mu}_z + \frac{1}{2}\boldsymbol{\sigma}^2_z)\right)\mathbf{M}_W \end{aligned} \quad (3.20)$$

again $\mathbf{m}$ determines binary mask computed according to the threshold t, $\mathbf{M}_W$ represent the means of $q(\tilde{\mathbf{W}})$ and $\boldsymbol{\mu}_z, \boldsymbol{\sigma}^2_z$ denote the means and variances of the local log-normals over z_i.

3.4 Experiments

We validated the compression and speed-up capabilities of our models on the well-known architectures of LeNet-300-100 [117], LeNet-5-Caffe[5] on MNIST [118] and, similarly with [145], VGG [192][6] on CIFAR 10 [112]. The groups of parameters were constructed by coupling the scale variables for each filter for the convolutional layers and for each input neuron for the fully connected layers. The precise algorithms that cover the forward pass through our networks for fully connected and convolutional network layers are depicted in Appendix F. When using the horseshoe prior, the scale τ_0 of the global half-Cauchy prior is set to a sensibly small value such as $\tau_0 = 1e-5$. This measure increases the prior mass at zero further, an essential trick for the sparse estimation and compression of our networks. Further we constrain

[5] `https://github.com/BVLC/caffe/tree/master/examples/mnist`

[6] The actual architecture we used is the adapted CIFAR 10 version described at `http://torch.ch/blog/2015/07/30/cifar.html`.

the standard deviations as described in [130]. We use “warm-up" [196] to avoid bad local optima when optimizing our the variational objective. Any further details of our experimental setup are described in Appendix A. To determine the pruning threshold, we manually inspect the distribution on of dropout rates. Usually we find two well separated clusters (one with signal other with noise). A sample visualization of our method for threshold determination is visualized in Appendix E.

Architecture learning & bit precisions

We first demonstrate the effectiveness of our method by comparing the two presented priors (BC-GNJ and BC-GHS) against other Baysian sparsification techniques. The results of our experiments are summarized in Table 3.1. From the scales, we infer the corresponding bit precision for each layers. We obtain them by computing the marginal posterior variances of each weight, [7]:

$$\mathbb{V}(w_{ij})_{NJ} = \sigma_{z_i}^2(\sigma_{ij}^2 + \mu_{ij}^2) + \sigma_{ij}^2\mu_{z_i}^2 \tag{3.21}$$

$$\mathbb{V}(w_{ij})_{HS} = (\exp(\sigma_{z_i}^2) - 1)\exp(2\mu_{z_i} + \sigma_{z_i}^2)(\sigma_{ij}^2 + \mu_{ij}^2) + \sigma_{ij}^2\exp(2\mu_{z_i} + \sigma_{z_i}^2) \tag{3.22}$$

[7] Because of the non-centered parametrization, we can easily compute: $\mathbb{V}(w_{ij}) = \mathbb{V}(z_i\tilde{w}_{ij}) = \mathbb{V}(z_i)(\mathbb{E}[\tilde{w}_{ij}]^2 + \mathbb{V}(\tilde{w}_{ij})) + \mathbb{V}(\tilde{w}_{ij})\mathbb{E}[z_i]^2$.

We approximate the mean variance of each layer to be the unit round-off necessary in order to represent all weights. We can compute the amount of significant bits from the unit round-off. We further add three exponent bit and one sign bit to estimate the final bit-precision of any layer. We detail this further in Appendix B. We observe our method to compress the LeNet-300-100 and LeNet-5-Caffe architectures significantly more than the Sparse Variational Dropout (VD) [145], Generalized Dropout (GD) [198] or Group Lasso (GL) [221] methods. On the VGG architecture, our method reduces the primarily the later layers from 512 to around 10 feature maps per layer. On top of the architecture reduction, all Bayesian compression methods we consider here reduce the standard 32-bit precision per layer drastically to up to sometimes only 5 bit precision.

Compression Rates

For the actual compression task we compare our method to current work in three different scenarios: (i) compression achieved only by pruning, here, for non-group methods we use the CSC format to store

Network & size	Method	Pruned architecture	Bit-precision
LeNet-300-100	Sparse VD	512-114-72	8-11-14
784-300-100	BC-GNJ	278-98-13	8-9-14
	BC-GHS	311-86-14	13-11-10
LeNet-5-Caffe	Sparse VD	14-19-242-131	13-10-8-12
	GD	7-13-208-16	-
20-50-800-500	GL	3-12-192-500	-
	BC-GNJ	8-13-88-13	18-10-7-9
	BC-GHS	5-10-76-16	10-10–14-13
VGG	BC-GNJ	63-64-128-128-245-155-63- -26-24-20-14-12-11-11-15	10-10-10-10-8-8-8- -5-5-5-5-5-6-7-11
(2× 64)-(2× 128)- -(3×256)-(8× 512)	BC-GHS	51-62-125-128-228-129-38- -13-9-6-5-6-6-6-20	11-12-9-14-10-8-5- -5-6-6-6-8-11-17-10

Table 3.1: We present sparsified architectures as inferred by Sparse VD [145], Generalized Dropout (GD) [198] and Group Lasso (GL) [221], Bayesian Compression (BC) with group normal-Jeffreys (GNJ) and group horseshoe (GHS) priors. The latter two are proposed in this work. The original architecture is displayed in the first column along with the architecture name. Further we present the amount of neurons after pruning in column 3 and the the average bit precisions of the weights per layer in column 4.

parameters; (ii) compression based on the former but with reduced bit precision per layer (only for the weights); and (iii) the maximum compression rate as proposed by [68]. We believe these to be relevant

			Compression Rates (Error %)		
Model Original Error %	Method	$\frac{\lvert\mathbf{w}\neq 0\rvert}{\lvert\mathbf{w}\rvert}$ %	Pruning	Fast Prediction	Maximum Compression
LeNet-300-100	DC	8.0	6 (1.6)	-	40 (1.6)
	DNS	1.8	28* (2.0)	-	-
1.6	SWS	4.3	12* (1.9)	-	64(1.9)
	Sparse VD	2.2	21(1.8)	84(1.8)	113 (1.8)
	BC-GNJ	10.8	9(1.8)	36(1.8)	58(1.8)
	BC-GHS	10.6	9(1.8)	23(1.9)	59(2.0)
LeNet-5-Caffe	DC	8.0	6*(0.7)	-	39(0.7)
	DNS	0.9	55*(0.9)	-	108(0.9)
0.9	SWS	0.5	100*(1.0)	-	162(1.0)
	Sparse VD	0.7	63(1.0)	228(1.0)	365(1.0)
	BC-GNJ	0.9	108(1.0)	361(1.0)	573(1.0)
	BC-GHS	0.6	156(1.0)	419(1.0)	771(1.0)
VGG	BC-GNJ	6.7	14(8.6)	56(8.8)	95(8.6)
8.4	BC-GHS	5.5	18(9.0)	59(9.0)	116(9.2)

Table 3.2: Compression results for our methods. "DC" corresponds to Deep Compression method introduced at [68], "DNS" to the method of [64] and "SWS" to the Soft-Weight Sharing of [209]. Numbers marked with * are best case guesses.

scenarios because (i) can be applied with already existing frameworks such as Tensorflow [1], (ii) is a practical scheme given upcoming GPUs and frameworks will be designed to work with low and mixed precision arithmetics [123, 66]. For (iii), we perform k-means clustering on the weights with k=32 and consequently store a weight index that points to a codebook of available weights. Note that the latter achieves highest compression rate but it is however fairly unpractical at test time since the original matrix needs to be restored for each layer. As we can observe at Table 3.2, our methods are competitive with the state-of-the art for LeNet-300-100 while offering significantly better compression rates on the LeNet-5-Caffe architecture, without any loss in accuracy. Do note that group sparsity and weight sparsity can be combined so as to further prune some weights when a particular group is

not removed, thus we can potentially further boost compression performance at e.g. LeNet-300-100. For the VGG network we observe that training from a random initialization yielded consistently less accuracy (around 1%-2% less) compared to initializing the means of the approximate posterior from a pretrained network, similarly with [145], thus we only report the latter results[8]. After initialization we trained the VGG network regularly for 200 epochs using Adam with the default hyperparameters. We observe a small drop in accuracy for the final models when using the deterministic version of the network for prediction, but nevertheless averaging across multiple samples restores the original accuracy. Note, that in general we can maintain the original accuracy on VGG without sampling by simply finetuning with a small learning rate, as done at [145]. This will still induce (less) sparsity but unfortunately it does not lead to good compression as the bit precision remains very high due to not appropriately increasing the marginal variances of the weights.

[8] We also tried to finetune the same network with Sparse VD, but unfortunately it increased the error considerably (around 3% extra error), therefore we do not report those results.

Speed and energy consumption

We demonstrate that our method is competitive with Wen et al. [221], denoted as GL, a method that explicitly prunes convolutional kernels to reduce compute time. We measure the time and energy consumption of one forward pass of a mini-batch with batch size 8192 through LeNet-5-Caffe. We average over 10^4 forward passes and all experiments were run with Tensorflow 1.0.1, cuda 8.0 and respective cuDNN. We apply 16 CPUs run in parallel (CPU) or a Titan X (GPU). Note that we only use the pruned architecture as lower bit precision would further increase the speed-up but is not implementable in any common framework. Further, all methods we compare to in the latter experiments would barely show an improvement at all since they do not learn to prune groups but only parameters. In figure 3.1 we present our results. As to be expected the largest effect on the speed up is caused by GPU usage. However, both our models and best competing models reach a speed up factor of around $8\times$. We can further save about $3\ \times$ energy costs by applying our architecture instead of the original one on a GPU. For larger networks the speed-up is even higher: for the VGG experiments with batch size 256 we have a speed-up factor of $51\times$.

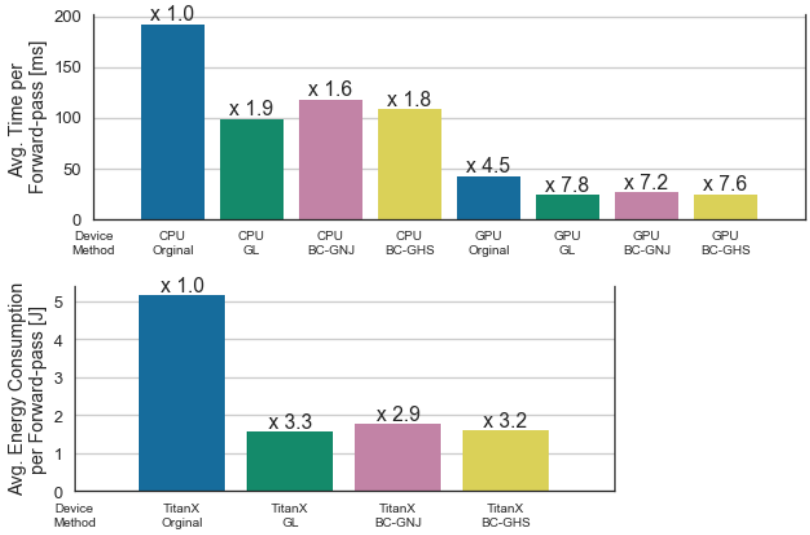

Figure 3.1: **Top:** Avg. Time a batch of 8192 samples takes to pass through LeNet-5-Caffe. Numbers on top of the bars represent speed-up factor relative to the CPU implementation of the original network. **Bottom:** Energy consumption of the GPU of the same process (when run on GPU).

3.5 Conclusion

We showed how we can obtain group sparsity in neural networks in a Bayesian way through scale mixtures of normals. The Log-uniform prior arises through weighted likelihood bootstrap [152], therefore has strong ties to frequentist statistics. The normal-Jeffreys / log-uniform prior assumes independence of the groups a-priori whereas the horseshoe prior assumes mild dependence due to the shared global scale variable s, which can subsequently allow for a more informative selection of groups.

As for future work; we plan to evaluate uncertainty estimates of the pruned networks, using an experimental procedure akin to [130]. We can further improve upon variational Bayesian compression schemes of neural networks by improving the posterior approximation; according to the MDL principle the best noisy weight encoding is the true posterior distribution.

4

Soft Weight-Sharing for Neural Network Compression

In this chapter, we investigate another variational inference based approximation to the Bayesian code. The scheme we propose uses a mixture prior and a Dirac posterior, also known as "soft weight-sharing" [153]. Due to the posterior choice, our scheme can be realized through a two part code. In that, it contrasts bits-back codes. The Dirac posterior will be 'assigned' to the mixture component that best describes it. The expected code length per parameter is thus $log(p(i))$, where $p(i)$ is the probability of a given component of the mixture prior. Note that we chose a Dirac posterior such that we can guarantee that only one component of the mixture claims responsibility for a given parameter.

Practically, we show that competitive compression rates can be achieved. We aid the bits-back coding by removing any redundant model parameters. We do this by making appropriate choices in the model prior.

4.1 Introduction

"Bigger is better" is the ruling maxim in deep learning land. Deep neural nets with billions of parameters are no longer an exception. Networks of such size are unfortunately not practical for mobile, on-device applications which face strong limitations with respect to memory and energy consumption. Compressing neural networks could not only improve memory and energy consumption, but also lead to less network bandwidth, faster processing and better privacy. It has been shown that large networks are heavily over-parametrized and can be compressed by approximately two orders of magnitude without significant loss of accuracy. Apparently, over-parametrization is beneficial for optimization, but not necessary for accurate prediction. This observation has opened the door for a number of highly successful compression algorithms, which either train the network from scratch [81, 90, 31, 34] or apply compression post-optimization [68, 67, 64, 28, 221].

It has been long known that compression is directly related to (variational) Bayesian inference and the minimum description principle [83]. One can show that good compression can be achieved by encoding the parameters of a model using a good prior and specifying the parameters up to an uncertainty given, optimally, by the posterior distribution. An ingenious bits-back argument can then be used to get a refund for using these noisy weights. A number of papers have appeared that encode the weights of a neural network with limited precision (say 8 bits per weight), effectively cashing in on this "bits-back" argument [65, 33, 214]. Some authors go so far of arguing that even a single bit per weight can be used without much loss of accuracy [32, 31].

In this work we follow a different but related direction, namely to learn the prior that we use to encode the parameters. In Bayesian statistics this is known as empirical Bayes. To encourage compression of the weights to K clusters, we fit a mixture of Gaussians prior model over the weights. This idea originates from the nineties, known as soft weight-sharing [153] where it was used to regularize a neural network. Here our primary goal is network compression, but as was shown in [83] these two objectives are almost perfectly aligned. By fitting the mixture components alongside the weights, the weights tend to concentrate very tightly around a number of cluster components, while the cluster centers optimize themselves to give the network high predictive accuracy. Compression is achieved because we only need to

encode K cluster means (in full precision) in addition to the assignment of each weight to one of these J values (using $\log(J)$ bits per weight). We find that competitive compression rates can be achieved by this simple idea.

4.2 MDL View on Variational Learning

Model compression was first discussed in the context of information theory. The minimum description length (MDL) principle identifies the best hypothesis to be the one that best compresses the data. More specifically, it minimizes the cost to describe the model (complexity cost $\mathcal{L}^C$) and the misfit between model and data (error cost $\mathcal{L}^E$) [174, 176]. It has been shown that variational learning can be reinterpreted as an MDL problem [220, 83, 87, 56]. In particular, given data $\mathcal{D} = \{\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{T} = \{\mathbf{t}_n\}_{n=1}^N\}$, a set of parameters $\mathbf{w} = \{w_i\}_{i=1}^I$ that describes the model and an approximation $q(\mathbf{w})$ of the posterior $p(\mathbf{w}|\mathcal{D})$, the variational lower bound, also known as negative variational free energy, $\mathcal{L}(q(\mathbf{w}), \mathbf{w})$ can be decomposed in terms of error and complexity losses

$$\begin{aligned}\mathcal{L}(q(\mathbf{w}), \mathbf{w}) &= -\mathbb{E}_{q(\mathbf{w})}\left[\log\left(\frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{q(\mathbf{w})}\right)\right] \\ &= \underbrace{\mathbb{E}_{q(\mathbf{w})}\left[-\log p(\mathcal{D}|\mathbf{w})\right]}_{\mathcal{L}^E} + \underbrace{\mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}))}_{\mathcal{L}^C}\end{aligned} \tag{4.1}$$

where $p(\mathbf{w})$ is the prior over $\mathbf{w}$ and $p(\mathcal{D}|\mathbf{w})$ is the model likelihood. According to Shannon's source coding theorem, $\mathcal{L}^E$ lower bounds the expected amount of information needed to communicate the targets $\mathbf{T}$, given the receiver knows the inputs $\mathbf{X}$ and the model $\mathbf{w}$. The functional form of the likelihood term is conditioned by the target distribution. For example, in case of regression the predictions of the model are assumed be normally distributed around the targets $\mathbf{T}$.

$$p(\mathcal{D}|\mathbf{w}) = p(\mathbf{T}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{t}_n|\mathbf{x}_n, \mathbf{w}) \tag{4.2}$$

where $\mathcal{N}(\mathbf{t}_n, \mathbf{x}_n, \mathbf{w})$ is a normal distribution. Another typical example is classification where the conditional distribution of targets given data is assumed to be Bernoulli distributed[1]. These assumptions eventually lead to the well known error functions, namely cross-entropy error and squared error for classification and regression, respectively.

[1] For more detailed discussion see [19].

Before however we can communicate the data we first seek to communicate the model. Similarly to $\mathcal{L}^E$, $\mathcal{L}^C$ is a lower bound for transmitting

the model. More specifically, if sender and receiver agree on a prior, $\mathcal{L}^C$ is the expected cost of communicating the parameters $\mathbf{w}$. This cost is again twofold,

$$\mathrm{KL}(q(\mathbf{w})||p(\mathbf{w})) = \mathbb{E}_{q(\mathbf{w})}\left[-\log p(\mathbf{w})\right] - H(q(\mathbf{w})) \tag{4.3}$$

where $H(\cdot)$ denotes the entropy. In [220] and [83] it was shown that noisy encoding of the weights can be beneficial due to the bits-back argument if the uncertainty does not harm the error loss too much. The number of bits to get refunded by an uncertain weight distribution $q(\mathbf{w})$ is given by its entropy. Further, it can be shown that the optimal distribution for $q(\mathbf{w})$ is the Bayesian posterior distribution. While bits-back is proven to be an optimal coding scheme [87], it is often not practical in real world settings. A practical way to cash in on noisy weights (or bits-back) is to only encode a weight value up to a limited number of bits. To see this, assume a factorized variational posterior $q(\mathbf{w}) = \prod q(w_i)$. Each posterior $q(w_i)$ is associated with a Dirac distribution up to machine precision, for example, a Gaussian distribution with variance σ, for small values of σ. This implies that we formally incur a very small refund per weight,

$$\begin{aligned} H(q(\mathbf{w})) &= -\int_{\Omega} q(\mathbf{w}) \log q(\mathbf{w}) \, \mathrm{d}\mathbf{w} \\ &= -\int_{\mathbb{R}^I} \mathcal{N}(\mathbf{w}|\mu, \sigma\mathbf{I}) \log \mathcal{N}(\mathbf{w}|\mu, \sigma\mathbf{I}) = [0.5 \cdot \log(2\pi e\sigma^2)]^I. \end{aligned} \tag{4.4}$$

Note that the more coarse the quantization of weights the more compressible the model. The bits-back scheme makes three assumptions: (i) weights are being transmitted independently, (ii) weights are independent of each other (no mutual information), and (iii) the receiver knows the prior. Han et al. [67] show that one can successfully exploit (i) and (ii) by using a form of arithmetic coding [222]. In particular, they employ range coding schemes such as the Sparse Matrix Format (discussed in Appendix A). This is beneficial because the weight distribution has low entropy. Note that the cost of transmitting the prior should be negligible. Thus a factorized prior with different parameters for each factor is not desirable.

The main objective of this work is to find a suitable prior for optimizing the cross-entropy between a delta posterior $q(\mathbf{w})$ and the prior $p(\mathbf{w})$ while at the same time keeping a practical coding scheme in mind. Recall that the cross entropy is a lower bound on the average number of bits required to encode the weights of the neural network (given infinite precision). Following [153] we will model the prior $p(\mathbf{w})$ as a

mixture of Gaussians,

$$p(\mathbf{w}) = \prod_{i=1}^{I} \sum_{j=0}^{J} \pi_j \mathcal{N}(w_i | \mu_j, \sigma_j^2). \tag{4.5}$$

We learn the mixture parameters μ_j, σ_j, π_j via maximum likelihood simultaneously with the network weights. This is equivalent to an empirical Bayes approach in Bayesian statistics. For state-of-the-art compression schemes pruning plays a major role. By enforcing an arbitrary "zero" component to have fixed $\mu_0 = 0$ location and π_0 to be close to 1, a desired weight pruning rate can be enforced. In this scenario π_0 may be fixed or trainable. In the latter case a Beta distribution as hyper-prior might be helpful. The approach naturally encourages quantization because in order to optimize the cross-entropy the weights will cluster tightly around the cluster means, while the cluster means themselves move to some optimal location driven by $\mathcal{L}^E$. The effect might even be so strong that it is beneficial to have a Gamma hyper-prior on the variances of the mixture components to prevent the components from collapsing. Furthermore, note that, mixture components merge when there is not enough pressure from the error loss to keep them separated because weights are attracted by means and means are attracted by weights hence means also attract each other. In that way the network learns how many quantization intervals are necessary. We demonstrate that behaviour in Figure 4.3.

4.3 Method

This section presents the procedure of network compression as applied in the experiment section. A summary can be found in Algorithm 1.

General set-up

We retrain pre-trained neural networks with soft weight-sharing and factorized Dirac posteriors. Hence we optimize

$$\begin{aligned} \mathcal{L}(\mathbf{w}, \{\mu_j, \sigma_j, \pi_j\}_{j=0}^{J}) &= \mathcal{L}^E + \tau \mathcal{L}^C \\ &= -\log p(\mathbf{T}|\mathbf{X}, \mathbf{w}) \\ &\quad - \tau \log p(\mathbf{w}, \{\mu_j, \sigma_j, \pi_j\}_{j=0}^{J}), \end{aligned} \tag{4.6}$$

via gradient descent, specifically using Adam [102]. The KL divergence reduces to the prior because the entropy term does not depend on any trainable parameters. Note that, similar to [153] we weigh the log-prior contribution to the gradient by a factor of $\tau = 0.005$. In the process of retraining the weights, the variances, means, and mixing proportions of all but one component are learned. For one component, we fix $\mu_{j=0} = 0$ and $\pi_{j=0} = 0.999$. Alternatively we can train $\pi_{j=0}$ as well but restrict it by a Beta distribution hyper-prior. Our Gaussian MM prior is initialized with $2^4 + 1 = 17$ components. We initialize the learning rate for the weights and means, log-variances and log-mixing proportions separately. The weights should be trained with approximately the same learning rate used for pre-training. The remaining learning rates are set to $5 \cdot 10^{-4}$. Note that this is a very sensitive parameter. The Gaussian mixtures will collapse very fast as long as the error loss does not object. However if it collapses too fast weights might be left behind, thus it is important to set the learning rate such that the mixture does collapse too soon. If the learning rate is too small the mixture will converge too slowly. Another option to keep the mixture components from collapsing is to apply an Inverse-Gamma hyperprior on the mixture variances.

Initialization of Mixture Model Components

In principle, we follow the method proposed by [153]. We distribute the means of the 16 non-fixed components evenly over the range of the pre-trained weights. The variances will be initialized such that each Gaussian has significant probability mass in its region. A good orientation for setting a good initial variance is weight decay rate the original network has been trained on. The trainable mixing proportions are initialized evenly $\pi_j = (1 - \pi_{j=0})/J$. We also experimented with other approaches such as distributing the means such that each component assumes an equal amount of probability. We did not observe any significant improvement over the simpler initialization procedure.

Post-Processing

After re-training we set each weight to the mean of the component that takes most responsibility for it i.e. we quantize the weights. Before quantizing, however, there might be redundant components as explained in section 4.2. To eliminate those we follow [2] by comput-

ing the KL divergence between all components. For a KL divergence smaller than a threshold, we merge two components as follows

$$\pi_{\text{new}} = \pi_i + \pi_j, \qquad \mu_{\text{new}} = \frac{\pi_i \mu_i + \pi_j \mu_j}{\pi_i + \pi_j}, \qquad \sigma^2_{\text{new}} = \frac{\pi_i \sigma_i^2 + \pi_j \sigma_j^2}{\pi_i + \pi_j} \quad (4.7)$$

for two components with indices i and j.
Finally, for practical compression we use the storage format used in [67] (see Appendix A).

Algorithm 1: *Soft weight-sharing for compression*, our proposed algorithm for neural network model compression. It is divided into two main steps: network re-training and post-processing.

Require: $\tau \leftarrow$ set the trade-off between error and complexity loss
Require: $\Theta \leftarrow$ set parameters for gradient decent scheme such as learning rate or momentum
Require: $\alpha, \beta \leftarrow$ set gamma hyper-prior parameter (optional)
$\mathbf{w} \leftarrow$ initialize network weights with pre-trained network weights
$\theta = \{\mu_j, \sigma_j, \pi_j\}_{j=1}^{J} \leftarrow$ initialize mixture parameters (see Sec. 4.3)
while $\mathbf{w}, \theta$ not converged **do**
$\mathbf{w}, \theta \leftarrow \nabla_{\mathbf{w},\theta} \mathcal{L}^E + \tau \mathcal{L}^C$ update $\mathbf{w}$ and θ with the gradient decent scheme of choice
end while
$\mathbf{w} \leftarrow \underset{\mu_k}{\text{argmax}} \frac{\pi_k \mathcal{N}(\mathbf{w}|\mu_k, \sigma_k)}{\sum \pi_j \mathcal{N}(\mathbf{w}|\mu_j, \sigma_j)}$ compute final weight by setting it to the mean that takes most responsibility (for details see Sec. 4.3)

4.4 Models

We test our compression procedure on two neural network models used in previous work we compare against in our experiments:

1. **LeNet-300-100** an MNIST model described in [117]. As no pre-trained model is available, we train our own, resulting in an error rate of 1.89%.

2. **LeNet-5-Caffe** a modified version of the LeNet-5 MNIST model in [117]. The model specification can be downloaded from the Caffe MNIST tutorial page [2]. As no pre-trained model is available, we train our own, resulting in an error rate of 0.88%.

3. **ResNets** have been invented by [73] and further developed by [76] and [228]. We choose a model version of the latter authors. In accordance with their notation, we choose a network with depth 16, width $k = 4$ and no dropout. This model has 2.7M parameters.

[2] https://github.com/BVLC/caffe/blob/master/examples/mnist/lenet_train_test.prototxt

In our experiments, we follow the authors by using only light augmentation, i.e., horizontal flips and random shifts by up to 4 pixels. Furthermore the data is normalized. The authors report error rates of 5.02% and 24.03% for CIFAR-10 and CIFAR-100 respectively. By reimplementing their model we trained models that achieve errors 6.48% and 28.23%.

4.5 Experiments

Initial experiment

First, we run our algorithm without any hyper-priors, an experiment on LeNet-300-100. In Figure 4.1 we visualise the original distribution over weights, the final distribution over weight and how each weight changed its position in the training process. After retraining, the distribution is sharply peaked around zero. Note that with our procedure the optimization process automatically determines how many weights per layer are pruned. Specifically in this experiment, 96% of the first layer (235K parameter), 90% of the second (30K) and only 18% of the final layer (10K) are pruned. From observations of this and other experiments, we conclude that the amount of pruned weights depends mainly on the number of parameters in the layer rather than its position or type (convolutional or fully connected).

Evaluating the model reveals a compression rate of 64.2. The accuracy of the model does not drop significantly from 0.9811 to 0.9806. However, we do observe that the mixture components eventually collapse, i.e., the variances go to zero. This makes the prior inflexible and the optimization can easily get stuck because the prior is accumulating probability mass around the mixture means. For a weight, escaping from those high probability plateaus is impossible. This motivates the use hyper-priors such as an Inverse-Gamma prior on the variances to essentially lower bound them.

Hyper-parameter tuning using Bayesian optimization

The proposed procedure offers various freedoms: there are many hyper-parameters to optimize, one may use hyper-priors as motivated in the

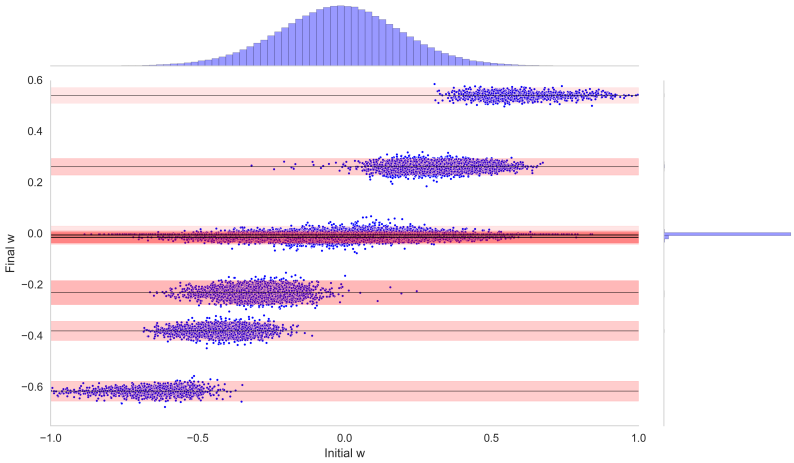

Figure 4.1: On top we show the distribution of a pretrained network. On the right the same distribution after retraining. The change in value of each weight is illustrated by a scatter plot.

previous section or even go as far as using other distributions as mixture components.

To cope with the variety of choices, we optimize 13 hyper-parameters using the Bayesian optimization tool Spearmint [194]. These include the learning rates of the weight and mixing components, the number of components, and τ. Furthermore, we assume an Inverse-Gamma prior over the variances separately for the zero component and the other components and a Beta prior over the zero mixing components.

In these experiments, we optimize re-training hyperparameters for LeNet-300-100 and LeNet-5-Caffe. Due to computational restrictions, we set the number of training epochs to 40 (previously 100), knowing that this may lead to solutions that have not fully converged. Spearmint acts on an objective that balances accuracy loss vs compression rate. The accuracy loss in this case is measured over the training data. The results are shown in Figure 4.2. In the illustration we use the accuracy loss as given by the test data. The best results predicted by our spearmint objective are colored in dark blue. Note that we achieve competitive results in this experiment despite the restricted optimization time of 40 epochs, i.e. 18K updates.

The conclusions from this experiment are a bit unclear, on the one hand we do achieve state-of-the-art results for LeNet-5-Caffe, on the other hand there seems to be little connection between the parameter settings of best results. One wonders if a 13 dimensional parameter space can be searched efficiently with the amount of runs we were conducting. It may be more reasonable to get more inside in the optimization process and tune parameters according to those.

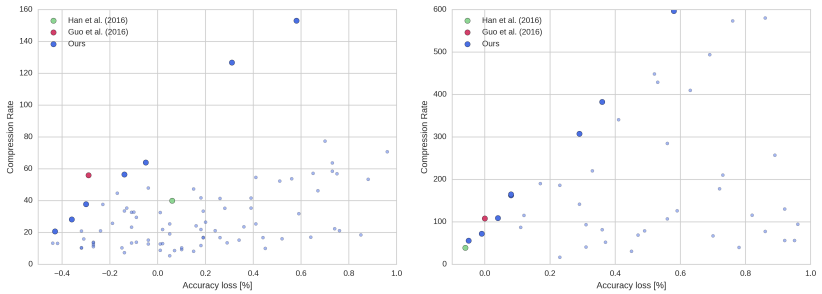

Figure 4.2: We show the results of optimizing hyper-parameters with spearmint. Specifically, we plot the accuracy loss of a re-trained network against the compression rate. Each point represents one hyper-parameter setting. The guesses of the optimizer improve over time. We also present the results of other methods for comparison. **Left**: LeNet-300-100 **Right**: LeNet-5-Caffe.

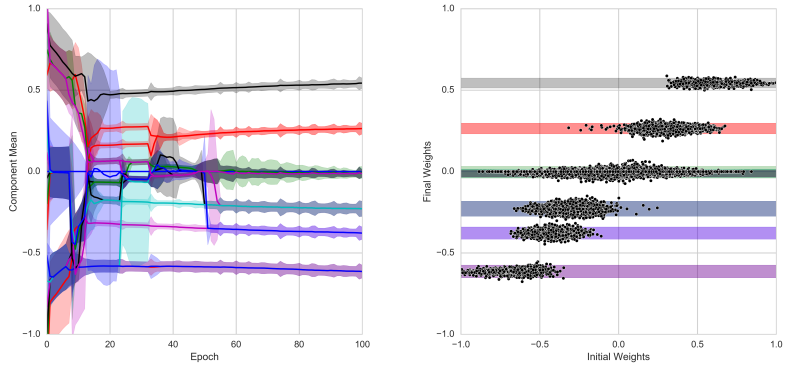

Figure 4.3: Illustration of our mixture model compression procedure on LeNet-5-Caffe. **Left**: Dynamics of Gaussian mixture components during the learning procedure. Initially there are 17 components, including the zero component. During learning components are absorbed into other components, resulting in roughly 6 significant components. **Right**: A scatter plot of initial versus final weights, along with the Gaussian components' uncertainties. The initial weight distribution is roughly one broad Gaussian, whereas the final weight distribution matches closely the final, learned prior which has become very peaked, resulting in good quantization properties.

Compression Results

We compare our compression scheme with [67] and [64] in Table 4.1. The results on MNIST networks are very promising. We achieve state-of-the-art compression rates in both examples. We can furthermore show results for a light version of ResNet with 2.7M parameters to illustrate that our method does scale to modern architectures. We used more components (64) here to cover the large regime of weights. However, for large networks such as VGG with 138M parameters the algorithm is too slow to get usable results. We propose a solution for this problem in Appendix C; however, we do not have any experimental results yet.

Model	Method	Top-1 Error[%]	Δ [%]	$\lvert\mathbf{W}\rvert[10^6]$	$\frac{\lvert\mathbf{W}_{\neq 0}\rvert}{\lvert\mathbf{W}\rvert}$ [%]	CR
LeNet-300-100	[67]	1.64 → 1.58	0.06	0.2	8.0	40
	[64]	2.28 → 1.99	-0.29		1.8	56
	Ours	1.89 → 1.94	-0.05		4.3	**64**
LeNet-5-Caffe	[67]	0.80 → 0.74	-0.06	0.4	8.0	39
	[64]	0.91 → 0.91	0.00		0.9	108
	Ours	0.88 → 0.97	0.09		0.5	**162**
ResNet (light)	Ours	6.48 → 8.50	2.02	2.7	6.6	45

Table 4.1: **Compression Results**. We compare methods based on the post-processing error (we also indicate the starting error), the accuracy loss Δ, the number of non zero weights $\lvert\mathbf{W}_{\neq 0}\rvert$ and the final compression rate CR based on the method proposed by [67].

4.6 Discussion and Future Work

In this work we revived a simple and principled regularization method based on soft weight-sharing and applied it directly to the problem of model compression. On the one hand we showed that we can optimize the MDL complexity lower bound, while on the other hand we showed that our method works well in practice when being applied to different models. A short-coming of the method at the moment is its computational cost and the ease of implementation. For the first, we provide a proposal that will be tested in future work. The latter is an open question at the moment. Note that our method—since it is optimizing the lower bound directly—will most likely also work when applied to other storage formats, such as those proposed originally by [83]. In the future we would like to extend beyond Dirac posteriors as done in [56] by extending the weight sharing prior to more general priors. For example, from a compression point of view, we could learn to prune entire structures from the network by placing Bernoulli priors over structures such as convolutional filters or ResNet units. Furthermore, it could be interesting to train models from scratch or in a student-teacher setting.

Part II

Learning to communicate with DLVMs

In the introduction we laid out how we can communicate latent variables with bits-back coding with feedback. Having this coding scheme in mind, in this section, we shall focus on two applications of communicating latent codes: 3D reconstructions and communication under varying information loss.

Reconstruction algorithms play a major role in science and civil engineering. Given many noisy and incomplete observations of a scene, an object or a process; one seeks to reconstruct this observed item based on this collection of observations. For example, in electron microscopy, an electron ray permits the object of interest to leave a 'shadow' on a sensor detector. The goal of reconstruction then is to reconstruct the true shape or pose of the object based on multiple sensor readings.

Looking at this system as a communication system, we may postulate the object itself as the message source, and the map that causes the projection onto the sensor system as the encoder. In contrast to other communication systems, in reconstruction problems we have no access to the encoding system. We are solely focused on learning to decode or reconstruct the message. Reconstruction algorithms thus constitute an important class of communication systems that we will investigate in Chapter 5.

In Chapter 6, assuming that we have access to bits-back codes and being able to learn encoding and decoding processes, we focus our attention to the question of how to transmit information despite an unpredictable loss of information during the transmission process. This is a problem of significance, for most digital and analog signal transfer.

5

Differentiable probabilistic models of scientific imaging with the Fourier slice theorem

Scientific imaging techniques, e.g. optical and electron microscopy or computed tomography, are used to study 3D structures through 2D observations. These observations are related to the 3D object through orthogonal integral projections. For computational efficiency, common 3D reconstruction algorithms model 3D structures in Fourier space, exploiting the Fourier slice theorem. At present it is somewhat unclear how to differentiate through the projection operator as required by learning algorithms with gradient-based optimization. This paper shows how back-propagation through the projection operator in Fourier space can be achieved. We demonstrate the approach on 3D protein reconstruction. We further extend the approach to learning probabilistic 3D object models. This allows us to predict regions of low sampling rates or to estimate noise. Higher sample efficiency can be reached by utilizing the learned uncertainties of the 3D structure as an unsupervised estimate of model fit. Finally, we demonstrate how the reconstruction algorithm can be extended with amortized inference on unknown attributes such as object pose. Empirical studies show that joint inference of the 3D structure and object pose becomes difficult when the underlying object contains symmetries, in which case pose estimation can easily get stuck in local optima, inhibiting a fine-grained high-quality estimate of the 3D structure.

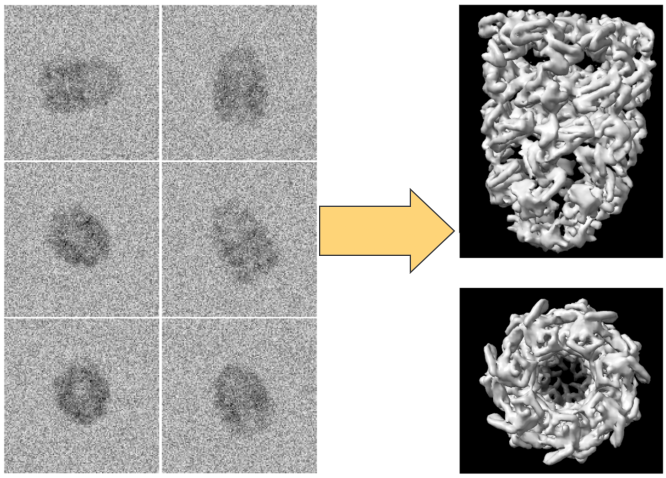

Figure 5.1: Example of electron cryomicroscopy with the GroEL-GroES protein [223]. *Left*: 2D observations obtained by projections with an electron beam. *Right*: Two different views of the ground truth 3D protein structure represented by its electron density.

5.1 Introduction

The main goal of many scientific imaging methods is to reconstruct a $(d+1)$-dimensional structure $\mathbf{v} \in \mathcal{V} \subseteq \mathbb{R}^{D^{d+1}}$ from N (d)-dimensional observations $\mathbf{x}_n \in \mathcal{I} \subseteq \mathbb{R}^{D^d}$, where d is either one or two. For the sake of simplicity we will talk about the case $d = 2$ in the rest of this work. The contributions of this paper are:

1. We view the process of image formation through a graphical model in which latent variables correspond to physical quantities such as the hidden structure $\mathbf{v}$ or the relative orientation/pose of a specimen. This enables one to predict errors in the reconstruction of 3D structures through uncertainty estimates. This is especially interesting when objects $\mathbf{v}$ are only partially observable, as is the case in certain medical scans, such as breast cancer scans. Moreover, uncertainty prediction enables more data efficient model validation.

2. Based on the aforementioned innovations, we propose a new method for (unsupervised) reconstruction evaluation. Particularly, we demonstrate that learned uncertainties can replace currently used data inefficient methods of evaluation (see Section 5.6). We thus learn better model fits than traditional methods given the same amount of data.

3. We extend current approaches such as [95] to describe the generative process as a differentiable map by adopting recent techniques from the deep learning community [93, 169]. We demonstrate that this fenables more advanced joint inference schemes over object

pose and structure estimation.

Our experimental validation focuses on single particle electron cryomicroscopy (cryoEM). CryoEM is a challenging scientific imaging task, as it suffers from complex sources of observation noise, low signal to noise ratios, and interference corruption. Interference corruption attenuates certain Fourier frequencies in the observations. Radiation exposure is minimized because electron radiation severely damages biological specimens during data collection. Minimal radiation, however, leads to low signal-to-noise ratios, where sensor cells record relatively low electron counts. Since imaging techniques like CT suffer from a subset of these difficulties, we believe that evaluating and analyzing our method on cryoEM problems is appropriate.

5.2 Background and related work

Modelling nano-scale structures such as proteins or viruses is a central task in structural biology. By freezing such structures and subsequently projecting them via a parallel electron beam to a sensor grid (see figure 5.2), CryoEM enables reconstruction and visualization of such structures. The technique has been described as revolutionary because researchers are capable of observing structures that cannot be crystallized, as required for X-ray crystallography [180].

The main task of reconstructing the structure from projections in cryoEM, and the wider field of medical imaging, is somewhat similar to multi-view scene reconstruction from natural images. There are, however, substantial differences. Most significantly, the projection operation in medical imaging is often an orthogonal integral projection, while in computer vision it is a non-linear perspective projection for which materials exhibit different degrees of opacity. Thus, the generative model in computer vision is more complex. Medical imaging domains, on the other hand, face significant noise and measurement uncertainties, with signal-to-noise ratios as low as 0.05 [16].

Most CryoEM techniques [39, 61, 184, 203] iteratively refine an initial structure by matching a maximum a posteriori (MAP) estimate of the pose (orientation and position) under the proposal structure with the image observation. These approaches suffer from a range of problems such as high sensitivity to poor initialization [78]. In contrast to

this approach, and closely related to our work, [22, 163] treat poses and structure as latent variables of a joint density model. MAP estimation enables efficient optimization in observation space. Previous work [190, 185, 95, 184] has suggested full marginalization, however due to its cost, it is usually intractable.

This paper extends the MAP approach by utilizing variational inference to approximate intractable integrals. Further, reparameterizing posterior distributions enables gradient based learning [106, 171]. To our knowledge this is the first such approach that provides an efficient way to learn approximate posterior distributions in this domain.

5.3 Observation Formation Model

Given a structure $\mathbf{v}$, we consider a generic generative model of observations, one that is common to many imaging modalities. As a specific example, we take the structure $\mathbf{v}$ to be a frozen (i.e. cryogenic) protein complex, although the procedure described below applies as well to CT scanning and optical microscopy. $\mathbf{v}$ is in a specific pose $\mathbf{p}_n$ relative to the direction of the electron radiation beam. This yields a pose-conditional projection, with observation $\mathbf{x}_n$. Specifically, the pose $\mathbf{p}_n = (\mathbf{r}_n, \mathbf{t}_n)$, consists of $\mathbf{r}_n \in SO(3)$, corresponding to the rotation of the object with respect to the microscope coordinate frame, and $\mathbf{t}_n \in E(3)$, the translation of the structure with respect to the origin.

The observations are then generated as follows: The specimen in pose $\mathbf{p}_n$ is subjected to radiation (the electron beam), yielding an orthographic integral projection onto the plane perpendicular to the beam. The expected projection can be formulated as

$$\bar{\mathbf{x}}_n = P\, T_{\mathbf{t}_n} R_{\mathbf{r}_n} \mathbf{v}\,. \tag{5.1}$$

Here P is the projection operator, $T_{\mathbf{t}_n}$ is the linear translation operator for translation $\mathbf{t}_n$, and $R_{\mathbf{r}_n}$ is the linear operator corresponding to rotation $\mathbf{r}_n$. Without loss of generality we can choose the projection direction to be along the z-direction $\mathbf{e}_z$. When the projection is recorded with a discrete sensor grid (i.e., sampled), information beyond the Nyquist frequency is aliased. Additionally, the recording is corrupted with noise stemming from the stochastic nature of electron detection events and sensor failures [45]. Low doses are necessary since electron exposure causes damage to sensitive biological molecules. Logically,

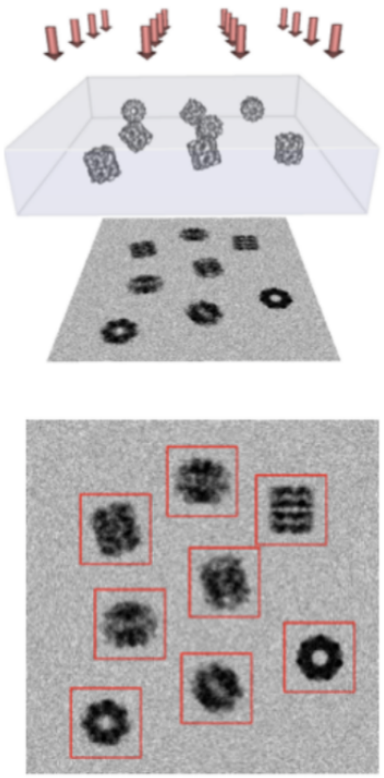

Figure 5.2: *Top:* Image formation on the example of cryo EM: The parallel Electron beam projects the electron densities on a surface where a grid of DDD sensors record the number of electrons that hit it. *Bottom:* To detect the projections (outlines in red) an algorithm seeks out areas of interest [115, 231] (Figure from [162]).

the effect is more severe for smaller objects of study.

Many sophisticated noise models have been proposed for these phenomena [47, 219, 186]. In this work, for simplicity, we assume isotropic Gaussian noise; i.e.,

$$p(\mathbf{x}_n|\mathbf{p}_n, \mathbf{v}) = \mathcal{N}(\mathbf{x}_n|\bar{\mathbf{x}}_n, 1\sigma_\epsilon^2), \tag{5.2}$$

where σ_ϵ models the magnitude of the observation noise. The image formation process is depicted in Figure 5.2.

The final section below discusses how one can generalize to more sophisticated (learnable) noise models. Note that we do not model interference patterns caused by electron scattering, called defocus and modelled with a contrast transfer function (CTF). This will lead to less realistic generative models, however we see the problem of CTF estimation as somewhat independent of our problem. Ideally, we would like to model the CTF across multiple datasets, but we leave this to future work.

5.4 Back-propagating through the generative model

In this section we aim to bridge the gap between our knowledge of the generative process $p(\mathbf{x}_n|\mathbf{p}_n, \mathbf{v})$ and a differentiable mapping that facilitates direct optimization of hidden variables $(\mathbf{p}_n, \mathbf{v})$ with gradient-descent style schemes. We start with an explanation of a naive differentiable implementation in position space, followed by a computationally more efficient version by shifting the computations to the Fourier domain (momentum space).

Naive implementation: project in position space

Our goal is to optimize the conditional log-likehood $\log p(\mathbf{x}_n|\mathbf{p}_n, \mathbf{v})$ with respect to the unobserved $\mathbf{p}_n$ and $\mathbf{v}$, maximizing the likelihood of the 2D observations. This requires (5.2) to be a differentiable operator with respect to $\mathbf{p}_n$ and $\mathbf{v}$. Note that the dependence on $\mathbf{p}_n$ and $\mathbf{v}$ is fully determined by (5.1). In order to achieve this, we first need to apply the group action $R_{\mathbf{r}_n}$ onto $\mathbf{v}$. Matrix representations of the group action such as the Euler angles matrix are defined on the sampling grid $G = \{(\nu_1^{(j)}, \nu_2^{(j)}, \nu_3^{(j)})\}_{j=1}^{D^3}$ of $\mathbf{v}$ rather than the voxel values $\{v_j\}_{j=1}^{D^3}$. For example, the action induced by a rotation around the z-axis by an angle α on the position $(\nu_1^{(j)}, \nu_2^{(j)}, \nu_3^{(j)})$ of an arbitrary voxel j can be written as,

$$R_\alpha \cdot {}^{\circ(j)} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \nu_1^{(j)} \\ \nu_2^{(j)} \\ \nu_3^{(j)} \end{pmatrix}. \tag{5.3}$$

This entails two problems. First, the volume after transformation should be sampled at the same grid points as before. This requires interpolation. Second, to achieve a differentiable map we need to formulate the transformation of position values as a transformation of the voxel values. [93] offers a solution to both problems, known as differentiable sampling[1].

The j-th voxel $v'_j = (\mathbf{v}')_j$ of the transformed volume, $\mathbf{v}' = R_{\mathbf{r}_n}\mathbf{v}$, with index vector $\zeta^{(j)}$, can be expressed as a weighted sum of all voxels before transformation $\{v_i, \nu^{(i)}\}_{i=1}^{D^3}$. The weights are determined by a sampling kernel $k(\cdot)$, the argument of which is the difference between the transformed voxel's position $\zeta^{(j)}$ and all transformed sampling grid vectors

[1] Originally invented to learn affine transformations on images to ease the classification task for standard neural networks, the approach has since been extend to 3D reconstruction problems from images [169].

$R_\alpha \cdot \nu^{(i)}$:

$$v'_j = \sum_{i=1}^{D^3} v_i \cdot k(R_\alpha^{-1} \cdot \zeta^{(j)} - \nu^{(i)}) \tag{5.4}$$

A popular kernel in this context is the linear interpolation sampling kernel[2]

$$k(R_\alpha^{-1} \cdot \zeta^{(j)} - \nu^{(i)}) = \prod_{m=1}^{3} \max(0, 1 - |(R_\alpha^{-1} \cdot \zeta^{(j)})_m - \nu_m^{(i)}|). \tag{5.5}$$

[2] Linear kernels are efficient and yield fairly good results. More complex ones such as the Lanczos re-sampling kernel may actually yield worse results due to smoothing.

Computing one voxel v'_j only requires a sum over 8 voxels from the original structure. These are determined by flooring and ceiling the elements of $(R_\alpha^{-1}\zeta^{(j)})_m$. Furthermore, the partial derivatives are provided by,

$$\frac{\partial v'_j}{\partial v_i} = k(R_\alpha^{-1} \cdot \zeta^{(j)} - \nu^{(i)}) \tag{5.6}$$

$$\frac{\partial v'_j}{\partial (R_\alpha^{-1}\zeta^{(j)})_m} = \sum_{i=1}^{D^3} v_i \prod_{l \neq k} \max(0, 1 - |(R_\alpha^{-1}\zeta^{(j)})_l - \nu_l^{(i)}|) \cdot \begin{cases} 0 & \text{if } |(R_\alpha^{-1}\zeta^{(j)})_m - \nu_m^{(i)}| \geq 1 \\ -1 & \text{elif } (R_\alpha^{-1}\zeta^{(j)})_m \geq \nu_m^{(i)} \\ 1 & \text{elif } (R_\alpha^{-1}\zeta^{(j)})_m < \nu_m^{(i)} \end{cases}. \tag{5.7}$$

This framework was originally proposed for any differentiable kernel and any differentiable affine position transformation $\nu \to \zeta$. In our setting, we restrict ourselves to linear interpolation kernels. The group actions represented by $R_\mathbf{r}$ are affine. In this work we represent rotations as Euler angles by using the Z-Y-Z convention. One could also use quaternions or exponential maps. As with rotation, the translation operation is also a transformation of the voxel grid, rather than the voxel values. Thus, (5.4) can also be used to obtain a differentiable translation operation.

Finally, the orthogonal integral projection operator is applied by summing voxel values along one principal direction. Since the position of the hidden volume is arbitrary we can fix this direction to be the Z-axis as discussed in section 5.3. Denoting a volume, rotated and translated according to $\mathbf{p} = (\mathbf{r}, \mathbf{t})$, by $\mathbf{v}' = T_\mathbf{t} R_\mathbf{r} \mathbf{v}$, the (ζ_1, ζ_2)-th element of its expected projection is given by

$$\bar{\mathbf{x}}_n[\zeta_1, \zeta_2] = (P_{3\to 2}\mathbf{v}')[\zeta_1, \zeta_2] = \sum_{\zeta_3} \mathbf{v}'[\zeta_1, \zeta_2, \zeta_3], \tag{5.8}$$

where $\mathbf{v}'[\zeta_1, \zeta_2, \zeta_3]$ denotes the element $(\zeta_1, \zeta_2, \zeta_3)$ of $\mathbf{v}'$.

This concludes a naive approach to modelling a differential map of the expected observation in position space. This approach is not particularly efficient, as according to (5.8) we need to interpolate all D^3 voxels to compute one D^2 dimensional observation. Moreover, back-propagating through this mapping requires transporting gradients through all voxels. Next, we show how to reduce the cost of this naive approach without a loss of precision by shifting the problem to the Fourier domain.

Projection-slice theorem

The projection-slice theorem or Fourier-slice theorem states the equivalence between the Fourier transform F_d of the projection $P_{d' \to d}$ of a d' dimensional function $f(r)$ onto a d-dimensional submanifold $\mathcal{F}_d P_{d' \to d} f(r)$ and a d-dimensional slice of the d'-dimensional Fourier transform of that function. This slice is a d-dimensional linear submanifold through the origin in the Fourier domain that is parallel to the projection submanifold $\mathcal{S}_d \mathcal{F}_{d'} f(r)$.

In our setting, given an axis-aligned projection direction (see (5.1)) and the discrete grid G, the expected observation in 2D Fourier space is equal to a central slice through the Fourier transformed 3D structure parallel to the projection direction $\mathbf{e}_z$:

$$\begin{aligned} \mathcal{F}_2 \bar{\mathbf{x}} =& \mathcal{F}_2 P_{3 \to 2} \mathbf{v}' \\ =& \mathcal{S}_2 (\mathcal{F}_3 \mathbf{v}') = \mathcal{S}_2 \hat{\mathbf{v}}', \end{aligned} \tag{5.9}$$

where $\mathcal{F}_2$ and $\mathcal{F}_3$ denote discrete Fourier transforms in 2 and 3 dimensions. The slice operator $\mathcal{S}$ is the Fourier equivariant of the projection operator. In our case, it is applied as follows:

$$(\mathcal{S}_2 \hat{\mathbf{v}}')[\omega_1, \omega_2] = \hat{\mathbf{v}}'[\omega_1, \omega_2, 0], \tag{5.10}$$

where ω_m are Fourier indexes.

This allows one to execute the generative model in position or momentum space. It has proven more efficient for most reconstruction algorithms to do computation in the Fourier domain [191]. This also applies to our algorithm: (i) We reconstruct the structure in the Fourier domain. This means we only need to apply an inverse Fourier transformation at the end of optimization. (ii) We may save the Fourier

transformed expected projections a-priori, further this is easily parallelized. Thus even though in its original formulation we can not expect a computational benefit, when sharing the computation across many data points to reconstruct one latent structure the gain is significant. We elaborate on this point with respect to differentiation below.

Differentiable orthographic integral projection

We incorporate the Fourier Slice Theorem into our approach to build an efficient differentiable generative model. For this we translate all elements of the model to their counterparts in the Fourier domain. The Gaussian noise model becomes [72],

$$p(\mathcal{F}_2\mathbf{x}_n|\mathbf{p}_n,\hat{\mathbf{v}}) = \mathcal{N}(\mathcal{F}_2\mathbf{x}_n|\mathcal{F}_2\bar{\mathbf{x}}_n, 1\frac{\sigma_\epsilon^2}{2}), \tag{5.11}$$

where $\hat{\mathbf{v}} = \mathcal{F}_3\mathbf{v}$. In the following, we aim to determine an expression for the expected Fourier projection $\mathcal{F}_2\bar{\mathbf{x}}_n = \mathcal{F}_2(P_{3\to2}T_{\mathbf{t}_n}R_{\mathbf{r}_n}\mathbf{v})$ by deriving the operators counterparts $\mathcal{F}_2(P_{3\to2}T_{\mathbf{t}_n}R_{\mathbf{r}_n}\mathbf{v}) = \hat{P}_{3\to2}\hat{T}_{\mathbf{t}_n}\hat{R}_{\mathbf{r}_n}\hat{\mathbf{v}}$.

We start by noting that it is useful to keep $\hat{\mathbf{v}}$ in memory $\hat{\mathbf{v}}$ to avoid computing the 3D discrete Fourier transform multiple times during optimization. The inverse Fourier transform $F_3^{-1}\hat{\mathbf{v}} = \mathbf{v}$ is then only applied once, after convergence of the algorithm. Next, we restate that the Fourier transformation is a rotation-equivariant mapping $\hat{R_{\mathbf{r}_n}} = R_{\mathbf{r}_n}$ [29]. This means the derivations from Section 5.4 with respect to the rotation apply in this context as well. A translation in the Fourier domain $\hat{T}_{\mathbf{t}_n}$, however, induces a re-weighting of the original Fourier coefficients,

$$(\hat{T}_{\mathbf{t}_n}R_{\mathbf{r}_n}\hat{\mathbf{v}})[\omega_1,\omega_2,\omega_3] = e^{-i2\pi\mathbf{t}_n\cdot\omega}(R_{\mathbf{r}_n}\hat{\mathbf{v}})[\omega_1,\omega_2,\omega_3]. \tag{5.12}$$

Finally, the last section established the equivalence of the slice and the projection operator $\hat{P}_{3\to2} = \mathcal{S}_2$ (see (5.9)) in momentum and position space. Specifically, for the linear interpolation kernel, we compute the set of interpolation points by flooring and ceiling the elements of the vector $R_{\mathbf{r}_n}^{-1}\omega^{(j)}$, $\forall\omega^{(j)} = (\omega_1^{(j)},\omega_2^{(j)},0)$. This entails 6 interpolation candidates per voxel of the central slice, in total $6D^2$. Remember, this computation above involved D^3 voxels and $6D^3$ candidates.

We can further improve the efficiency of the algorithm by swapping the projection and translation operators. That is, due to parallel radiation, and hence orthographic projection,

$$PT_{\mathbf{t}_n}R_{\mathbf{r}_n}\mathbf{v} = T_{\boldsymbol{\tau}_n}PR_{\mathbf{r}_n}\mathbf{v}, \tag{5.13}$$

where $\boldsymbol{\tau}_n = (\mathbf{t}_n \cdot \mathbf{e}_x)\mathbf{e}_x + (\mathbf{t}_n \cdot \mathbf{e}_y)\mathbf{e}_y$. This is more efficient because we reduce the translation to a two dimensional translation. Thus this modifies (5.12) to its two dimensioanl equivalent.

For the naive implementation the cost of a real space forward projection is $\mathrm{O}(D^3)$. In contrast, converting the volume to the Fourier space $\mathrm{O}(D^3 \log D)$, projecting $\mathrm{O}(D^2)$ and applying the inverse Fourier transform $\mathrm{O}(D^2 \log D)$. At first glance this implies a higher computational cost. However, for large datasets the cost of transforming the volume is amortized over all data points. For gradient descent schemes, we iterate over the dataset more than once, hence the cost of Fourier transforming the observations is further amortized. Furthermore, it is often reasonable to consider only a subset of Fourier frequencies, so back projection becomes $\mathrm{O}(r^2)$ with $r < D$. The efficiency of this algorithm in the context of cryo-EM was first recognized by Grigorieff [60]. (We provide code for the differentiable observation model and in particular for the Fourier operators: `https://github.com/KarenUllrich/pytorch-backprojection`.)

5.5 Variational inference

Here we describe the variational inference procedure for 3D reconstruction in Fourier space, enabled by the Fourier slice theorem. We assume we have a dataset of observations from a single type of protein with ground truth structure $\mathbf{v}$, and its Fourier transform $\hat{\mathbf{v}} = \mathcal{F}_3\mathbf{v}$. We consider two scenarios. In the first, both the pose of the protein and the projection are observed, and inference is only performed over the global protein structure. In the second, the pose of the protein for each observation is unknown. Therefore, inference is done over poses and the protein structure.

The first scenario is synonymous with the setting in tomography, where we observe a frozen cell or larger complex positioned in known poses. This case is often challenging because the sample cannot be observed from all viewing angles. For example, in cryo-EM tomography the specimen frozen in an ice slice can only be rotated till the verge of the slice comes into view. We find similar problems in CT scanning, for example in breast cancer scans. The second scenario is relevant for cryo-EM single particle analysis. In this case multiple identical particles are confined in a frozen sample and no information on their

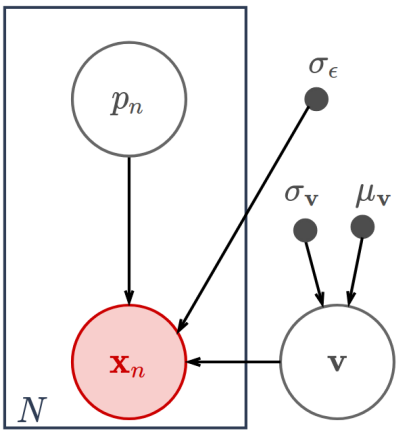

Figure 5.3: *Graphical model*: Latent structure $\mathbf{v}$, pose p_n and noise σ_ϵ can be learned from observations $\mathbf{x}_n$ through back-propagation. The latent structure distribution is thereby characterized by a set of parameters, in the Gaussian example $\mu_{\mathbf{v}}$ and $\sigma_{\mathbf{v}}$.

structure or position is available a priori.

In this work we lay the foundations for doing inference in either of the two scenarios. However, our experiments demonstrate that joint inference over the poses and the 3D structure is very sensitive to getting stuck in local minima that correspond to approximate symmetries of the 3D structure. Therefore, the main focus of this work is the setting where the poses are observed.

Inference over the 3D structure

Here the data comprise image projections and poses, $\{(\mathbf{x}_n, \mathbf{p}_n)\}_{n=1}^N$, with Fourier transformed projections denoted $\hat{\mathbf{x}}_n = \mathcal{F}_2\mathbf{x}_n$. Our goal is to learn a model $q(\hat{\mathbf{v}})$ of the latent structure that as closely as possible resembles the true posterior $p(\hat{\mathbf{v}}|\{\hat{\mathbf{x}}_n\}, \{\mathbf{p}_n\})$. For this, we assume a joint latent variable model $p(\{\hat{\mathbf{x}}_n\}_{n=1}^N, \hat{\mathbf{v}}|\{\mathbf{p}_n\}_{n=1}^N) = p(\{\hat{\mathbf{x}}_n\}_{n=1}^N|\{\mathbf{p}_n\}_{n=1}^N, \hat{\mathbf{v}})p(\hat{\mathbf{v}})$. To avoid clutter below, we use short-hand notations like $\{\hat{\mathbf{x}}_n\}$ for $\{\hat{\mathbf{x}}_n\}_{n=1}^N$.

Specifically, we minimize an upper bound to the Kullback-Leibler (KL) divergence:

$$\begin{aligned} & D_{\mathrm{KL}}\left[q(\hat{\mathbf{v}}) \| p(\hat{\mathbf{v}}|\{\hat{\mathbf{x}}_n\}, \{\mathbf{p}_n\})\right] \\ & \geq -\int \mathrm{d}\hat{\mathbf{v}}\, q(\hat{\mathbf{v}}) \ln\left(\frac{p(\{\hat{\mathbf{x}}_n\}|\{\mathbf{p}_n\}, \hat{\mathbf{v}})p(\hat{\mathbf{v}})}{q(\hat{\mathbf{v}})}\right) \\ & = \sum_{n=1}^{N} -\mathbb{E}_{q(\hat{\mathbf{v}})}\left[\ln p(\hat{\mathbf{x}}_n|\mathbf{p}_n, \hat{\mathbf{v}})\right] + D_{\mathrm{KL}}\left[q(\hat{\mathbf{v}}) \| p(\hat{\mathbf{v}})\right]. \end{aligned} \tag{5.14}$$

Here, we have assumed that, given the volume, the data are IID: $p(\{\hat{\mathbf{x}}_n\}|\{\mathbf{p}_n\}, \hat{\mathbf{v}}) = \prod_{n=1}^N p(\hat{\mathbf{x}}_n|\mathbf{p}_n, \hat{\mathbf{v}})$. We have bounded the divergence by the data log-likelihood $\ln p(\{\hat{\mathbf{x}}_n\})$, a constant with respect to $\hat{\mathbf{v}}$ and $\{\mathbf{p}_n\}$. This is equivalent to lower bounding the model evidence by

introducing a variational posterior [98].

In this work we focus on modelling $q(\hat{\mathbf{v}})$ as isotropic Gaussian distribution. The prior is assumed to be a standard Gaussian. In practice, we use stochastic gradient descent-like optimization, with the data organized in mini-batches. That is, we learn the distribution parameters $\eta = \{\mu_{\mathbf{v}}, \sigma_{\mathbf{v}}\}$ for $q(\hat{\mathbf{v}}) = q_\eta(\hat{\mathbf{v}})$ through stochastic optimization, efficiently by using the reparameterization trick [106, 171].

In (5.14), the reconstruction term depends on the number of datapoints. The KL-divergence between the prior and approximate posterior does not. As the sum of the mini-batch objectives should be equal to the objective of the entire dataset, the mini-batch objective is

$$\sum_{n \in D_i} -\mathbb{E}_{q(\hat{\mathbf{v}})} \left[\ln p(\hat{\mathbf{x}}_n | \mathbf{p}_n, \hat{\mathbf{v}}) \right] + \frac{|D_i|}{N} D_{\mathrm{KL}} \left[q(\hat{\mathbf{v}}) \| p(\hat{\mathbf{v}}) \right]. \tag{5.15}$$

where D_i is the set of indices of the data in minibatch i, and $|D_i|$ denotes the size of the i-th minibatch.

Joint inference over the 3D structure and poses

In the second scenario the pose of the 3D structure for each observation is unknown. The data thus comprises the observed projections $\{\mathbf{x}_n\}_{n=1}^N$. Again, we perform inference in the Fourier domain, with transformed projections $\{\hat{\mathbf{x}}_n\}_{n=1}^N$ as data. We perform joint inference over the poses $\{\mathbf{p}_n\}_{n=1}^N$ and the volume $\hat{\mathbf{v}}$. We assume the latent variable model can be factored as follows, $p(\{\hat{\mathbf{x}}_n\}, \hat{\mathbf{v}}, \{\mathbf{p}_n\}) = p(\{\hat{\mathbf{x}}_n\} | \{\mathbf{p}_n\}, \hat{\mathbf{v}}) p(\{\mathbf{p}_n\}) p(\hat{\mathbf{v}})$.

Upper bounding the KL-divergence, as above, we obtain

$$\begin{aligned}
&D_{\mathrm{KL}} \left[q(\hat{\mathbf{v}}) q(\{\hat{\mathbf{x}}_n\}) \| p(\hat{\mathbf{v}}, \{\hat{\mathbf{x}}_n\}, \{\mathbf{p}_n\}) \right] \\
&\geq - \int \int \mathrm{d}^N \mathbf{p}_n \mathrm{d}\hat{\mathbf{v}} \; q(\{\mathbf{p}_n\}) q(\hat{\mathbf{v}}) \\
&\qquad\qquad \times \ln \left(\frac{p(\{\hat{\mathbf{x}}_n\} | \{\mathbf{p}_n\}, \hat{\mathbf{v}}) p(\{\mathbf{p}_n\}) p(\hat{\mathbf{v}})}{q(\{\mathbf{p}_n\}) q(\hat{\mathbf{v}})} \right) \\
&= \sum_{n=1}^N -\mathbb{E}_{q(\hat{\mathbf{v}})} \mathbb{E}_{q(\mathbf{p}_n)} \left[\ln p(\hat{\mathbf{x}}_n | \mathbf{p}_n, \hat{\mathbf{v}}) \right] \\
&\quad + \sum_{n=1}^N D_{\mathrm{KL}} \left[q(\mathbf{p}_n) \| p(\mathbf{p}_n) \right] + D_{\mathrm{KL}} \left[q(\hat{\mathbf{v}}) \| p(\hat{\mathbf{v}}) \right].
\end{aligned} \tag{5.16}$$

The prior, approximate posterior and conditional likelihood all factorize across datapoints: $p(\{\mathbf{p}_n\}) = \prod_{n=1}^N p(\mathbf{p}_n)$, $q(\{\mathbf{p}_n\}) = \prod_{n=1}^N q(\mathbf{p}_n)$,

and $p(\{\hat{\mathbf{x}}_n\}|\{\mathbf{p}_n\}, \hat{\mathbf{v}}) = \prod_{n=1}^{N} p(\hat{\mathbf{x}}_n|\mathbf{p}_n, \hat{\mathbf{v}})$. Like (5.15), for mini-batch optimization algorithms we make use of the objective

$$\sum_{n \in D_i} -\mathbb{E}_{q(\hat{\mathbf{v}})}\mathbb{E}_{q(\mathbf{p}_n)} \left[\ln p(\hat{\mathbf{x}}_n|\mathbf{p}_n, \hat{\mathbf{v}})\right] \tag{5.17}$$
$$+ \sum_{n \in D_i} D_{\mathrm{KL}} \left[q(\mathbf{p}_n) \| p(\mathbf{p}_n)\right] + \frac{|D_i|}{N} D_{\mathrm{KL}} \left[q(\hat{\mathbf{v}}) \| p(\hat{\mathbf{v}})\right].$$

In Section 5.5, we learn the structure parameters shared across all data points. Here the pose parameters are unique per observation and therefore require separate inference procedures per data point. As an alternative, we can also learn a function that approximates the inference procedure; This is called amortized inference [106]. In practice, a complex parameterized function f_ϕ such as a neural network predicts the parameters of the variational posterior $\eta = f_\phi(\cdot)$.

5.6 Experiments

We empirically test the formulation above with simulated data from the well-known GroEL-GroES protein [223]. To this end we generate three datasets, each with 40K projections onto 128×128 images at a resolution of 2.8Å per pixel. The three datasets have signal-to-noise ratios (SNR) of 1.0, 0.04 and 0.01, referred to below as the *noise-free*, *medium-noise* and *high-noise* cases. Figure 9.16 shows one sample per noise level. As previously stated in Section 5.3, we do not model the microscope's defocus or electron scattering effects, as captured by the CTF [109].

Using synthetic data allows us to evaluate the algorithm with the ground-truth structure, e.g., in terms of mean-squared error (MSE). With real-data, where ground truth is unknown, the resolution of a fitted 3D structure is often quantified using *Fourier Shell Correlation* [179]: The N observations are partitioned randomly into two sets, A and B, each of which is then independently modeled with the same reconstruction algorithm. The normalized cross-correlation coefficient is then computed as a function of frequency $f = 1/\lambda$ to assess the agreement between the two reconstructions.

Given two 3D structures, F_A and F_B, in the Fourier domain, FSC at

frequency f is given by

$$FSC(f|F_A, F_B) = \frac{\sum_{f_i \in S_f} F_A(f_i) \cdot F_B(f_i)^*}{\sqrt[2]{\sum_{f_i \in S_f} |F_A(f_i)|^2 \cdot \sum_{f_i \in S_f} |F_B(f_i)|^2}} . \tag{5.18}$$

where S_f denotes the set of frequencies in a shell at distance f from the origin of the Fourier domain (i.e. with wavelength λ). This yields FSC curves like those in Fig. 5.4. The quality (resolution) of the fit can be measured in terms of the frequency at which this curve crosses a threshold τ. When one of F_A or F_B is ground truth, then $\tau = 0.5$, and when both are noisy reconstructions it is common to use $\tau = 0.143$ [179]). The structure is then said to be resolved to wavelength $\lambda = 1/f$ for which $FSC(f) = \tau$.

Comparison to Baseline algorithms

When poses are known, the current state-of-the-art (SOTA) is a conventional tomographic reconstruction (a.k.a. back-projection). When poses are unknown, there are several well-known SOTA cryo-EM algorithms [39, 61, 184, 203]. All provide point estimates of the 3D structure. In terms of graphical models, point estimates correspond to the case in which the posterior $q(\hat{\mathbf{v}})$ is modeled as a delta function, $\delta(\hat{\mathbf{v}}|\mu_{\mathbf{v}})$, the parameter of which is the 3D voxel array, $\mu_{\mathbf{v}}$.

We compare this baseline to a model in which the posterior $q(\hat{\mathbf{v}})$ is a multivariate diagonal Gaussian, $\mathcal{N}(\hat{\mathbf{v}}|\mu_{\mathbf{v}}, 1\sigma^2_{\mathbf{v}})$. While the latent structure is modeled in Fourier domain, the spatial domain signal is real-valued. We restrict the learnable parameters, $\mu_{\mathbf{v}}$ and $\sigma_{\mathbf{v}}$, accordingly [3]. We use the reparameterization trick thus correlate the samples accordingly. Finally, the prior in (5.14) in a multivariate standard Gaussian, $p(\mathbf{v}) = \mathcal{N}(\mathbf{v}|\mathbf{0}, 1)$.

[3] That is for $\mu_{\mathbf{v}}$, $\Re(\mu_{\mathbf{v}})[\zeta] = \Re(\mu_{\mathbf{v}})[-\zeta]$ and $\Im(\mu_{\mathbf{v}})[\zeta] = -\Im(\mu_{\mathbf{v}})[-\zeta]$ with $\zeta = (\zeta_1, \zeta_2, \zeta_3)$. Further, a noise sample is shared accordingly.

Table 5.1 shows results with these two models, with known poses (the tomographic setting), and with *noise-free* observations. Given the sensor resolution of $r = 2.8$, the highest possible resolution would be the Nyquist wavelength of 5.6. Our results show that both models approach this resolution, and in reasonable time.

Posterior	$\delta(\mathbf{v}\|\mu_\mathbf{v})$	$\mathcal{N}(\mathbf{v}\|\mu_\mathbf{v}, 1\sigma^2_\mathbf{v})$
Time until converged [s]	$\sim$ 390	$\sim$ 480
MSE [10^{-3}/voxel]	2.29	2.62
Resolution [Å]	5.82	5.82

Table 5.1: Results for modelling protein structure as latent variable. Fitting a Gaussian or Dirac posterior distribution with VI leads to similar model fits, as measured by MSE and FSC between fit and ground truth with $\tau = 0.5$.

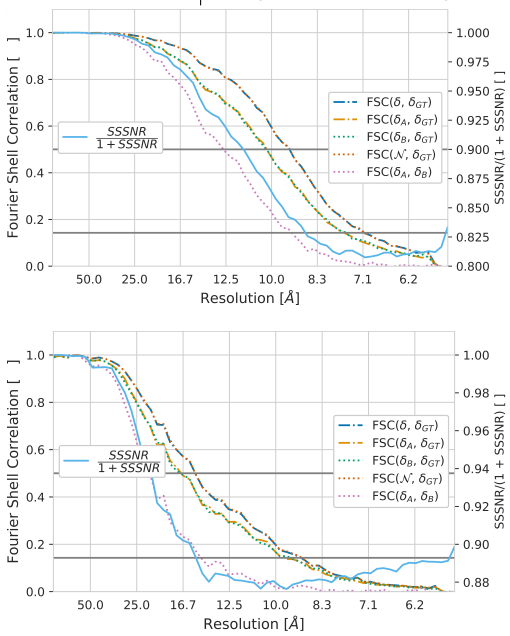

Figure 5.4: The FSC curves for various model fits. The grey lines indicate resolution thresholds. The higher the resolution of a structure the better the model fit. We contrast the FSC curves with the proposal we make to evaluate model fit.

Uncertainty estimation leads to data efficiency

In this section, we explore how modelling the latent structure with uncertainty can improve data efficiency. For this, recall that FSC is computed by comparing reconstructions based on dataset splits, A and B. As an alternative, we propose to utilize the learned model uncertainties $\sigma_\mathbf{v}$ to achieve a similar result. We thus only need one model fit that includes both dataset splits. Specifically, we propose to extend a measure first presented in Unser et al. [210]: the spectral SNR to the 3D case, and hence refer to it as spectral shell SNR (SS-SNR). When modelling the latent structure as diagonal Gaussian $\mathcal{N}(\mathbf{v}|\mu_\mathbf{v}, 1\sigma^2_\mathbf{v})$, the SS-SNR can be computed to be

$$\alpha(f) = \frac{\sum_{f_i \in S_f} |\mu_\mathbf{v}(f_i)|^2}{\sum_{f_i \in S_f} \sigma^2_\mathbf{v}(f_i)}. \tag{5.19}$$

Following the formulation by [210], we can then express the FSC in terms of the SS-SNR, i.e., $FSC \approx \alpha/(1+\alpha)$.

Figure 5.4 shows FSC curves based on reconstructions from the *medium-noise* (top) and *high-noise* (bottom) datasets. First, we aim to demonstrate that the FSC curves between the Gaussian fit (with all data) vs ground truth, and the MAP estimate model with all data vs ground truth, i.e., FSC($f|\delta$, δ_{GT}) and FSC($f|\mathcal{N}$, δ_{GT}), yield the same fit quality. Note that we would not usually have access to the ground truth structure. Secondly, because in a realistic scenario we would not have access to the ground truth we would need to split the dataset in two. For this we evaluate the FSC between ground truth and two disjoint splits of the dataset FSC($f|\delta_A$, δ_{GT}) and FSC($f|\delta_B$, δ_{GT}). This curve not surprisingly lies under the previous curves. Also note, that the actual measure we would consider FSC($f|\delta_A$, δ_B) is more conservative. Finally, we show that $\alpha/(1+\alpha)$ curve has the same inflection points as the FSC curve. As one would expect, it lies above the conservative FSC($f|\delta_A$, δ_B).

Using α one can quantify the model fit with learned uncertainty rather than FSC curve. As a consequence there is no need to partition the data and perform two separate reconstructions, each with only half the data.

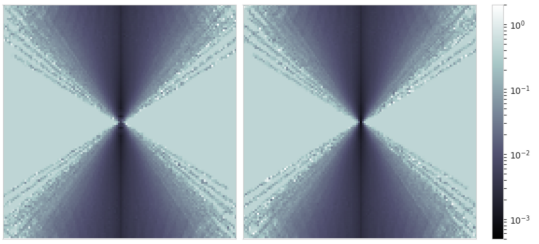

Figure 5.5: Center slice through the learned Fourier volume uncertainties $\sigma_{\mathbf{v}}$. *Left:* real part, *Right:* imaginary part. We learn the model fit with observations coming only from a 30° cone, a scenario similar to breast cancer scans where observations are available only from some viewing directions. Uncertainty close to 1 means that the model has no information in these areas, close to zero represents areas of high sampling density. In contrast to other models, our model can identify precisely where information is missing (high variance).

Uncertainty identifies missing information

Above we discussed how global uncertainty estimation can help estimate the quality of fit. Here we demonstrate how local uncertainty can help evaluate the local quality of fit[4]. In many medical settings, such as breast cancer scans, limited viewing directions are available. This is an issue in tomography, and also occurs in single particle cryo-EM when the distribution of particle orientations around the viewing sphere are highly anisotropic. To recreate the same effect we construct a dataset of 40K observations, as before with no noise to separate the sources of information corruption[5]. We restrict viewing directions to Euler angles (α, β, γ) with $\beta \in (-15, +15)$; i.e., no observations outside a 30° cone As above, we assume a Gaussian posterior over the latent protein structure.

[4] Other studies recognize the importance of local uncertainty estimation, measuring FSC locally in a wavelet basis [24, 113, 215].

[5] For completeness we present the same experiment with noise in the appendix.

Figure 5.5 shows the result. We can show that the uncertainty the model has learned correlates with regions in which data has been observed (uncertainty close to 0) and has not been observed (uncertainty close to 1). Due to the pressure from the KL-divergence (see (5.14)) the latter areas of the model default to the prior $\mathcal{N}(\mathbf{v}|\mathbf{0}, 1)$. This approach can be a helpful method to identify areas of low local quality of fit.

Limits: Treating poses as random variables

Extending our method from treating structures as latent variables to treating poses as latent variables is difficult. In the following we analyze why this is the case. Note though that, our method of estimating latent structure can be combined with common methods of pose estimation such as branch and bound [163] without losing any of the benefits we offer. However, ideally it would be interesting to also learn latent pose posteriors. This would be useful to for example detect outliers in the dataset which is common in real world scenarios.

In an initial experiment, we fix the volume to the ground truth. We subsequently only estimate the poses with a simple Dirac model for each data point $\mathbf{p}_n \sim \delta(\mathbf{p}_n|\mu_{\mathbf{p}_n})$. In figure 5.6, we demonstrate the problem of SGD-based learning for this. For one example (samples on top), we show its true error surface, the global optimum (red star) and the changes over the course of optimization (red line). We observe that due to the high symmetries in the structure, the true pose posterior error surface has many symmetries as well. An estimate depending on its starting position, seems to converge to the closest local optimum only rather than the global one. We would hope to be able to fix this problem in the future by applying more advanced density estimation approaches.

5.7 Model Criticism and Future Work

This paper introduces practical probabilistic models into the scientific imaging pipeline, where practical refers to scalability through the use of the reparameterization trick. We show how to turn the operators in the pipeline into differentiable maps, as this is required to apply the trick. The main focus of the experiments is to show why this

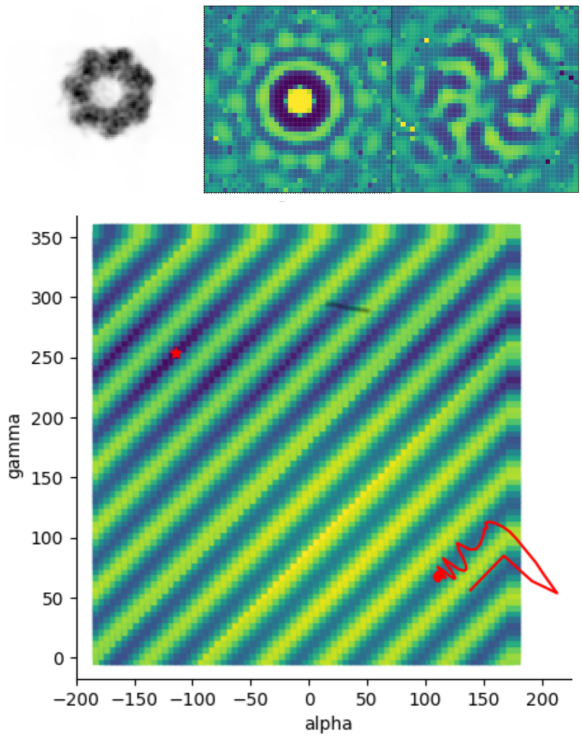

Figure 5.6: Example of gradient descent failing to estimate the latent pose of a protein. *Small images from left to right:* observation, real and imaginary part of the Fourier transpose. *Large figure:* Corresponding error surface of the poses for 2 of 3 Euler angles. The red curve shows the progression of the pose estimate over the course of optimization. It is clear that the optimization fails to recover the true global optimum (red star).

novelty is important, addressing issues such as data efficiency, local uncertainty, and cross validation. Specifically, we found that a parameterized distribution, i.e. the Gaussian, achieves the same quality of fit as a point estimate, i.e. the dirac, while relying on less data. We conclude that our latent variable model is a suitable building block. It can be plugged into many SOTA approaches seamlessly, such as [39, 61, 184, 203]. We also established that the learned uncertainty is predictive of locations with too few samples. Finally, we demonstrated the limits of our current methods in treating poses as latent variables. This problem, however, does not limit the applicability of our method to latent structures. We thus propose to combine common pose estimation with our latent variable structure estimation. This method benefit from the uncertainty measure but also find globally optimal poses.

In future work we hope to find a way to efficiently learn pose posterior distributions as well. We hope that a reasonable approach would be to use multi-modal distributions and thus more advanced density estimation techniques. We will also try to incorporate amortized inference, mentioned in Section 5.5. Amortization would give the additional advantage of being able to transfer knowledge from protein to protein. Transfer could then lead to more advanced noise and CTF models. Bias in transfer will be a key focus of this effort; i.e., we only want to transfer features of the noise and not the latent structure. An-

other problem we see with the field of reconstruction algorithms is that the model evaluation can only help to detect variance but not bias in a model class. This is a problem with FSC comparison, but also with our proposal. We believe that an estimate of the data-log-likelihood of a hold out test dataset is generally much better suited. In a probabilistic view, this can be achieved by importance weighting the ELBO [172].

6

Neural Communication Systems with Bandwidth-limited Channel

Reliably transmitting messages despite information loss due to a noisy channel is a core problem of information theory. One of the most important aspects of real world communication, e.g. via wifi, is that it may happen at varying levels of information transfer. The bandwidth-limited channel models this phenomenon. In this study we consider learning coding with the bandwidth-limited channel (BWLC). Recently, neural communication models such as variational auto-encoders have been studied for the task of source compression. We build upon this work by studying neural communication systems with the BWLC. Specifically, we find three modelling choices that are relevant under expected information loss. First, instead of separating the sub-tasks of compression (source coding) and error correction (channel coding), we propose to model both jointly. Framing the problem as a variational learning problem, we conclude that joint systems outperform their separate counterparts when coding is performed by flexible learnable function approximators such as neural networks. To facilitate learning, we introduce a differentiable and computationally efficient version of the bandwidth-limited channel. Second, we propose a design to model missing information with a prior, and incorporate this into the channel model. Finally, sampling from the joint model is improved by introducing auxiliary latent variables in the decoder. Experimental results

justify the validity of our design decisions through improved distortion and FID scores.

6.1 Introduction

The 21st century is often referred to as the information age. Information is being created, stored and sent at rates never before seen. To cope with this deluge of information, it is vital to design optimal communication systems. Such systems solve the problem of reliably transmitting information from sender to receiver given some form of information loss due to transmission errors (i.e. through a noisy channel). As the size of the transmitted messages goes to infinity for memory-less communication channels, the joint source-channel coding theorem [187] states that it is optimal to split the communication task into two sub-tasks: (i) removing redundant information from the message (source coding) and (ii) re-introducing some redundancy into the encoded message to allow for message reconstruction despite the channel information loss (channel coding). As a result, separate systems have been studied extensively in the literature and are the standard way of coding for many scenarios. However, it is also well known that there are limits to the optimality of separate systems in practical settings. Most importantly for this work, limitations arise when we seek to encode finite length messages as is the case with any real-world application [111]. These limits result in two important consequences: (i) When there is a budget on transmission bits, a challenging unequal allocation of resources must be made between the source and channel to optimize reconstruction results. (ii) Decoding via the maximum-likelihood principle becomes an NP-hard problem [18]. Thus approximations need to be made that can lead to highly sub-optimal solutions [108, 49, 218].

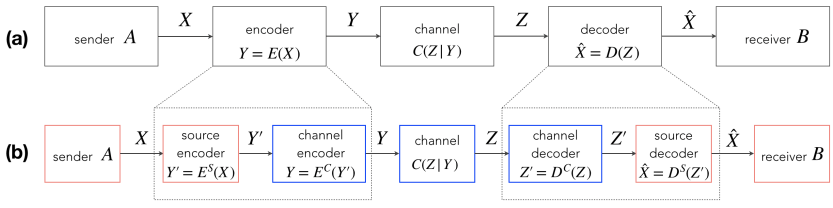

Figure 6.1: **(a)** Joint communication system: A message X passes through a joint source and channel encoder before it passes a channel and is subsequently decoded. **(b)** Separate communication system: This system distinguishes source encoding and decoding (red) from channel encoding and decoding (blue). The red and blue systems are designed independently of each other.

Recent work [30, 46], has thus looked at the problem of learning to jointly communicate. This includes systems that learn to do source

and channel coding jointly from data. Practically this can be achieved by learning neural network encoders and decoders, where channels are simulated by adding noise to encoded messages. Several desirable properties of such systems were shown, including improvements in decoding speed and code length. Complementary to this body of work, we focus this study on the investigation of neural joint models with the bandwidth-limited channel. Specifically, we direct our experimentation on the bandwidth-limited channel due to it's ubiquity as a fundamental component in the real world communication systems. The main contributions of this work include:

1. We cast the problem of learning joint communication as a variational learning problem, parallel to other work [30].
2. We justify the importance of jointly learned systems by empirically evaluating the gap between neural systems for joint and separate communication.
3. We design standard channels such as the Gaussian and Binary channel as differentiable probabilistic nodes, which serve as base for our design of the bandwidth-limited channel.
4. We show how transmission rate can be improved through learned prior models.
5. We introduce an auxiliary latent variable decoder and show how it can improve image reconstructions in the low bandwidth regime.

6.2 Notation and preliminaries

We mark sets as calligraphic letters (i.e. $\mathcal{X}$), random variables as capital letters (i.e. X) and their values as lower case letters (i.e. x). We use capital letters to denote probability distributions (i.e. $P(X)$) and lower case letters for the corresponding densities (i.e. $p(x)$). We will refer to the entropy of stochastic processes. This describes the average rate at which a process emits information. Formally,

$$H(X) = \mathbb{E}_{P(X)}[-\log P(X)], \tag{6.1}$$

where $\mathbb{E}$ is the expectation. Further, we expect the reader to be familiar with the distortion-rate theory. Appendix 9.4 summarizes these shortly and makes connections to neural compression systems.

6.3 Source and Channel Coding for Communication Systems

In this section the reader will be introduced to communication systems and in particular to the challenge of joint coding when a finite bit-length budget is given.

Communication is defined by an entity A called the sender, or source, that induces a state X, the message, in another entity B, the receiver. We call this transfer of information successful if A and B agree about the message being sent: $X = \hat{X}$, or if the message distortion $||X - \hat{X}||$ does not exceed a certain level D. Real-world communication is an inherently noisy physical process where many uncontrollable or unpredictable factors may interfere with a sent message before it reaches its receiver. To account for this interference, communication is typically organized into three distinct components which we illustrate in Figure 6.1: (i) The *encoder* $Y = E(X)$, whose role is to compress its inputs (i.e. to remove redundant information) and subsequently prepare them for transmission through the channel with minimal distortion. (ii) The *channel* $Z = C(Z|Y)$ over which we have no control, and represents the unpredictable distortions caused by the physical transmission process. (iii) The *decoder* $\hat{X} = D(Z)$, whose goal is to reverse the process to the original datum X from the received code Z.

Channel capacity The most important characteristic of a channel is its capacity. In order to evaluate it, we may compute the number of distinguishable messages that we can send through the channel given an encoder. The logarithm of that number is referred to as *information capacity of the channel*. It is given by the maximum of the mutual information of Y and Z, $I(Y;Z)$, taken over all possible input distributions $P(Y)$,

$$C = \max_{P(Y)} I(Y;Z). \tag{6.2}$$

Other relevant properties of channel models include (i) *bandwidth*, the number of information units passing a channel per time unit, (ii) *memory*, the independence of joint probabilities of a transmitted sequence, where a channel with fully independent joint probabilities is called *memoryless* , and (iii) *feedback*, the ability for the sender side of the system to know what bits have arrived at the receiver side, resulting in $Y = E(X, Z)$. Note that in this work, we will constrain our research question to feedback and memoryless channels.

Joint source-channel coding The channel capacity fundamentally restricts the ability of a communication system to transfer messages. The source-channel coding theorem (SCCT) specifies this restriction as follows. For i.i.d. variables and memoryless channels, given a certain tolerated message distortion D, we must send codes with length $R(D)$. The data may be recovered by the receiver at distortion D if and only if $R(D) < C$.

Furthermore, it can be shown that there exists a two stage method that is as good as any alternative to transmit information over a noisy channel reliably: source coding and channel coding. These two steps can be accomplished by two distinctly designed systems, referred to as the source encoder and decoder, $E^S(\cdot)$ and $D^S(\cdot)$, and the channel encoder and decoder, $E^C(\cdot)$ and $D^C(\cdot)$, respectively. It is easy to see why this result had great impact on the design of communication systems in practice. A communication problem is essentially defined by its source and its channel. Any such tuple defines an individual problem, resulting in an enormous problem space. By separation, it is possible to independently reuse good source or channel solutions for other problems.

However, there are also restrictions to the applicability of the SCCT. For finite length messages, we have to trade bits for compression and channel coding against each other. This is not trivial [161, 37, 111]. On top of this, encoders and decoders are being idealized to be any function. In practical settings however, we may not be able to identify optimal encoders. Further, they are computationally restricted. In the era of machine learning, however, hypothesis spaces can be searched increasingly quickly in an automated fashion, allowing researchers to search over the space of joint solutions for the first time. For these reasons, we propose to learn joint communication systems using flexible function approximators such as deep neural networks.

6.4 Learned Communication Systems

In this section, we outline how to learn neural encoders and decoders for a given joint communication problem defined by a channel and a source. Our approach requires a differentiable path through the communication system. For this, we design appropriate channel models. Additionally we introduce a new design for the bandwidth-limited

channel, adapted from classical models, and explain how to do marginalization of bands in practice. Consequently, we frame learning in the joint and in the separate model as a variational optimization problem. Our approach is related to auto-encoders [217] and variational auto-encoders [171, 106]. We will outline the connection here. Finally, we introduce auxiliary latent variables (ALV) to the decoding process as means to combat low reconstruction quality when the information transmitted though such a model is limited.

Channel Models

To enable back-propagation through a communication system, we shall introduce the most common channel models in the literature and explain how to build them in a differentiable fashion.

Gaussian Channel We start this discussion with the Gaussian channel model, the most important continuous alphabet channel. It is a time discrete channel that distorts incoming signal Y by i.i.d. Gaussian noise W.

$$Z_i = Y_i + W_i, \qquad W_i \sim \mathcal{N}(0, \sigma^2) \tag{6.3}$$

However, this particular definition is of limited use. When the noise is fixed but the power of the input is not, one can easily design channel encodings that essentially ignore that noise. It is thus common to power constrain the input, or equivalently keep a constant *signal-to-noise ratio* (SNR), s. It can be shown that the channel capacity of a power limited Gaussian channel is equal to $C = \frac{1}{2}\log(1+s)$ bits per transmission [35]. For a differentiable Gaussian channel with constant SNR, we assume the channel input to be an isotropic Gaussian distribution $Y_i \sim \mathcal{N}(\mu_{Y_i}, \sigma^2_{Y_i})$. We propose to use the reparameterization trick [106, 171], where a probabilistic node is separated into a parameter independent stochastic node and a deterministic one. By using the trick twice we can rewrite the channel to

$$Z_i = Y_i + \frac{\mu_{Y_i}}{s} \cdot W_i, \qquad W_i \sim \mathcal{N}(0, 1). \tag{6.4}$$

Bandwidth-limited channel Related to the Gaussian channel, and one of the most important models for communication, e.g. over a radio and wifi, is the bandwidth-limited channel. The channel capacity for a Gaussian bandwidth limited channel is known to be linearly related to the bandwidth $C \sim B$. In the classical literature, this is described as a continuous time, white noise and bandwidth-pass filtered channel

[1]; however, in this work, we adopt the concept to be a discrete time channel, for which we introduce the bandwidth B as a discrete latent variable,

[1] see [35], chap. 10 for more details

$$
\begin{aligned}
C(Z|Y) &= \sum_B C(Z, B|Y) \\
&= \sum_B P(B) \underbrace{\prod_{t=1}^{B} C(Z_t|Y_t, \{Z_\tau\}_{\tau=1}^{t-1}) \prod_{t=B}^{T} P_{Y_t}(Z_t)}_{=C(Z|B,Y)}.
\end{aligned} \tag{6.5}
$$

where t is a discrete time step. In words, a signal X gets encoded into a sequence $Y = \{Y_t\}_{t=1}^{T}$. The sequence gets transmitted up to B by sending $Y = \{Y_t\}_{t=1}^{B}$ though a channel $C(Z_t|Y_t)$ such as the Gaussian channel. Other information $Y = \{Y_t\}_{t=B+1}^{T}$ is lost. This information is replaced by samples from a prior over Y_t, $P_{Y_t}(Z_t)$. The full integration over the input domain required to compute the integral $P_{Y_t}(Y_t) = \int e(Y_t|x_t)\, dx$ is expensive. Thus, we will introduce an approximation to it, i.e., a standard Gaussian prior or a more elaborate model such as the ConvDraw prior [58].

To summarize, we have introduced a differentiable and computationally efficient version of the bandwidth-limited channel. For this, we turn it into a time discrete channel by introducing the discrete latent variable B. To marginalize over the latent variable we may either perform Monte Carlo sampling or complete marginalization. The model also requires a model for codes that have been dropped. This is similar to the prior in a variational auto-encoder and can be learned to arbitrary complexity.

Other differentiable models include the erasure channel, first considered in [100]. However, this channel is mainly relevant for feedback systems, we will thus not discuss it in this context. Another relevant channel is the Binary channel, which we detail in the appendix. For real-world channels there is the option to learn a parametric model that emulates them by sending random information units. Subsequently this learned parametric model can be utilized as channel model. If only a black-box model of the channel is available, our proposed framework may be extended by using discrete optimization schemes. For example VIMCO [144] has been used in [30]. The implementation of the channels we consider here can be found online, `github.com/anonymous_code`.

Separate Source-Channel coding

As described in section 6.3, the joint communication problem can be broken down into two independent problems; the source coding and channel coding problem. Here, we demonstrate how to apply the variational auto-encoder as a source coder and an auto-encoder as channel coder. Note that, there is no exchange of information between those two systems. We provide a visual aid for this section in Figure 6.2.

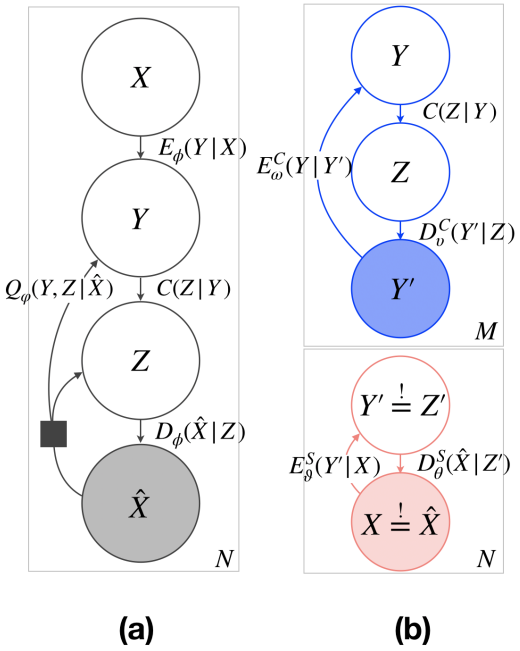

Figure 6.2: **(a)** *Graphical model of the jointly learned communication system.* The message X is passed by the encoder and the channel, to be reconstructed into $\hat{X}$. Note that because marginalization is not possible we apply a variational approximation Q to aid inference. **(b)** *Graphical models of the separately learned communication system.* Two systems are learned independently: a VAE for source compression (red) and an AE for channel transmission (blue).

Source-VAE In recent years, neural networks have been shown to be useful source compressors. Specifically, variational auto-encoders (VAEs) have been pointed to as natural source coding systems [106, 5], showing great practical success [13, 14, 142, 232, 208]. Such a source-VAE is essentially a learned probabilistic model. Based on a set of samples emitted by the source $\mathbf{X} = \{x_n\}_{n=1}^N$, we aim to learn the source encoder $E^S_\vartheta(Y'|X)$ and the source decoder $D^S_\theta(\hat{X}|Z')$, both parameterized functions of ϑ and θ, respectively. The learning objective thereby originates from looking at the model as a latent-variable model (with the encoding Z being the latent variable) for which we aim to do maximum marginal likelihood learning of the parameters. The involved marginalization, however, forces the introduction of a variational approximation, the encoder, to construct a lower bound on the marginal log likelihood, known as variational inference. For this, we set the source encoder to be the variational approximation to the source de-

coder, such that $Y' \stackrel{!}{=} Z'$ and $X \stackrel{!}{=} \hat{X}$:

$$\begin{aligned}\mathbb{E}_{P(X)}\left[\log P(X|\theta,\vartheta)\right] \geq \mathbb{E}_{P(X)}\Bigg[\underbrace{\mathbb{E}_{E_\vartheta^S(Y'|X)}[\log D_\theta^S(\hat{X}|Z')]}_{=:-D}\Bigg] \\ - \mathbb{E}_{P(X)}\Bigg[\underbrace{KL(E_\vartheta^S(Y'|X)||P_\theta(Z))}_{=:-R}\Bigg]\end{aligned} \tag{6.6}$$

This bound is known as the evidence lower bound (ELBO). Optimizing ELBO is equivalent to optimizing a rate(R)-distortion(D) problem. We can adjust the rate-distortion trade-off to a desired rate or distortion by introducing a parameter β into the objective, this framework is well known as β-VAE [79, 5].

Generally, it is possible to optimize decoder and encoder independently. This however would only make sense if we consider channel coding systems that do not try to reconstruct their inputs. Note that in contrast to the original formulation in section 6.3, encoder and decoder have been turned into probabilistic mappings rather than deterministic ones. This allows one to find an ideal compression rate given a certain distortion-rate trade-off β. The rate can practically be achieved with the so called bits-back coding [80, 207]. For inference it became common that the parameters for the encoder distribution may be predicted by a neural network parameterized by ϑ. This is called amortized inference. The parameters of this inference model and the generative model, the decoder, are trained jointly though stochastic maximization of ELBO. To do this efficiently, it is common to use the reparameterization trick [106, 171].

Finally, it is important to note that the prior distributions in the context of compression may not be learned $P_\theta(Z) = P(Z)$. This would conflict the independence of the source and channel.

Channel-AE For training a neural channel coding system, we will use samples from the source independent prior $\{y'_m\}_{m=1}^M$, $y'_m \sim P(Z)$. After using a deterministic encoding $Y = E_\omega^C(Y|Y')$, we send Y though the probabilistic channel $Z \sim C(Z|Y)$, after which we try to recover the inputs by channel decoding $Z' = D_\upsilon^C(Z'|Z)$. The system is trained by minimizing a measure of distortion between Y' and Z'.

Note, that a for a simple additive withe Gaussian noise channel there exists a near optimal channel coding scheme: LDPC. However, in more general scenarios they do not perform as well anymore and can be

beaten by neural network architectures [100]. Further, it has been shown that neural networks can decode them efficiently [147]. For the sake of generalizing to more complex channels we thus propose general purpose neural network channel coding.

Joint Source-Channel coding

For the jointly optimized system, we translate the communication system as described in section 6.3 into a generative model $P_\phi(\hat{X}|X) = \int e_\phi(y|X)c(z|y)d_\phi(\hat{X}|x)\,dy\,dz$. Similar to the previous section, we think of the encoder and decoder as parameterized mappings, while the channel model is taken as given. We are interested in performing maximum likelihood learning of the model parameters ϕ, by optimizing

$$\begin{aligned} &\mathbb{E}_{P(X)}\left[\log P_\phi(\hat{X}|X)\right] \\ &= \mathbb{E}_{P(X)}\left[\log \int e_\phi(y|X)c(z|y)d_\phi(\hat{X}|x)\,dy\,dz\right] \end{aligned} \tag{6.7}$$

The required marginalization in (6.7), however, leads to generally intractable integrals. One frequently applied solution is to introduce a variational approximation $Q_\varphi(Y,Z|\hat{X})$ to the posterior, to construct a lower bound on the marginal likelihood.

$$\begin{aligned} &\log P_\phi(\hat{X}|X) \geq \mathbb{E}_{Q_\varphi(Y,Z|\hat{X})}[\log D_\phi(\hat{X}|Z)] \\ &- D_{\mathrm{KL}}(Q_\varphi(Y,Z|\hat{X}) \parallel E_\phi(Y|X)C(Z|Y)) \end{aligned} \tag{6.8}$$

As before, this represents an ELBO. Note though that, the first term in (6.8) refers to the quality of the message reconstruction and the second to how closely the receiver understands the sender. This is different to the previous section where the message never actually passes the communication system. The variational posterior plays a very different role there where it is assumed to be the encoder. In the joint scenario the posterior only serves to train the system, at test time, however it is of no interest. To sum it up, our proposed framework optimizes the

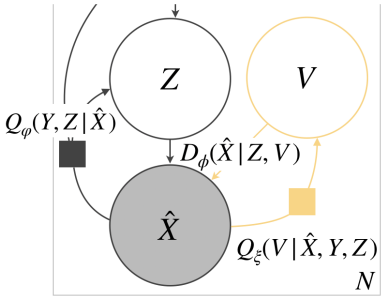

Figure 6.3: Excerpt of the graphical model in Figure 6.2 **(a)**. We show how the decoder changes when introducing auxiliary latent variables V.

actual objective of communication, the message reconstruction. For channels that do not allow for information transfer this model turns into a VAE.

Auxiliary latent variable Decoders When the information transmitted by the channel is variable, i.e. for the bandwidth-limited channel, a model has to adapt to low and high information transmission rates. To contest information loss due to a noisy channel, we propose to introduce auxiliary latent variables V to the decoder model. This model choice acknowledges the implicit marginalization over lost information. Although expected message distortion is unchanged, when sampling from such a model, message reconstructions should occur more in distribution with the true source distribution (e.g. one would expect sharper images).

We can enforce this change to the decoder by adapting the distortion term in (6.8). As before we would need to marginalize over V but choose the variational approach instead,

$$\begin{aligned} -D \geq \mathbb{E}_{Q_\varphi(Y,Z|\hat{X})}[\mathbb{E}_{Q_\xi(V|\hat{X},Y,Z)}[D_\phi(\hat{X}|Z,V)] \\ - D_{\mathrm{KL}}(Q_\xi(V|\hat{X},Y,Z)|P(V))]. \end{aligned} \tag{6.9}$$

Again, we introduced an approximate posterior to circumvent the intractable task, where $P(V)$ is a prior over these newly introduced latent variables. Just as before, the parameters of the variational distribution shall be inferred by a deep neural network with parameters ξ. We indicated the components introduced to the communication model in yellow in Figure 6.3. We note that we could execute the same idea using other conditional generative models, and corresponding inference methods, such as conditional GANs. We leave this exploration to future research.

6.5 Related Work

The field of learned image and video compression has enhanced rapidly over the past few years. While [133] give a recent concise overview of the field, here we focus on probabilistic auto-encoding approaches first proposed by [204]. The main focus of the field of image compression is to close the gap between theoretical ideas and well performing systems. One block of efforts focuses on learning representations. While

VAEs tend to work better in the continuous regime, most codes and channels can best be described by binary representations. To bridge this, it has been proposed to (i) quantize continuous representations by convolving them with a uniform distribution [13, 14, 142, 3], (ii) learn discrete representations directly [213, 11, 188] or even (iii) learn to generate common codecs e.g. JPEG [96, 126]. Note that some of these systems rely on learned priors; however, these are actually not suitable for separate coding. Other work is focused on biasing compression towards image features important for perception or system security [122, 4]. For situations where sequences of source inputs are communicated, neural buffers have also been explored to allow reordering of elements to improve code length [57]. Another branch of research focuses on the architecture of encoder and decoder models [58, 38, 232]. Additionally, there is work looking into performing tasks on compressed representations directly [206]. Important to mention also are efforts to make the often expensive encoder and decoder more computationally efficient [12].

In contrast to neural source coding, neural channel coding has yet to be explored so extensively. However, first studies [147, 62, 23, 44] demonstrate great success with neural encoder/decoder architectures. For example, it was shown that a neural model can find a solution to the Gaussian feedback channel which benefits from the feedback, a result known before but not demonstrated by any channel code yet [100, 101].

Most related to our work in spirit is a range of end-to-end learned joint communication systems. [46] apply a joint source channel system to text; [21] use auto-encoders to transmit messages over the AWGN channel; and [229] use joint systems for data compression. Closest to our work is the study by [30] where they learn the communication system in a variational fashion as well, but exclusively look at the binary erasure channel. The discrete channel leads to another variant of the learning scheme.

6.6 Experiments

We focus our experiments on the bandwidth-limited channel with additive white Gaussian noise (AWGN) and power restricted inputs. The latter ensures a limited channel capacity.

We empirically investigate our three hypothesis for improved modelling with bandwidth-limited channel. First we repeat an experiment, first performed by [30] on the Binary channel. For this, we compare neural joint and separate models, finding the joint model consistently outperforms it's separate counterpart. These findings echo aforementioned work. We therefore expand upon this by focusing the remainder of our experiments on a neural joint model with the AWGN bandwidth-limited channel. Second, we go on to show the importance different prior and decoder choices for the performance of this model.

All results are evaluated on CelebA [127]. All images were re-scaled to a resolution of 32×32 pixels. Encoders and decoders have generally been chosen to be Residual Networks [75], due to their wide usage in a range of generative modelling tasks, e.g. in [58].

Comparing Joint and Separate Neural Models with Gaussian Channel

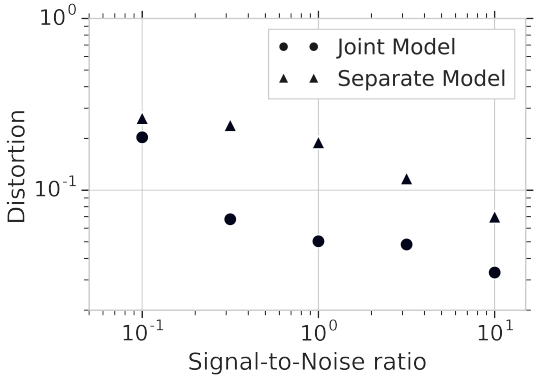

Figure 6.4: Results of comparing distortion for joint and separate neural communication systems with ResNet encoders and decoders [75, 58] at various signal-to-noise ratios for the Gaussian channel. The joint model outperforms the separate one consistently. Our results echo [30] that model with Binary channel.

As previously discussed in section 6.3, we can not predict precisely how a separate model would compare in contrast to a joint one. We hence compare separate and joint neural models as described in section 6.4 for the AWGN at various SNRs. For both models we choose the same posterior distribution for latent encoding: isotropic Gaussians. We additionally choose the observation model to be Gaussian since it is quite common to measure distortion in L2-space. Encoder and decoders of both models share the same architecture configuration. For the separate system, we choose a standard Gaussian to be the prior for the source-VAE and simultaneously the data source for the channel-AE. We note that this cannot be a data dependent prior as this would leak information to the channel coding system. For both models we hyper-optimize over a range of beta values on a log-scaled grid[2]. We optimize both models with an SGD algorithm.

[2] To get an understanding of the sensitivity to this parameter we put all results in the appendix.

We evaluate both systems by sending a message through the encoder, channel and eventually the decoder, subsequently measuring the L2-distortion between sent and received messages. The quantitative results are presented in Figure 6.4 on the left. For any of the 5 SNRs that we run our experiment on, we find the joint model outperforms the separate one. We observe, though, that the difference between the systems shrinks towards either end of the range of the SNRs presented. This effect can be explained: For very high SNRs the channel model becomes somewhat redundant, thus both systems resemble a source-VAE. At very low SNRs both systems fail to communicate as they approach the channel capacity. Our findings are in line with other recent work: [30] show that joint systems outperform hybrid models (neural source coding, hand-designed channel coding) and [100] show, for some feedback channels, learned neural models outperform hand-crafted channel codes.

Communication Model Design for Bandwidth-limited Channel

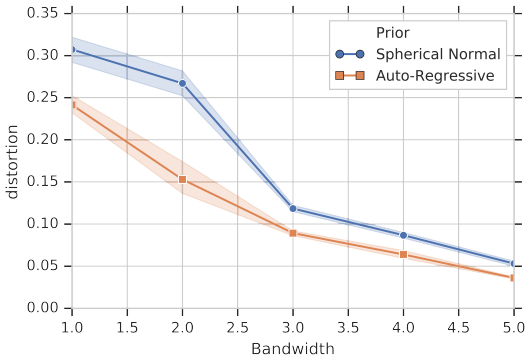

Figure 6.5: We consider joint models trained based on the AWGN bandwidth-limited channel with a fixed SNR of 1. In both figures we contrast message quality with bandwidth. The higher the bandwidth the more information is transmitted to the receiver. We measure message quality by distortion in L2-space. We compare two approximations to the channel encoding distributions. Our complex prior [58] outperforms a simpler one. Further, we observe a linear relationship between bandwidth and distortion.

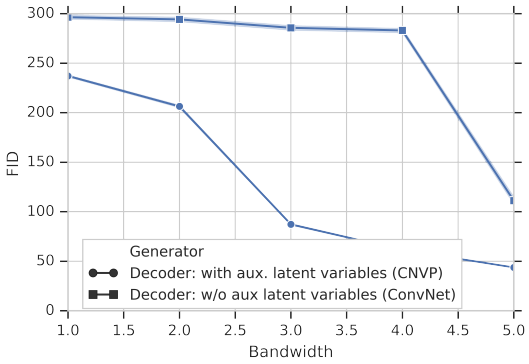

Figure 6.6: We consider joint models trained based on the AWGN bandwidth-limited channel with a fixed SNR of 1. In both figures we contrast message quality with bandwidth. The higher the bandwidth the more information is transmitted to the receiver. We measure message quality by FID score. Lower FID score is better. We compare decoders without auxiliary latent variables to decoders with auxiliary latent variables.

After verifying the importance of joint modelling for Gaussian channels, we will now investigate the performance of a joint model on the AWGN bandwidth-limited channel design we introduced in section

6.4. In this experiment we fix the SNR of the AWGN to 1.

Two choices for the model are relevant, the prior that models channel codes and the decoder. Both deal with a lack of information in the low bandwidth regime.

Prior As mentioned in the section 6.4, we require an approximation to P_{Y_i}. In our first experiment, we investigate how much the complexity of this approximation influences the quality of message reconstruction. Here we shall compare a spherical Gaussian and ConvDraw prior (see [58]) to contrast a simple with a complex approximation. We consider a 100 dimensional latent space. The space is partitioned into 5 parts. Each part representing another band. Other specifications of the experiment are equivalent to the previous section. We present our findings in Figure 6.5. We observe that, as expected, the message distortion decreases when we transmit more information, for both approximations. We additionally observe that the quality of reconstruction increases when the more complex prior model is used, and the distortion gap between priors increases when less information units are being transported through the channel. Furthermore, for both prior choices, the distortion decreases almost linearly with the bandwidth increasing. This result is in line with classic findings that show a linear relationship between channel capacity and bandwidth of an input power restricted AWGN. Finally, we shall give a visual impression of the reconstructions at various bandwidth in Figure 1 in the appendix.

Decoder For small bandwidths, we find that loss of information leads to blurry reconstructions even with learned priors. To combat this, we contrast a model without auxiliary latent variables with our proposed auxiliary latent variable model. Specifically, for these two models, we use an unconditional ConvDraw decoder and a conditional ConvDraw decoder [58] respectively. As a measure of in-distribution affiliation we use the well established FID measure [164]. This measure has mainly served to evaluate the quality of GAN samples. Smaller FID measures are better. In this experiment, we use the more complex autoregressive prior model. Other experiment details remain the same as before. The results of this experiment are presented in Figure 6.6. For both decoders, as expected, the sample quality drops for smaller bandwidth. However, the model with auxiliary latent variables significantly outperforms the one without across the full range of bandwidth presented here. We thus conclude auxiliary latent variable decoders can significantly improve the quality of communicated messages in some respects, and therefore encourage their continued exploration.

6.7 Discussion

In this chapter, we derived a generative model for joint coding with the bandwidth-limited channel and showed how to perform learning based on variational inference. For this, we introduced a differentiable and efficient model of the channel. Since back-propagation through the channel is now possible, we demonstrate how we can learn flexible function approximators for coding by Monte Carlo sampling.

To justify the usage of joint coding instead of channel coding, we first compared joint with separate communication models. Joint models were shown to consistently and significantly outperform their separate counterparts. Given joint coding as a basis, we investigate our main hypothesis that when a channel transfers little or variable amounts of information, the decoder might be helped by understanding the source distribution. We put this idea into practice by focusing on two modelling choices. First, when there is no information transferred, the decoder may draw a sample from the encoding distribution $P_Y(Z_i)$ to get a source-typical encoding. We test how the complexity of the distribution model influences reconstruction performance. We find the more complex model to improve the distortion especially in the low transmission regime. Second, when sampling message reconstructions from the communication system, missing information leads to averaged reconstructions (i.e. blurry images). We prevent this by introducing auxiliary latent variable decoders. In experiments, we show that these decoders improve message reconstruction considerably in terms FID score.

Further, this model serves as a simple method to learn a latent encoding that is sorted according to information content and channel noise, eliminating the need to pass the latent code through a lossless compressor before transmitting the data. This is an essential property for sequential information transfer. In future work, we want to explore this aspect more extensively. Future efforts in this field would focus on reinforcing our finding further by investigating the same hypothesis in other data domains and with other channels.

7

Conclusion

In this thesis, we have revisited the connection between statistical inference in DLVMs and coding by means of the MDL principle. In particular, we have focused our attention on realizing approximate Bayesian codes. We showed that fully Bayesian codes are more effective than the more common two-part codes (see Equation (1.12)).

Thus, in the first part of this thesis, we identified two approximate Bayesian codes to compress functions. We approximate the Bayesian code with the help of variational inference. The research questions of this part, hence, focus on identifying feasible coding schemes for the prior and posterior distributions assumed by the approximate inference. In answering our first research question (Research Question 1), in Chapter 3, we described how quantization and pruning can be understood as performing variational inference over latent parameter distributions. As an example, we applied our ideas to Gaussian parameter prior and posterior distributions. By learning the parameter's variance we could identify the level of quantization it can sustain. However, the map from a posterior's variance to the corresponding parameter's precision is approximate. We are hopeful that future efforts dispose of this shortcoming.

The second research question (Research Question 2) of part 1 of the thesis aimed at an alternative prior and posterior choice. We discovered that by choosing a mixture prior, we can turn the problem of sending continuous parameter distributions into a problem of sending discrete ones. In Chapter 4, we demonstrated the use of this idea by

choosing a Gaussian mixture prior and a Dirac posterior. We subsequently used Huffman encoding or arithmetic coding to compress the discrete symbols. Again the solution we propose is only an approximation of the original research question . Originally, we were hoping for the prior to be a mixture of posterior distributions for the compression scheme to work exactly. At the same time, we had to ensure that only one mixture component claims responsibility for a posterior. Thus, a Dirac posterior posed a good compromise between these two objectives.

In the second part of this book, we focus our efforts on communication with DLVMs. We zoom into two relevant applications; communication with fixed partially unknown encoding and communication with a varying level of information transfer. In Chapter 5, we showed how Research Question 3 can be answered by framing the scientific imaging process of collecting projections with partially known encoding processes as a communication process. We found that our method can discover protein structures as precise as other benchmark algorithms while allowing for construction with local uncertainty when part of the message is not possibly reconstructable. We ope that this work can be extended in the future to include inference over poses as well. For this we seek to define distributions on $SE(3)$. In Chapter 6, we identify three modelling choices when modelling with limited information transfer; (i) Instead of separating the sub-tasks of compression (source coding) and error correction (channel coding),we propose to model both jointly. (ii) We propose a design to model missing information instead of ignoring it. (iii) By introducing auxiliary latent variables in the decoder,we can sample more realistic messages.

Next, we will explore open questions in the field of coding and respective statistical inference; and discuss relevant possible applications of this line of work.

The frontiers for improving communication are wide. We believe generative modelling can improve many communication applications such as video calling and live television. These examples stick out to us because, due to the urgency of instantaneous execution, communication needs to happen most efficiently and the cost of encoding and decoding is nearly irrelevant assuming sender and receiver have enough computational resources. Generative models could improve these applications in a real sense very soon. Especially given that many applications such as skype or zoom host their own platforms where the respective owners control the data formats. Good generative models should be able to capture a conversation to the point where if a party

drops out, we would able to predict the conversation up to a certain degree. Generative modelling also allows for inpainting of missing information as demonstrated in Chapter 5. One research direction in this domain could be to look into fast (approximately) exact inference encoders, such as annealed importance sampling. This contrasts the approximate inference schemes, i.e., amortized inference we mainly rely on today.

In light of these applications, we see channel coding as a discipline that needs more attention. Even though some pioneering work has been published in the last few years [100, 101, 46, 30, 229, 147, 62, 23, 44], there are considerable challenges in incorporating feedback-channels with memory. Beyond the classical idea of modelling a simplified channel, it might be worth looking into learning the channel model itself. A naive approach would be to send random messages through a given channel. The set of random messages and received messages constitutes a learning problem. However, it is hard to say what a sufficient set of random messages would constitute when there is no information about respective sources. We may go even further with this approach and adaptively learn to adjust to a channel environment. This may be useful in situations where the signal varies or when there is more than one channel to choose from. We can also think of exploring new materials as channels by means of learning. For example, a project by Anderson et al. [6] recently discovered that storing information on glass may be more durable than on regular ferromagnets. Understanding information storage as communication over time, we may ask to identify the channel for this specific material.

There are also systematic challenges. Machine learning in communication algorithms promises highly user adapted models. Soon different users may have different personalized compression algorithms, e.g. a cat loving vs a car loving person may have different compressors to store their vast amount of cat and car pictures, respectively. However in order to make good on the promise of individualized functions for each user of a communication system, we need to rethink storage, and, more generally, communication formats. How are we going to communicate and store, data and models in the future on real systems? We consider function compression to be a vital part of this question. Devices have limited storage and computational budgets that differ from device to device. How can this be incorporated in a generalized format?

For function compression, most work has focused on reducing parameters in numbers or precision. We think it is worth investing resources

in other ways to restrict function complexity. New frontiers in function compression could evolve around restricting the function space rather than the parameter space; or acting Bayesian on a flop level. The latter includes considerations with respect to architecture search, the cost of non-linearities, etc. Potentially, we should, as originally proposed by Solomonoff [195], understand communication (literally) as communication of universal binary codes; and optimize them accordingly.

We are excited for the possibilities that casting various problems as communication problems hold for the future. Finally, we would like to emphasis that we are aware that many problem known from similar domains are also present in modelling communication; (i) Data and selection bias can cause problems when, for example, training a channel model or learning a generative model to model a conversation. (ii) Domain shifts originating from attempting to reuse a source or a channel model in a different context can lead to another form of bias.

8

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

[2] Adhikari, P. R. and Hollmén, J. (2012). Multiresolution mixture modeling using merging of mixture components.

[3] Agustsson, E., Mentzer, F., Tschannen, M., Cavigelli, L., Timofte, R., Benini, L., and Gool, L. V. (2017). Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pages 1141–1151.

[4] Agustsson, E., Tschannen, M., Mentzer, F., Timofte, R., and Van Gool, L. (2018). Generative adversarial networks for extreme learned image compression. *arXiv preprint arXiv:1804.02958*.

[5] Alemi, A. A., Poole, B., Fischer, I., Dillon, J. V., Saurous, R. A., and Murphy, K. (2017). Fixing a broken elbo. *arXiv preprint arXiv:1711.00464*.

[6] Anderson, P., Black, R., Cerkauskaite, A., Chatzieleftheriou, A., Clegg, J., Dainty, C., Diaconu, R., Drevinskas, R., Donnelly, A., Gaunt, A. L., et al. (2018). Glass: A new media for a new era? In *10th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 18)*.

[7] Andrews, D. F. and Mallows, C. L. (1974). Scale mixtures of normal

distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 99–102.

[8] Armagan, A., Clyde, M., and Dunson, D. B. (2011). Generalized beta mixtures of gaussians. In *Advances in neural information processing systems*, pages 523–531.

[9] Azarkhish, E., Rossi, D., Loi, I., and Benini, L. (2017). Neurostream: Scalable and energy efficient deep learning with smart memory cubes. *arXiv preprint arXiv:1701.06420*.

[10] Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

[11] Ballé, J., Johnston, N., and Minnen, D. (2018a). Integer networks for data compression with latent-variable models.

[12] Ballé, J., Laparra, V., and Simoncelli, E. P. (2015). Density modeling of images using a generalized normalization transformation. *arXiv preprint arXiv:1511.06281*.

[13] Ballé, J., Laparra, V., and Simoncelli, E. P. (2016). End-to-end optimized image compression. *CoRR*, abs/1611.01704.

[14] Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. (2018b). Variational image compression with a scale hyperprior. *CoRR*, abs/1802.01436.

[15] Barron, A. R. and Cover, T. M. (1991). Minimum complexity density estimation. *IEEE transactions on information theory*, 37(4):1034–1054.

[16] Baxter, W. T., Grassucci, R. A., Gao, H., and Frank, J. (2009). Determination of signal-to-noise ratios and spectral snrs in cryo-em low-dose imaging of molecules. *Journal of structural biology*, 166(2):126–132.

[17] Beale, E., Mallows, C., et al. (1959). Scale mixing of symmetric distributions with zero means. *The Annals of Mathematical Statistics*, 30(4):1145–1151.

[18] Berlekamp, E., McEliece, R., and Van Tilborg, H. (1978). On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386.

[19] Bishop, C. M. (2006). Pattern recognition. *Machine Learning*.

[20] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*.

[21] Bourtsoulatze, E., Kurka, D. B., and Gündüz, D. (2019). Deep joint source-channel coding for wireless image transmission. *IEEE Transactions on Cognitive Communications and Networking*.

[22] Brubaker, M. A., Punjani, A., and Fleet, D. J. (2015). Building proteins in a day: Efficient 3d molecular reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3099–3108.

[23] Cammerer, S., Gruber, T., Hoydis, J., and ten Brink, S. (2017). Scaling deep learning-based decoding of polar codes via partitioning. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE.

[24] Cardone, G., Heymann, J. B., and Steven, A. C. (2013). One number does not fit all: Mapping local variations in resolution in cryo-em reconstructions. *Journal of structural biology*, 184(2):226–236.

[25] Carvalho, C. M., Polson, N. G., and Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika*, 97(2):465–480.

[26] Chai, S., Raghavan, A., Zhang, D., Amer, M., and Shields, T. (2017). Low precision neural networks using subband decomposition. *arXiv preprint arXiv:1703.08595*.

[27] Chaitin, G. J. (1969). On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the ACM (JACM)*, 16(1):145–159.

[28] Chen, W., Wilson, J. T., Tyree, S., Weinberger, K. Q., and Chen, Y. (2015). Compressing convolutional neural networks. *arXiv preprint arXiv:1506.04449*.

[29] Chirikjian, G. and Kyatkin, A. (2000). *Engineering Applications of Noncommutative Harmonic Analysis: With Emphasis on Rotation and Motion Groups*. CRC Press.

[30] Choi, K., Tatwawadi, K., Grover, A., Weissman, T., and Ermon, S. (2019). Neural joint source-channel coding. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 1182–1192.

[31] Courbariaux, M. and Bengio, Y. (2016). Binarynet: Training deep neural networks with weights and activations constrained to $+1$ or -1. *arXiv preprint arXiv:1602.02830*.

[32] Courbariaux, M., Bengio, Y., and David, J.-P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pages 3123–3131.

[33] Courbariaux, M., David, J.-P., and Bengio, Y. (2014). Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*.

[34] Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks: Training neural networks with weights and activations constrained to $+1$ or -1.

[35] Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

[36] Cox, R. T. (1946). Probability, frequency and reasonable expectation. *American journal of physics*, 14(1):1–13.

[37] Csiszar, I. (1982). Linear codes for sources and source networks: Error exponents, universal coding. *IEEE Transactions on Information Theory*, 28(4):585–592.

[38] De Fauw, J., Dieleman, S., and Simonyan, K. (2019). Hierarchical autoregressive image models with auxiliary decoders. *arXiv preprint arXiv:1903.04933*.

[39] De la Rosa-Trevín, J., Otón, J., Marabini, R., Zaldivar, A., Vargas, J., Carazo, J., and Sorzano, C. (2013). Xmipp 3.0: an improved software suite for image processing in electron microscopy. *Journal of structural biology*, 184(2):321–328.

[40] Denil, M., Shakibi, B., Dinh, L., de Freitas, N., et al. (2013). Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pages 2148–2156.

[41] Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277.

[42] Dong, X., Chen, S., and Pan, S. (2017a). Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *NeurIPS*, pages 4857–4867.

[43] Dong, X., Huang, J., Yang, Y., and Yan, S. (2017b). More is less: A more complicated network with less inference complexity. *arXiv preprint arXiv:1703.08651*.

[44] Dörner, S., Cammerer, S., Hoydis, J., and ten Brink, S. (2017). Deep learning based communication over the air. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):132–143.

[45] Egelman, E. H. (2016). The current revolution in cryo-em. *Biophysical journal*, 110(5):1008–1012.

[46] Farsad, N., Rao, M., and Goldsmith, A. (2018). Deep learning for joint source-channel coding of text. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2326–2330. IEEE.

[47] Faruqi, A., Cattermole, D., Henderson, R., Mikulec, B., and Raeburn, C. (2003). Evaluation of a hybrid pixel detector for electron microscopy. *Ultramicroscopy*, 94(3-4):263–276.

[48] Federici, M., Ullrich, K., and Welling, M. (2017). Improved bayesian compression. *Bayesian Deep Learning Workshop at NeurIPS*.

[49] Feldman, J., Wainwright, M. J., and Karger, D. R. (2005). Using linear programming to decode binary linear codes. *IEEE Transactions on Information Theory*, 51(3):954–972.

[50] Figueiredo, M. A. (2002). Adaptive sparseness using jeffreys' prior. *Advances in neural information processing systems*, 1:697–704.

[51] Frey, B. J., Brendan, J. F., and Frey, B. J. (1998). *Graphical models for machine learning and digital communication*. MIT press.

[52] Frey, B. J. and Hinton, G. E. (1997). Efficient stochastic source coding and an application to a bayesian network source model. *The Computer Journal*, 40(2_and_3):157–165.

[53] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *ICML*.

[54] Gal, Y., Hron, J., and Kendall, A. (2017). Concrete dropout. In *NeurIPS*, pages 3581–3590.

[55] Gong, Y., Liu, L., Yang, M., and Bourdev, L. (2014). Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

[56] Graves, A. (2011). Practical variational inference for neural networks. In *NeurIPS*.

[57] Graves, A., Menick, J., and Oord, A. v. d. (2018). Associative compression networks. *arXiv preprint arXiv:1804.02476*.

[58] Gregor, K., Besse, F., Rezende, D. J., Danihelka, I., and Wierstra, D. (2016). Towards conceptual compression. In *NIPS*.

[59] Gregor, K., Danihelka, I., Mnih, A., Blundell, C., and Wierstra, D. (2013). Deep autoregressive networks. *arXiv preprint arXiv:1310.8499*.

[60] Grigorieff, N. (1998). Three-dimensional structure of bovine nadh: ubiquinone oxidoreductase (complex i) at 22 å in ice. *Journal of molecular biology*, 277(5):1033–1046.

[61] Grigorieff, N. (2007). Frealign: high-resolution refinement of single particle structures. *Journal of structural biology*, 157(1):117–125.

[62] Gruber, T., Cammerer, S., Hoydis, J., and ten Brink, S. (2017). On deep learning-based channel decoding. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE.

[63] Grünwald, P. D. (2007). *The minimum description length principle*. MIT press.

[64] Guo, Y., Yao, A., and Chen, Y. (2016). Dynamic network surgery for efficient dnns. In *NeurIPS*.

[65] Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. (2015). Deep learning with limited numerical precision. *CoRR, abs/1502.02551*, 392.

[66] Gysel, P. (2016). Ristretto: Hardware-oriented approximation of convolutional neural networks. *Master's thesis, University of California*.

[67] Han, S., Mao, H., and Dally, W. J. (2015). Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR, abs/1510.00149*, 2.

[68] Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*.

[69] Han, S., Pool, J., Narang, S., Mao, H., Gong, E., Tang, S., Elsen, E., Vajda, P., Paluri, M., Tran, J., et al. (2017). Dsd: Dense-sparse-dense training for deep neural networks. *ICLR*.

[Han et al.] Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *NeurIPS*.

[71] Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *NeurIPS*, pages 164–171.

[72] Havin, V. and JÃűricke, B. (1994). *The Uncertainty Principle in Harmonic Analysis*. Springer-Verlag.

[73] He, K., Zhang, X., Ren, S., and Sun, J. (2015a). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

[74] He, K., Zhang, X., Ren, S., and Sun, J. (2015b). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

[75] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[76] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*.

[77] He, Y., Zhang, X., and Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *CVPR*, pages 1389–1397.

[78] Henderson, R., Sali, A., Baker, M. L., Carragher, B., Devkota, B., Downing, K. H., Egelman, E. H., Feng, Z., Frank, J., Grigorieff, N., et al. (2012). Outcome of the first electron microscopy validation task force meeting. *Structure*, 20(2):205–214.

[79] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 3.

[80] Hinton, G. and Van Camp, D. (1993a). Keeping neural networks simple by minimizing the description length of the weights. In *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*. Citeseer.

[81] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[82] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[83] Hinton, G. E. and Van Camp, D. (1993b). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM.

[84] Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length and helmholtz free energy. In *Advances in neural information processing systems*, pages 3–10.

[85] Ho, J., Lohn, E., and Abbeel, P. (2019). Compression with flows via local bits-back coding. In *Advances in Neural Information Processing Systems*, pages 3874–3883.

[86] Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

[87] Honkela, A. and Valpola, H. (2004). Variational learning and bits-back coding: an information-theoretic view to bayesian learning. *IEEE Transactions on Neural Networks*, 15(4):800–810.

[88] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[89] Hu, H., Peng, R., Tai, Y.-W., and Tang, C.-K. (2016). Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*.

[90] Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 1mb model size. *arXiv preprint arXiv:1602.07360*.

[91] Ioannou, Y., Robertson, D., Shotton, J., Cipolla, R., and Criminisi, A. (2015). Training cnns with low-rank filters for efficient image classification. *arXiv preprint arXiv:1511.06744*.

[92] Jaakkola, T. S. (1997). *Variational methods for inference and estimation in graphical models*. PhD thesis, Massachusetts Institute of Technology.

[93] Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025.

[94] Jaderberg, M., Vedaldi, A., and Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*.

[95] Jaitly, N., Brubaker, M. A., Rubinstein, J. L., and Lilien, R. H. (2010). A bayesian method for 3d macromolecular structure inference using class average images from single particle electron microscopy. *Bioinformatics*, 26(19):2406–2415.

[96] Jiang, F., Tao, W., Liu, S., Ren, J., Guo, X., and Zhao, D. (2017). An end-to-end compression framework based on convolutional neural

networks. *IEEE Transactions on Circuits and Systems for Video Technology*, PP:1–1.

[97] Jin, X., Yuan, X., Feng, J., and Yan, S. (2016). Training skinny deep neural networks with iterative hard thresholding methods. *arXiv preprint arXiv:1607.05423*.

[98] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1998). An introduction to variational methods for graphical models. In *Learning in graphical models*, pages 105–161. Springer.

[99] Karaletsos, T. and Rätsch, G. (2015). Automatic relevance determination for deep generative models. *arXiv preprint arXiv:1505.07765*.

[100] Kim, H., Jiang, Y., Kannan, S., Oh, S., and Viswanath, P. (2018a). Deepcode: Feedback codes via deep learning. In *Advances in Neural Information Processing Systems*, pages 9458–9468.

[101] Kim, H., Jiang, Y., Rana, R., Kannan, S., Oh, S., and Viswanath, P. (2018b). Communication algorithms via deep learning. *arXiv preprint arXiv:1805.09317*.

[102] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[103] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR), San Diego*.

[104] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751.

[105] Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparametrization trick. *Advances in Neural Information Processing Systems*.

[106] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[107] Kingma, F. H., Abbeel, P., and Ho, J. (2019). Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. *arXiv preprint arXiv:1905.06845*.

[108] Koetter, R. and Vontobel, P. O. (2003). Graph-covers and iterative decoding of finite length codes. In *Proc. 3rd Intern. Symp. on Turbo Codes and Related Topics*, pages 1–5. Citeseer.

[109] Kohl, H. and Reimer, L. (2008). *Transmission electron microscopy: physics of image formation*. Springer.

[110] Kolmogorov, A. N. (1965). Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, 1(1):1–7.

[111] Kostina, V. and Verdú, S. (2013). Lossy joint source-channel coding in the finite blocklength regime. *IEEE Transactions on Information Theory*, 59(5):2545–2575.

[112] Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

[113] Kucukelbir, A., Sigworth, F. J., and Tagare, H. D. (2014). Quantifying the local resolution of cryo-em density maps. *Nature methods*, 11(1):63.

[114] Kumar, A., Eslami, S., Rezende, D. J., Garnelo, M., Viola, F., Lockhart, E., and Shanahan, M. (2018). Consistent generative query networks. *arXiv preprint arXiv:1807.02033*.

[115] Langlois, R., Pallesen, J., Ash, J. T., Ho, D. N., Rubinstein, J. L., and Frank, J. (2014). Automated particle picking for low-contrast macromolecules in cryo-electron microscopy. *Journal of structural biology*, 186(1):1–7.

[116] Lawrence, N. D. (2002). Note relevance determination. In *Neural Nets WIRN Vietri-01*, pages 128–133. Springer.

[117] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[118] LeCun, Y., Cortes, C., and Burges, C. J. (1998b). The mnist database of handwritten digits.

[119] LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *NeurIPS*, pages 598–605.

[120] Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. (2017). Pruning filters for efficient convnets. *ICLR*.

[121] Li, M., Vitányi, P., et al. (2008). *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer.

[122] Li, M., Zuo, W., Gu, S., Zhao, D., and Zhang, D. (2018). Learning convolutional networks for content-weighted image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3214–3223.

[123] Lin, D. D. and Talathi, S. S. (2016). Overcoming challenges in fixed point training of deep convolutional networks. *Workshop ICML*.

[124] Lin, D. D., Talathi, S. S., and Annapureddy, V. S. (2015). Fixed point quantization of deep convolutional networks. *arXiv preprint arXiv:1511.06393*.

[125] Liu, B., Wang, M., Foroosh, H., Tappen, M., and Pensky, M. (2015a). Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814.

[126] Liu, Z., Liu, T., Wen, W., Jiang, L., Xu, J., Wang, Y., and Quan, G. (2018). Deepn-jpeg: a deep neural network favorable jpeg-based image compression framework. In *Proceedings of the 55th Annual Design Automation Conference*, page 18. ACM.

[127] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015b). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[128] Louizos, C. (2015). Smart regularization of deep architectures. *Master's thesis, University of Amsterdam*.

[129] Louizos, C., Ullrich, K., and Welling, M. (2017). Bayesian compression for deep learning. In *NeurIPS*, pages 3288–3298.

[130] Louizos, C. and Welling, M. (2017). Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. *ArXiv e-prints*.

[131] Louizos, C., Welling, M., and Kingma, D. P. (2018). Learning sparse neural networks through *l*_0 regularization. *ICLR*.

[132] Luo, J.-H., Wu, J., and Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In *CVPR*, pages 5058–5066.

[133] Ma, S., Zhang, X., Jia, C., Zhao, Z., Wang, S., and Wanga, S. (2019). Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology*.

[134] MacKay, D. J. (1995a). Developments in probabilistic modelling with neural networksâĂŤensemble learning. In *Neural Networks: Artificial Intelligence and Industrial Applications*, pages 191–198. Springer.

[135] MacKay, D. J. (1995b). Probable networks and plausible predictionsâĂŤa review of practical bayesian methods for supervised

neural networks. *Network: Computation in Neural Systems*, 6(3):469–505.

[136] Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, abs/1611.00712.

[137] Mariet, Z. and Sra, S. (2016). Diversity networks: Neural network compression using determinantal point processes. *ICLR*.

[138] Marino, J., Yue, Y., and Mandt, S. (2018). Iterative amortized inference. *arXiv preprint arXiv:1807.09356*.

[139] Mathieu, M. (2015). Masked autoencoder for distribution estimation.

[140] Mellempudi, N., Kundu, A., Mudigere, D., Das, D., Kaul, B., and Dubey, P. (2017). Ternary neural networks with fine-grained quantization. *arXiv preprint arXiv:1705.01462*.

[141] Merolla, P., Appuswamy, R., Arthur, J., Esser, S. K., and Modha, D. (2016). Deep neural networks are robust to weight binarization and other non-linear distortions. *arXiv preprint arXiv:1606.01981*.

[142] Minnen, D., Ballé, J., and Toderici, G. D. (2018). Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780.

[143] Mitchell, T. J. and Beauchamp, J. J. (1988). Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032.

[144] Mnih, A. and Rezende, D. J. (2016). Variational inference for monte carlo objectives. *arXiv preprint arXiv:1602.06725*.

[145] Molchanov, D., Ashukha, A., and Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*.

[146] Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

[147] Nachmani, E., Be'ery, Y., and Burshtein, D. (2016). Learning to decode linear codes using deep learning. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 341–346. IEEE.

[148] Nalisnick, E., Anandkumar, A., and Smyth, P. (2015). A scale mixture perspective of multiplicative noise in neural networks. *arXiv preprint arXiv:1506.03208*.

[149] Neal, R. M. (1995). *Bayesian learning for neural networks*. PhD thesis, Citeseer.

[150] Neklyudov, K., Molchanov, D., Ashukha, A., and Vetrov, D. P. (2017). Structured bayesian pruning via log-normal multiplicative noise. In *NeurIPS*, pages 6775–6784.

[151] Neville, S. E., Ormerod, J. T., Wand, M., et al. (2014). Mean field variational bayes for continuous sparse signal shrinkage: pitfalls and remedies. *Electronic Journal of Statistics*, 8(1):1113–1151.

[152] Newton, M. A. and Raftery, A. E. (1994). Approximate bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 3–48.

[153] Nowlan, S. J. and Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493.

[154] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[155] Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016b). Pixel recurrent neural networks. *ICML*.

[156] Paisley, J., Blei, D., and Jordan, M. (2012). Variational bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*.

[157] Papaspiliopoulos, O., Roberts, G. O., and Sköld, M. (2007). A general framework for the parametrization of hierarchical models. *Statistical Science*, pages 59–73.

[158] Peters, J. W. and Welling, M. (2018). Probabilistic binary neural networks. *arXiv preprint arXiv:1809.03368*.

[159] Peterson, C. (1987). A mean field theory learning algorithm for neural networks. *Complex systems*, 1:995–1019.

[160] Pettersen, E. F., Goddard, T. D., Huang, C. C., Couch, G. S., Greenblatt, D. M., Meng, E. C., and Ferrin, T. E. (2004). Ucsf chimeraâĂŤa visualization system for exploratory research and analysis. *Journal of computational chemistry*, 25(13):1605–1612.

[161] Pilc, R. J. (1967). *Coding theorems for discrete source-channel pairs.* PhD thesis, Massachusetts Institute of Technology.

[162] Pintilie, G. (2010). Greg pintilie's homepage: `people.csail.mit.edu/gdp/cryoem.html`.

[163] Punjani, A., Rubinstein, J. L., Fleet, D. J., and Brubaker, M. A. (2017). cryosparc: algorithms for rapid unsupervised cryo-em structure determination. *Nature Methods*, 14(3):290–296.

[164] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[165] Ranganath, R., Gerrish, S., and Blei, D. M. (2014). Black box variational inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*.

[166] Ranganath, R., Tran, D., and Blei, D. (2016). Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333.

[167] Ranzato, M., Poultney, C., Chopra, S., and Cun, Y. L. (2007). Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144.

[168] Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer.

[169] Rezende, D. J., Eslami, S. A., Mohamed, S., Battaglia, P., Jaderberg, M., and Heess, N. (2016). Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems*, pages 4996–5004.

[170] Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. *ICML*.

[171] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014a). Stochastic backpropagation and approximate inference in deep generative models. In *ICML*.

[172] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014b). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1278–1286.

[173] Rissanen, J. (1978a). Modeling by shortest data description. *Automatica*, 14(5):465–471.

[174] Rissanen, J. (1978b). Modeling by shortest data description. *Automatica*, 14(5):465–471.

[175] Rissanen, J. (1983). A universal data compression system. *IEEE Transactions on information theory*, 29(5):656–664.

[176] Rissanen, J. (1986). Stochastic complexity and modeling. *The annals of statistics*, pages 1080–1100.

[177] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[178] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). Fitnets: Hints for thin deep nets. *ICLR*.

[179] Rosenthal, P. B. and Henderson, R. (2003). Optimal determination of particle orientation, absolute hand, and contrast loss in single-particle electron cryomicroscopy. *Journal of Molecular Biology*, 333(4):721–745.

[180] Rupp, B. (2009). *Biomolecular crystallography: principles, practice, and application to structural biology*. Garland Science.

[181] Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*.

[182] Salimans, T., Kingma, D., and Welling, M. (2015). Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226.

[183] Scardapane, S., Comminiello, D., Hussain, A., and Uncini, A. (2016). Group sparse regularization for deep neural networks. *arXiv preprint arXiv:1607.00485*.

[184] Scheres, S. H. (2012). Relion: implementation of a bayesian approach to cryo-em structure determination. *Journal of structural biology*, 180(3):519–530.

[185] Scheres, S. H., Gao, H., Valle, M., Herman, G. T., Eggermont, P. P., Frank, J., and Carazo, J.-M. (2007a). Disentangling conformational states of macromolecules in 3d-em through likelihood optimization. *Nature methods*, 4(1):27.

[186] Scheres, S. H., Núñez-Ramírez, R., Gómez-Llorente, Y., San Martín, C., Eggermont, P. P., and Carazo, J. M. (2007b). Modeling experimental image formation for likelihood-based classification of electron microscopy data. *Structure*, 15(10):1167–1177.

[187] Shannon, C. E. (1948). A mathematical theory of communication, part ii. *Bell Syst. Tech. J.*, 27:623–656.

[188] Shen, Y., Liu, L., and Shao, L. (2019). Unsupervised binary representation learning with deep variational networks. *International Journal of Computer Vision*, pages 1–15.

[189] Shi, S. and Chu, X. (2017). Speeding up convolutional neural networks by exploiting the sparsity of rectifier units. *arXiv preprint arXiv:1704.07724*.

[190] Sigworth, F. (1998). A maximum-likelihood approach to single-particle image refinement. *Journal of structural biology*, 122(3):328–339.

[191] Sigworth, F. J. (2016). Principles of cryo-em single-particle image processing. *Microscopy*, 65(1):57–67.

[192] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *ICLR*.

[193] Sites, M. (2008). Ieee standard for floating-point arithmetic.

[194] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

[195] Solomonoff, R. J. (1964). A formal theory of inductive inference. part i part 2. *Information and control*, 7(1):1–22.

[196] Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016). Ladder variational autoencoders. *arXiv preprint arXiv:1602.02282*.

[197] Srinivas, S. and Babu, R. V. (2015). Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*.

[198] Srinivas, S. and Babu, R. V. (2016). Generalized dropout. *arXiv preprint arXiv:1611.06791*.

[199] Srinivas, S., Subramanya, A., and Venkatesh Babu, R. (2017). Training sparse neural networks. In *CVPR Workshops*, pages 138–145.

[200] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[201] Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. (2017). Efficient processing of deep neural networks: A tutorial and survey. *arXiv preprint arXiv:1703.09039*.

[202] Tai, C., Xiao, T., Wang, X., et al. (2015). Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*.

[203] Tang, G., Peng, L., Baldwin, P. R., Mann, D. S., Jiang, W., Rees, I., and Ludtke, S. J. (2007). Eman2: an extensible image processing suite for electron microscopy. *Journal of structural biology*, 157(1):38–46.

[204] Theis, L., Shi, W., Cunningham, A., and Huszár, F. (2017). Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*.

[205] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

[206] Torfason, R., Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., and Van Gool, L. (2018). Towards image understanding from deep compression without decoding. *arXiv preprint arXiv:1803.06131*.

[207] Townsend, J., Bird, T., and Barber, D. (2019). Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*.

[208] Tschannen, M., Agustsson, E., and Lucic, M. (2018). Deep generative models for distribution-preserving lossy compression. In *NeurIPS*.

[209] Ullrich, K., Meeds, E., and Welling, M. (2017). Soft weight-sharing for neural network compression. *ICLR*.

[210] Unser, M., Trus, B. L., and Steven, A. C. (1987). A new resolution criterion based on spectral signal-to-noise ratios. *Ultramicroscopy*, 23(1):39–51.

[211] Uria, B., Côté, M.-A., Gregor, K., Murray, I., and Larochelle, H. (2016). Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220.

[212] Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798.

[213] van den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

[214] Venkatesh, G., Nurvitadhi, E., and Marr, D. (2016). Accelerating deep convolutional networks using low-precision and sparsity. *arXiv preprint arXiv:1610.00324*.

[215] Vilas, J. L., Gómez-Blanco, J., Conesa, P., Melero, R., de la Rosa-Trevín, J. M., Otón, J., Cuenca, J., Marabini, R., Carazo, J. M., Vargas, J., et al. (2018). Monores: automatic and accurate estimation of local resolution for electron microscopy maps. *Structure*, 26(2):337–344.

[216] Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

[217] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408.

[218] Vontobel, P. O. and Koetter, R. (2007). On low-complexity linear-programming decoding of ldpc codes. *European transactions on telecommunications*, 18(5):509–517.

[219] Vulović, M., Ravelli, R. B., van Vliet, L. J., Koster, A. J., Lazić, I., Lücken, U., Rullgård, H., Öktem, O., and Rieger, B. (2013). Image formation modeling in cryo-electron microscopy. *Journal of structural biology*, 183(1):19–32.

[220] Wallace, C. S. (1990). Classification by minimum-message-length inference. In *International Conference on Computing and Information*, pages 72–81. Springer.

[221] Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. (2016). Learning structured sparsity in deep neural networks. In *NeurIPS*, pages 2074–2082.

[222] Witten, I. H., Neal, R. M., and Cleary, J. G. (1987). Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540.

[223] Xu, Z., Horwich, A. L., and Sigler, P. B. (1997). The crystal structure of the asymmetric groel–groes–(adp) 7 chaperonin complex. *Nature*, 388(6644):741.

[224] Yang, T.-J., Chen, Y.-H., and Sze, V. (2017). Designing energy-efficient convolutional neural networks using energy-aware pruning. *CVPR*.

[225] Ye, S., Zhang, T., Zhang, K., Li, J., Xu, K., Yang, Y., Yu, F., Tang, J., Fardad, M., Liu, S., et al. (2018). Progressive weight pruning of deep neural networks using admm. *arXiv preprint arXiv:1810.07378*.

[226] You, Z., Yan, K., Ye, J., Ma, M., and Wang, P. (2019). Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *NeurIPS*, pages 2130–2141.

[227] Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V. I., Han, X., Gao, M., Lin, C.-Y., and Davis, L. S. (2018). Nisp: Pruning networks using neuron importance score propagation. In *CVPR*, pages 9194–9203.

[228] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

[229] Zarcone, R., Paiton, D., Anderson, A., Engel, J., Wong, H. P., and Olshausen, B. (2018). Joint source-channel coding with neural networks for analog data compression and storage. In *2018 Data Compression Conference*, pages 147–156. IEEE.

[230] Zhang, T., Ye, S., Zhang, K., Ma, X., Liu, N., Zhang, L., Tang, J., Ma, K., Lin, X., Fardad, M., et al. (2018). Structadmm: A systematic, high-efficiency framework of structured weight pruning for dnns. *arXiv preprint arXiv:1807.11091*.

[231] Zhao, J., Brubaker, M. A., and Rubinstein, J. L. (2013). Tmacs: A hybrid template matching and classification system for partially-automated particle selection. *Journal of structural biology*, 181(3):234–242.

[232] Zhou, L., Cai, C., Gao, Y., Su, S., and Wu, J. (2018). Variational autoencoder for low bit-rate image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2617–2620.

[233] Zhu, C., Han, S., Mao, H., and Dally, W. J. (2017). Trained ternary quantization. *ICLR*.

Acknowledgements

I would like to thank my supervisor *Max Welling* for providing opportunities, giving me the chance to pursue a PhD topic suitable to my interests, and for his guidance and advice. I have learned and developed a lot during the past 4 years. I would like to thank my co-promoter *Patrick Forré* for always making the time to explain literally anything to me. Further, I would like to extend my gratitude to *Ted Meeds* who believed in me and supported me with many measurable actions for a long period of time. Another one of my academic mentors is *Rianne van den Berg* who showed me how to navigate many invisible barriers. I would also like to thank *Marcus Brubaker* and *David Fleet*, both of whom invested in me and taught me so much during a multiple year long collaboration. During my internship I met *Danilo J. Rezende*, who in his kindness invested so much time into me. I have learned so much in a short amount of time and I will carry this huge privilege for a long time to come. I would also like to thank *Christos Louizos* and *Fabio Viola*, excellent engineers and scientists who patiently molded me into a better software engineer.

I am most grateful to all of you. Thank you, you made my development possible.

I would like to thank *Jorn Peters* for designing figures for this thesis and *Fabian Stübner* and *Annegret Jahn* for designing its cover. Further, I thank *Elise van der Pol* and *Andy Keller* for translation and proof reading the thesis.

Of course, I need to thank my amazing friends, lab mates and family for having the patience to stick with me. There are too many supporters and too many actions to name all of them. Know that I am deeply grateful to all of you.

9

Appendix

9.1 Appendix Chapter 2

A. Detailed experimental setup

Our method was implemented with Tensorflow [1]. We used the Adam [103] optimization scheme with the default hyperparameters to learn parameters. To initialize the means of the conditional Gaussian $q_\phi(\mathbf{W}|\mathbf{z})$, we use the technique proposed by He et al. [74]. The log of the standard deviations, on the other hand, were initialized by sampling from the distribution $\mathcal{N}(-9, 1e-4)$. Further we initialised $q_\phi(\mathbf{z})$ to have an overall mean $\mathbf{z} \approx 1$ and a low overall variance ($\approx 1e-8$). This is necessary to make sure that all groups are active in the initial phase of training.

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*

One of the more important tricks to make our method work is to constrain the standard deviations. For LeNet-300-100 we constrained the first layer's standard deviation ≤ 0.2, for LeNet-5-Caffe we to ≤ 0.5. Other layers stayed unconstrained. For the VGG architecture we constrained the 64 and 128 feature maps layers to ≤ 0.1, and the 256 feature map layers to ≤ 0.2. Any other layers were again left unconstrained. We used the "warm-up" method as proposed by [196]. For this we slowly annealed the negative KL-divergence from the approximate posterior to the prior using a linear schedule during the first 100

epochs. In order to not train VGG from scratch, we took the pretrained VGG and initialized our posterior means accordingly. During training, we disabled batch-normalization. As is usually the case we rescaled MNIST to a range $[-1, 1]$. For CIFAR 10, we tool the preprocessed dataset by [228]. AS noted before, when computing the final architecture, we use pruned outputs to inform pruning of inputs. This hold true for both fully connected and convolutional layers.

B. Standards for Floating-Point Arithmetic

Floating points values eventually need to be represented in a binary basis in a computer. The most common standard today is the IEEE 754-2008 convention [193]. It defines x-bit base-2 formats, officially referred to as binary$x$, with $x \in \{16, 32, 64, 128\}$. The formats are also widely known as half, single, double and quadruple precision floats, respectively and used in almost all programming languages as a standard. The format considers 3 kinds of bits: one sign bit, w exponent bits and p precision bits.

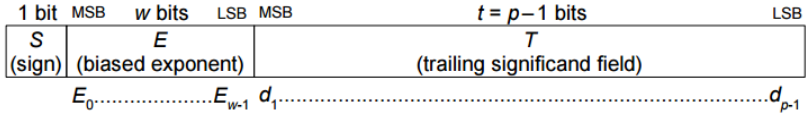

Figure 9.1: A symbolic representation of the binary$x$ format [193].

The Sign bit determines the sign of the number to be represented. The exponent E is an w-bit signed integer, e.g. for single precision $w = 8$ and thus $E \in [-127, 128]$. In practice, exponents range from is smaller since the first and the last number are reserved for special numbers. The true significand or mantissa includes t bits on the right of the binary point. There is an implicit leading bit with value one. A values is consequently decomposed as follows

$$\text{mantissa} = 1 + \sum_{i=1}^{t} b_i 2^{-i} \tag{9.1}$$

$$\text{value} = (-1)^{\text{sign bit}} \times 2^{E} \times \text{mantissa} \tag{9.2}$$

In table 9.1, we summarize common and less common floating point formats.

There is however the possibility to design a self defined format. There are 3 important quantities when choosing the right specification: overflow, underflow and unit round off also known as machine precision.

Bits per Float	Exponent width [bit]	Significand precision [bit]	underflow level	overflow level	unit roundoff
64	11	52	2.22×10^{-308}	1.79×10^{308}	2.22×10^{-16}
32	8	23	1.17×10^{-38}	3.40×10^{38}	1.19×10^{-7}
16	5	10	6.10×10^{-05}	6.54×10^{4}	9.76×10^{-4}

Table 9.1: Floating point formats

Each one can be computed knowing the number of exponent and significant bits. in our work for example we consider a format that uses significantly less exponent bits since network parameters usually vary between [-10,10]. We set the unit round off equal to the precision and thus can compute the significant bits necessary to represent a specific weight.

Beyond designing a tailored floating point format for deep learning, recent work also explored the possibility of deep learning with mixed formats [123, 66]. For example, imagine the activations having high precision while weights can be low precision.

C. Shrinkage properties of the normal-Jeffreys and horseshoe priors

In this section, we shall compare the two priors proposed in chapter 3. For this we rely on hte analysi provided by [25]. When performing a change of variables, we can express the scale mixture distribution (see Equation 3.3) as function of the shrinkage coefficient, $\lambda = \frac{1}{1+z^2}$:

$$\lambda \sim p(\lambda); \qquad w \sim \mathcal{N}\left(0, \frac{1-\lambda}{\lambda}\right) \tag{9.3}$$

In this parameterization, we can infer that our method is a to a continuous relaxation of the spike-and-slab prior. To do so we choose $\lambda = 0$, thus $p(w|\lambda = 0) = \mathcal{U}(-\infty, \infty)$. In other words, there is no shrinkage/regularization for w. On the other hand, when $\lambda = 1$, $p(w|\lambda = 1) = \delta(w = 0)$ follows. This in term means w is exactly zero. If we now set $\lambda = \frac{1}{2}$ we obtain $p(w|\lambda = \frac{1}{2}) = \mathcal{N}(0, 1)$.

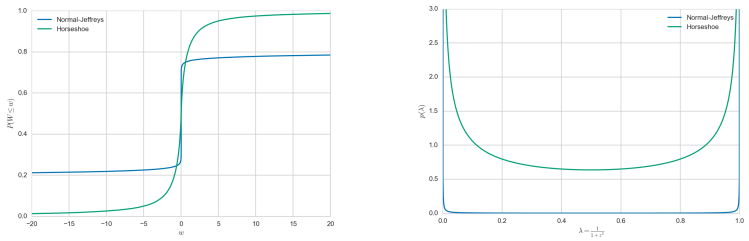

Figure 9.2: We compare the behavior of the log-uniform / normal-Jeffreys (NJ) prior against the horseshoe (HS) prior (where $s = 1$). At zero, both show similar behaviour. However, the normal-Jeffreys prior displays an extremely heavy tail.

We can now compare the log-uniform and the horseshoe prior by examining the implied prior on λ. Following [25], we note that the normal-Jeffreys / log-uniform prior on z correlate with the following

prior on the shrinkage coefficient $p(\lambda) = \mathcal{B}(\epsilon, \epsilon)$ with $\epsilon \approx 0$; and the half-Cauchy prior on z to a beta prior, $p(\lambda) = \mathcal{B}(\frac{1}{2}, \frac{1}{2})$. We plot the densities of both distributions in Figure **??**. We notice that the log-uniform prior exhibits a distribution that concentrates most of its mass at either $\lambda \approx 0$ or $\lambda \approx 1$. This means, the prior causes to either prune the parameters or maintain them close to their ML estimate[2] . The horseshoe prior on the other hand keeps some probability mass for λ between the extremes. This behaviour might potentially, benefit regularization and generalization.

[2] Becuase $p(w|\lambda \approx 1) = \mathcal{U}(-\infty, \infty)$.

D. Negative KL-divergences for log-normal approximating posteriors

In this section we will compute the negative KL-divergences from $q(z)$ to inverse gamma, and gamma to half-normal distribution. For this, we first consider the log-normal approximating posterior to be $q(z) = \mathcal{LN}(\mu, \sigma^2)$, $p(z)$ shall be an inverse gamma distribution, $p(z) = \mathcal{IG}(\alpha, \beta)$. We can now express the KL-divergence as:

$$-KL(q(z)||p(z)) = \int q(z) \log p(z) dz - \int q(z) \log q(z) dz \tag{9.4}$$

We may compute the entropy term (the second term in Equation (9.4)) as follows;

$$\mathcal{H}_q = -\int q(z) \log q(z) dz = \frac{1}{2} \log \sigma^2 + \mu + \frac{1}{2} + \frac{1}{2} \log(2\pi). \tag{9.5}$$

The cross entropy term (first term in Equation (9.4)) is;

$$\begin{aligned} \mathcal{CE}_{qp} &= \int q(z) \left(\alpha \log \beta - \log \Gamma(\alpha) - (\alpha + 1) \log z) - \frac{\beta}{z} \right) dz \\ &= \alpha \log \beta - \log \Gamma(\alpha) - (\alpha + 1) \, \mathbb{E}_{q(z)}[\log z] - \beta \, \mathbb{E}_{q(z)}[z^{-1}]. \end{aligned} \tag{9.6}$$

Note that $\mathbb{E}_{q(z)}[\log z] = \mu$ because the natural logarithm of a random variable distributed according to a a log-normal distribution $\mathcal{LN}(\mu, \sigma^2)$ is a normally distributed random variable $\mathcal{N}(\mu, \sigma^2)$.

Furthermore, $\mathbb{E}_{q(z)}[z^{-1}] = \exp(-\mu + \frac{\sigma^2}{2})$ because when $x \sim \mathcal{LN}(\mu, \sigma^2)$ then $\frac{1}{x} \sim \mathcal{LN}(-\mu, \sigma^2)$. We thus can compute the cross-entropy term precisely;

$$\mathcal{CE}_{qp} = \alpha \log \beta - \log \Gamma(\alpha) - (\alpha + 1)\mu - \beta \exp(-\mu + \frac{\sigma^2}{2}). \tag{9.7}$$

Hence we conclude the KL-divergence;

$$\begin{aligned} -KL(q(z)||p(z)) =& \alpha \log \beta - \log \Gamma(\alpha) - \alpha\mu - \beta \exp(-\mu + 0.5\sigma^2) \\ &+ 0.5(\log \sigma^2 + 1 + \log(2\pi)). \end{aligned} \tag{9.8}$$

Next let $p(z)$ be a Gamma prior $p(z) = \mathcal{G}(\alpha, \beta)$. We can update the cross-entropy according to our change to be;

$$\mathcal{CE}_{qp} = \int q(z)\Big(-\alpha \log \beta - \log \Gamma(\alpha) - \frac{z}{\beta} + (\alpha - 1)\log z \Big) dz \tag{9.9}$$

$$= -\alpha \log \beta - \log \Gamma(\alpha) - \beta^{-1}\, \mathbb{E}_{q(z)}[z] + (\alpha - 1)\, \mathbb{E}_{q(z)}[\log z] \tag{9.10}$$

$$= -\alpha \log \beta - \log \Gamma(\alpha) - \beta^{-1} \exp(\mu + \frac{\sigma^2}{2}) + (\alpha - 1)\mu. \tag{9.11}$$

Again we conclude the corresponding KL-divergence;

$$\begin{aligned} KL(q(z)||p(z)) = \alpha \log \beta + \log \Gamma(\alpha) - \alpha\mu \\ + \beta^{-1} \exp(\mu + 0.5\sigma^2) \\ - 0.5(\log \sigma^2 + 1 + \log(2\pi)). \end{aligned} \tag{9.12}$$

Summarizing the previously made efforts we can compute the KL-divergence from $p(s_a, s_b, \tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}})$ to $q_\phi(s_a, s_b, \tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}})$ by;

$$\begin{aligned} -KL(q_\phi(s_a)||p(s_a)) = \log \tau_0 + \tau_0^{-1} \exp\big(\mu_{s_a} + \frac{1}{2}\sigma_{s_a}^2\big) \\ + \frac{1}{2}\big(\mu_{s_a} + \log \sigma_{s_a}^2 + 1 + \log 2\big) \end{aligned} \tag{9.13}$$

$$\begin{aligned} -KL(q_\phi(s_b)||p(s_b)) = -\exp\big(\frac{1}{2}\sigma_{s_b}^2 - \mu_{s_b}\big) \\ + \frac{1}{2}\big(-\mu_{s_b} + \log \sigma_{s_b}^2 + 1 + \log 2\big) \end{aligned} \tag{9.14}$$

$$\begin{aligned} -KL(q_\phi(\tilde{\boldsymbol{\alpha}})||p(\tilde{\boldsymbol{\alpha}})) = \sum_i^A \Big(\exp\big(\mu_{\tilde{\alpha}_i} + \frac{1}{2}\sigma_{\tilde{\alpha}_i}^2\big) \\ + \frac{1}{2}\big(\mu_{\tilde{\alpha}_i} + \log \sigma_{\tilde{\alpha}_i}^2 + 1 + \log 2\big)\Big) \end{aligned} \tag{9.15}$$

$$\begin{aligned} -KL(q_\phi(\tilde{\boldsymbol{\beta}})||p(\tilde{\boldsymbol{\beta}})) = \sum_i^A \Big(-\exp\big(\frac{1}{2}\sigma_{\tilde{\beta}_i}^2 - \mu_{\tilde{\beta}_i}\big) \\ + \frac{1}{2}\big(-\mu_{\tilde{\beta}_i} + \log \sigma_{\tilde{\beta}_i}^2 + 1 + \log 2\big)\Big). \end{aligned} \tag{9.16}$$

Note that the KL-divergence corresponding to the weight distribution $q_\phi(\tilde{\mathbf{W}})$ is given by Equation (3.8).

F. Algorithms for the feedforward pass

We present the algorithms to compute a forward pass int Algorithms 9.3, 9.4, 9.5, and 9.6. Note that we use the local reparametrization trick for both fully connected and convolutional layers. The approximate posteriors correspond to the Bayesian compression (BC) scheme with group

Require: $\mathbf{H}, \mathbf{M}_w, \mathbf{\Sigma}_w$
1: $\hat{\mathbf{E}} \sim \mathcal{N}(0, 1)$
2: $\mathbf{Z} = \boldsymbol{\mu}_z + \boldsymbol{\sigma}_z \odot \hat{\mathbf{E}}$
3: $\hat{\mathbf{H}} = \mathbf{H} \odot \mathbf{Z}$
4: $\mathbf{M}_h = \hat{\mathbf{H}}\mathbf{M}_w$
5: $\mathbf{V}_h = \hat{\mathbf{H}}^2\mathbf{\Sigma}_w$
6: $\mathbf{E} \sim \mathcal{N}(0, 1)$
7: return $\mathbf{M}_h + \sqrt{\mathbf{V}_h} \odot \mathbf{E}$

Figure 9.3: Fully connected BC-GNJ layer h.

Require: $\mathbf{H}, \mathbf{M}_w, \mathbf{\Sigma}_w$
1: $\mathbf{M}_h = \mathbf{H} * \mathbf{M}_w$
2: $\mathbf{V}_h = \mathbf{H}^2 * \mathbf{\Sigma}_w$
3: $\hat{\mathbf{E}} \sim \mathcal{N}(0, 1)$
4: $\hat{\boldsymbol{\mu}}_z = \text{reshape}(\boldsymbol{\mu}_z, [K, 1, 1, N_f])$
5: $\hat{\boldsymbol{\sigma}}_z = \text{reshape}(\boldsymbol{\sigma}_z, [K, 1, 1, N_f])$
6: $\mathbf{Z} = \hat{\boldsymbol{\mu}}_z + \hat{\boldsymbol{\sigma}}_z \odot \hat{\mathbf{E}}$
7: $\mathbf{E} \sim \mathcal{N}(0, 1)$
8: return $\mathbf{M}_h \odot \mathbf{Z} + \sqrt{\mathbf{V}_h \odot \mathbf{Z}^2} \odot \mathbf{E}$

Figure 9.4: Convolutional BC-GNJ layer h.

normal-Jeffreys (BC-GNJ) and group Horseshoe (BC-GHS) priors. The groups are defined to be the weights corresponding to one input neuron for fully connected layers; and output feature maps for convolutional layers. $\mathbf{M}_w, \mathbf{\Sigma}_w$ represent the means and variances of a layer, and $\mathbf{H}$ denotes a minibatch of activations with corresponding size K. Naturally, in the first layers; $\mathbf{H} = \mathbf{X}$ with $\mathbf{X}$ being the minibatch of inputs. Note that, in the context of convolutional layers $*$ denotes the convolution operator, N_f the number of convolutional filters, and we assume the shape of a minibatch to be according to the [batch, height, width, feature maps] convention.

Require: $\mathbf{H}, \mathbf{M}_w, \mathbf{\Sigma}_w$
1: $\hat{\boldsymbol{\epsilon}} \sim \mathcal{N}(0,1)$
2: $\mu_s = .5\mu_{s_a} + .5\mu_{s_b}$
3: $\sigma_s = \sqrt{.25\sigma^2_{s_a} + .25\sigma^2_{s_b}}$
4: $\log \mathbf{s} = \mu_s + \sigma_s \odot \hat{\boldsymbol{\epsilon}}$
5: $\boldsymbol{\mu}_{\tilde{z}} = .5\boldsymbol{\mu}_{\tilde{\alpha}} + .5\boldsymbol{\mu}_{\tilde{\beta}} + \log \mathbf{s}$
6: $\boldsymbol{\sigma}_{\tilde{z}} = \sqrt{.25\boldsymbol{\sigma}^2_{\tilde{\alpha}} + .25\boldsymbol{\sigma}^2_{\tilde{\beta}}}$
7: $\hat{\mathbf{E}} \sim \mathcal{N}(0,1)$
8: $\mathbf{Z} = \exp(\boldsymbol{\mu}_{\tilde{z}} + \boldsymbol{\sigma}_{\tilde{z}} \odot \hat{\mathbf{E}})$
9: $\hat{\mathbf{H}} = \mathbf{H} \odot \mathbf{Z}$
10: $\mathbf{M}_h = \hat{\mathbf{H}}\mathbf{M}_w$
11: $\mathbf{V}_h = \hat{\mathbf{H}}^2\mathbf{\Sigma}_w$
12: $\mathbf{E} \sim \mathcal{N}(0,1)$
13: return $\mathbf{M}_h + \sqrt{\mathbf{V}_h} \odot \mathbf{E}$

Figure 9.5: Fully connected BC-GHS layer h.

9.2 Appendix Chapter 3

Review of state-of-the-art Neural Network Compression

We apply the compression scheme proposed by [68, 67] that highly optimizes the storage utilized by the weights. First of all, the authors store the weights in regular compressed sparse-row (CSR) format. Instead of storing $|W^{(l)}|$ parameters with a bit length of (commonly) $p_{\text{orig}} = 32$ bit, CSR format stores three vectors (A, IR, IC).

- **A** stores all non-zero entries. It is thus of size $|W^{(l)}|_{\neq 0} \times p_{\text{orig}}$, where $|W^{(l)}|_{\neq 0}$ is the number of non-zero entries in $W^{(l)}$.

- **IR** Is defined recursively: $\text{IR}_0 = 0$, $\text{IR}_k = \text{IR}_{k-1} +$ (number of non-zero entries in the $(k-1)$-th row of $W^{(l)}$). It got $K+1$ entries each of size p_{orig}.

- **IC** contains the column index in $W^{(l)}$ of each element of A. The size is hence, $|W^{(l)}|_{\neq 0} \times p_{\text{orig}}$.

Require: $\mathbf{H}, \mathbf{M}_w, \mathbf{\Sigma}_w$
1: $\mathbf{M}_h = \mathbf{H} * \mathbf{M}_w$
2: $\mathbf{V}_h = \mathbf{H}^2 * \mathbf{\Sigma}_w$
3: $\hat{\boldsymbol{\epsilon}} \sim \mathcal{N}(0, 1)$
4: $\mu_s = .5\mu_{s_a} + .5\mu_{s_b}$
5: $\sigma_s = \sqrt{.25\sigma_{s_a}^2 + .25\sigma_{s_b}^2}$
6: $\log \mathbf{s} = \text{reshape}(\mu_s + \sigma_s \odot \hat{\boldsymbol{\epsilon}}, [K, 1, 1, 1])$
7: $\boldsymbol{\mu}_{\tilde{z}} = \text{reshape}(.5\boldsymbol{\mu}_{\tilde{\alpha}} + .5\boldsymbol{\mu}_{\tilde{\beta}}, [K, 1, 1, N_f])$
8: $\boldsymbol{\sigma}_{\tilde{z}} = \text{reshape}(\sqrt{.25\boldsymbol{\sigma}_{\tilde{\alpha}}^2 + .25\boldsymbol{\sigma}_{\tilde{\beta}}^2}, [K, 1, 1, N_f])$
9: $\hat{\mathbf{E}} \sim \mathcal{N}(0, 1)$
10: $\mathbf{Z} = \exp(\boldsymbol{\mu}_{\tilde{z}} + \log \mathbf{s} + \boldsymbol{\sigma}_{\tilde{z}} \odot \hat{\mathbf{E}})$
11: $\mathbf{E} \sim \mathcal{N}(0, 1)$
12: return $\mathbf{M}_h \odot \mathbf{Z} + \sqrt{\mathbf{V}_h \odot \mathbf{Z}^2} \odot \mathbf{E}$

Figure 9.6: Convolutional BC-GHS layer h.

An example shall illustrate the format, let

$$W^{(l)} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

than

$$\begin{aligned} \text{A} &= [1, 2, 2, 5, 1] \\ \text{IR} &= [0, 1, 2, 2, 4, 5] \\ \text{IC} &= [3, 1, 0, 1, 3] \end{aligned}$$

The compression rate achieved by applying the CSC format naively is

$$r_p = \frac{|W^{(l)}|}{2|W^{(l)}|_{\neq 0} + (K + 1)} \tag{9.17}$$

However, this result can be significantly improved by optimizing each of the three arrays.

Storing the index array IR

To optimize IR, note that the biggest number in IR is $|W^{(l)}|_{\neq 0}$. This number will be much smaller than $2^{p_{\text{orig}}}$. Thus one could try to find

$p \in \mathbf{Z}_+$ such that $|W^{(l)}|_{\neq 0} < 2^{p_{\text{prun}}}$. A codebook would not be necessary. Thus instead of storing $(K+1)$ values with p_{orig}, we store them with p_{prun} depth.

Storing the index array IC

Instead of storing the indexes, we store the differences between indexes. Thus there is a smaller range of values being used. We further shrink the range of utilized values by filling A with zeros whenever the distance between two non-zero weights extends the span of 2^p_{prun}. [67] propose p = 5 for fully connected layers and p = 8 for convolutional layers. An illustration of the process can is shown in Fig. 9.7. Furthermore, the indexes will be compressed Hoffman encoding.

Figure 9.7: Illustration of the process described in 9.2. IC is represented by relative indexes(diff). If the a relative index is larger than $8(= 2^3)$, A will be filled with an additional zero. Figure from [67].

Storing the weight array A

In order to minimize the storage occupied by A. We quantize the values of A. Storing indexes in A and a consecutive codebook. Indexing can be improved further by again applying Huffman encoding.

Scalability

Neural Networks are usually trained with a form of batch gradient decent (GD) algorithm. These methods fall into the umbrella of stochastic optimization [177]. Here the model parameters $\mathbf{W}$ are updated iteratively. At each iteration t, a set of B data instances is used to compute a noisy approximation of the posterior derivative with respect to $\mathbf{W}$ given all data instances N.

$$\nabla_{\mathbf{W}} \log p(\mathbf{W}|\mathcal{D}) = \frac{N}{B} \sum_{n=1}^{B} \nabla_{\mathbf{W}} \log p(\mathbf{t}_n|\mathbf{x}_n, \mathbf{w}) + \sum_{i=1}^{I} \nabla_{\mathbf{W}} \log p(w_i) \tag{9.18}$$

This gradient approximation can subsequently be used in various update schemes such as simple GD.

For large models estimating the prior gradient can be an expensive

operation. This is why we propose to apply similar measures for the gradient estimation of the prior as we did for the likelihood term. To do so, we sample K weights randomly. The noisy approximation of the posterior derivative is now:

$$\nabla_{\mathbf{W}} \log p(\mathbf{W}|\mathcal{D}) = \frac{N}{B} \sum_{n=1}^{B} \nabla_{\mathbf{W}} \log p(\mathbf{t}_n|\mathbf{x}_n, \mathbf{w}) + \frac{I}{K} \sum_{i=1}^{K} \nabla_{\mathbf{W}} \log p(w_i) \tag{9.19}$$

Filter Visualisation

In Figure 9.8 we show the pre-trained and compressed filters for the first and second layers of LeNet-5-Caffe. For some of the feature maps from layer 2 seem to be redundant hence the almost empty columns. In Figure 9.9 we show the pre-trained and compressed filters for the first and second layers of LeNet-300-100.

Configuring the Hyper-priors

Gamma Distribution

The Gamma distribution is the conjugate prior for the precision of a univariate Gaussian distribution. It is defined for positive random variables $\lambda > 0$.

$$\Gamma(\lambda|\alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \tag{9.20}$$

For our purposes it is best characterised by its mode $\lambda^* = \frac{\alpha - 1}{\beta}$ and its variance $\text{var}_\gamma = \frac{\alpha}{\beta^2}$. In our experiments we set the desired variance of the mixture components to 0.05. This corresponds to $\lambda^* = 1/(0.05)^2 = 400$. We show the effect of different choices for the variance of the Gamma distribution in Figure 9.10.

Beta Distribution

The Beta distribution is the conjugate prior for the Bernoulli distribution, thus is often used to represent the probability for a binary event.

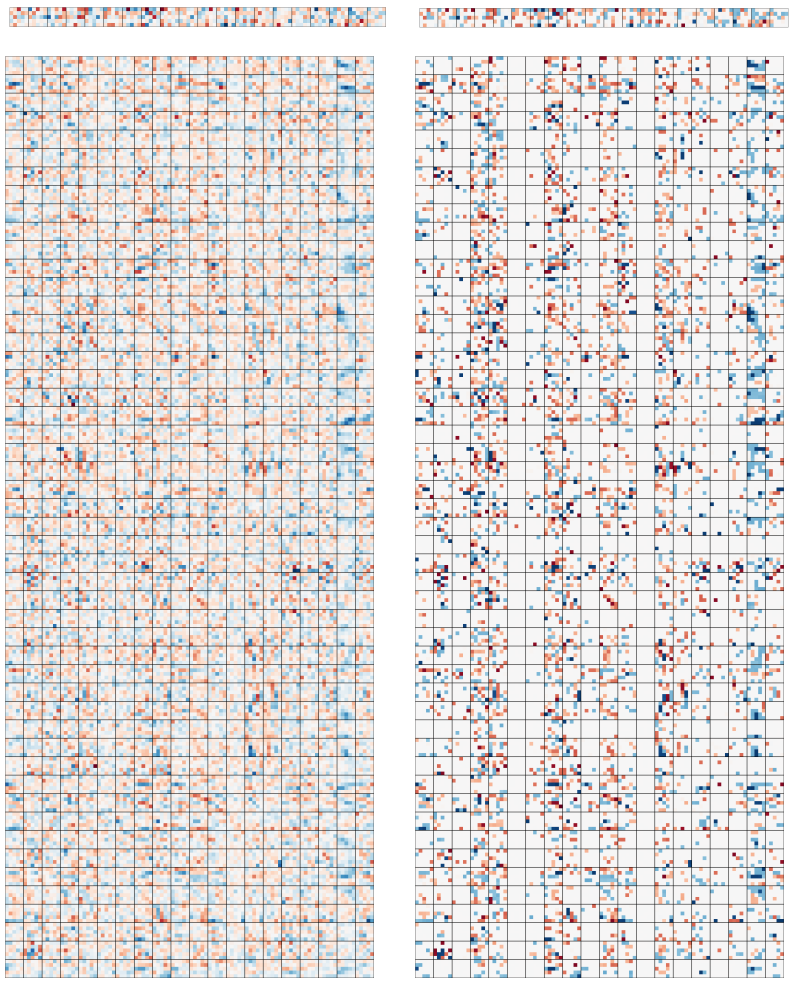

Figure 9.8: Convolution filters from LeNet-5-Caffe. **Left:** Pre-trained filters. **Right:** Compressed filters. The top filters are the 20 first layer convolution weights; the bottom filters are the 20 by 50 convolution weights of the second layer.

It is defined for some random variable $\pi_{j=0} \in [0, 1]$

$$\mathcal{B}(\pi_{j=0}|\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}(\pi_{j=0})^{\alpha-1}(1-\pi_{j=0})^{\beta-1} \tag{9.21}$$

with $\alpha, \beta > 0$. α and β can be interpreted as the effective number of observations prior to an experiment, of $\pi_{j=0} = 1$ and $\pi_{j=0} = 0$, respectively. In the literature, $\alpha + \beta$ is defined as the pseudo-count. The higher the pseudo-count the stronger the prior. In Figure 9.11, we show the Beta distribution at constant mode $\pi^*_{j=0} = \dfrac{\alpha - 1}{\alpha + \beta - 2} = 0.9$. Note, that, the beta distribution is a special case of the Dirichlet distribution in a different problem setting it might be better to rely on this distribution to control all π_j.

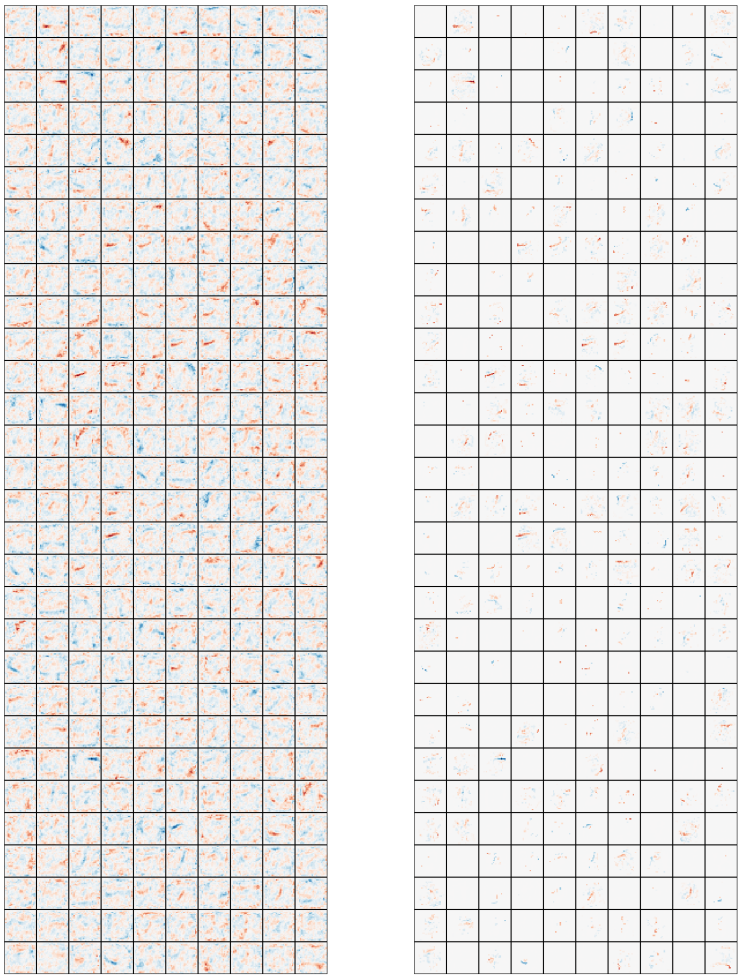

Figure 9.9: Feature filters for LeNet-300-100. **Left:** Pre-trained filters. **Right:** Compressed filters.

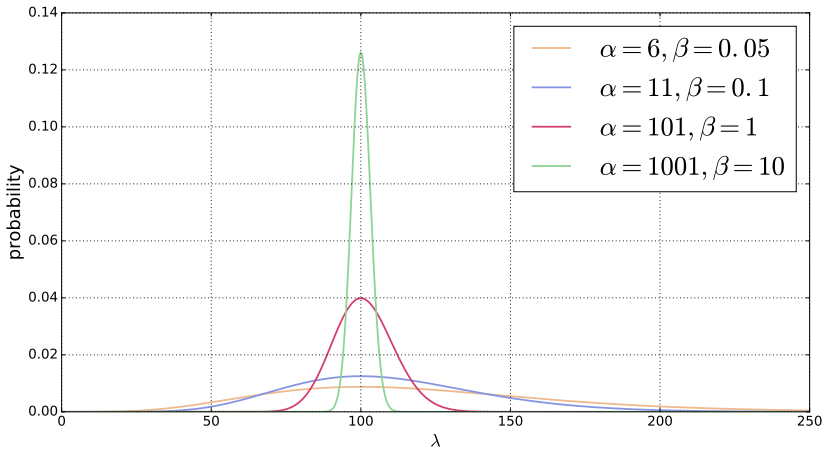

Figure 9.10: Gamma distribution with $\lambda^* = 100$. α and β correspond to different choices for the variance of the distribution.

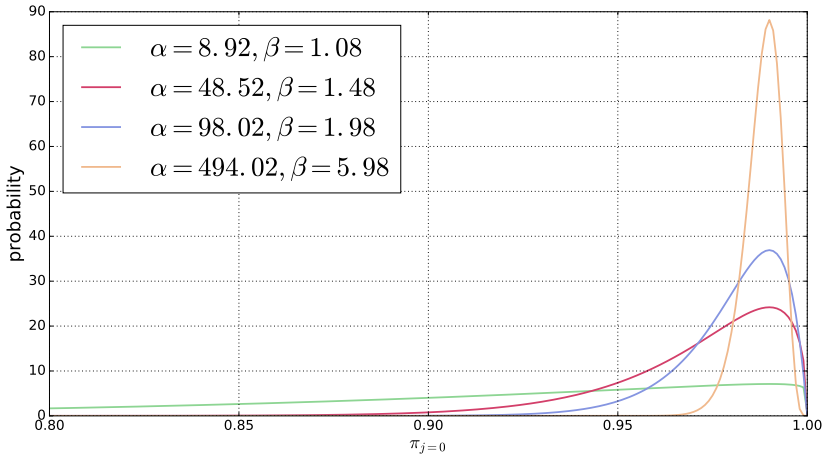

Figure 9.11: Beta distribution with $\pi^*_{j=0} = 0.9$. α and β correspond to different choices for the pseudo-count.

9.3 Appendix Chapter 4

Visual impressions

First, to give a visual impression of the observations, we learn from we present in Figure 9.16 samples from the 3 data sets we use. All of the datasets are based on the same protein estimate of the GroEL-GroES protein[223]. They differ in the signal-to-noise ratio (SNR). The left row represents noise free data, the middle a moderate common noise level, and the right an extreme level of noise. For each observation example, we show also the corresponding Fourier transformations, their real part in the second column and their imaginary part in the third column. Further we present the qualitative results of fitting the mid

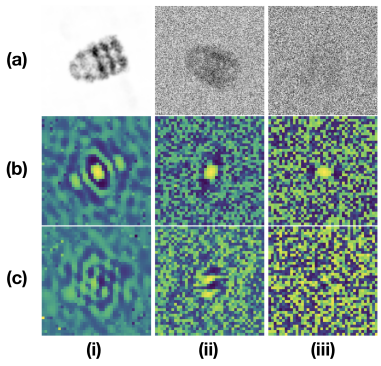

Figure 9.12: *Left to Right:* We show samples from the dataset we use: (i) no noise (such as in Experiment 6.1), (ii) moderate noise and (iii) high noise (such as in experiment 6.2). *Top to Bottom:* Observation in (a) real space, first 20 Fourier shells (b) real part and (c) imaginary part (for better visibility log-scaled). The latter two are being used for the optimization due to the application of the Fourier slice theorem explained in Section 5.4.

and high level datasets with our method. We visualize the respective protein fit from experiment 6.2 in Figure 9.13 with the Chimera X software package [160]. The two pictures on the top row represent the middle noise fit, respectively the bottom two the high noise fit.

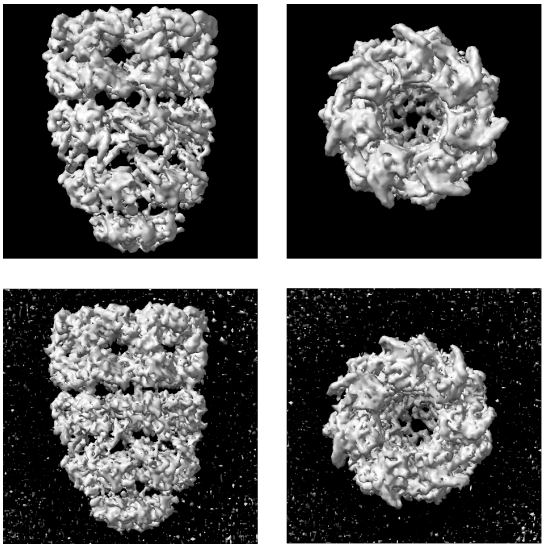

Figure 9.13: *Top:* Side and top view of the GroEL-GroES protein fit with moderate noise level data. *Bottom:* Side and top view of the respective high noise level dataset.

Extension to experiment 5.6

We shall execute the same experiment as in section 5.6 given the dataset with intermediate noise. We display the experiments of this result in Figure 9.14. It is clear that while, missing information leads to large deviation in variance we also find that the noise leads to some variance in the observed area. Again we visualize the result of the fit in Figure 9.15.

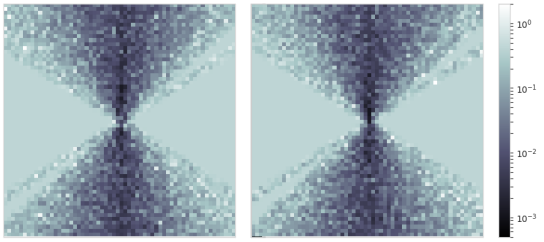

Figure 9.14: Center slice through the learned Fourier volume uncertainties $\sigma_{\mathbf{v}}$. *Left:* real part, *Right:* imaginary part. We learn the model fit with observations coming only from a 30° cone, a scenario similar to breast cancer scans where observations are available only from some viewing directions. Uncertainty close to 1 means that the model has no information in these areas, close to zero represents areas of high sampling density. In contrast to other models, our model can identify precisely where information is missing (high variance).

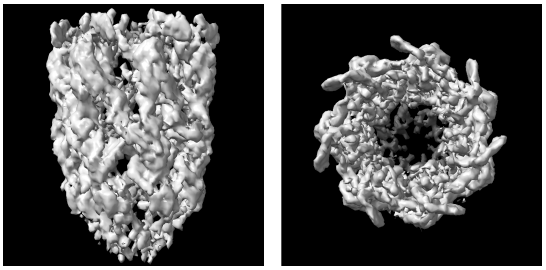

Figure 9.15: Side and top view of the GroEL-GroES protein fit with moderate noise level data and all observations stemming from a limited pose space.

Amortized inference and variational EM for pose estimation

We have not used amortized inference in our experiments. In experiment 6.4 we have modelled poses as local variables and trained them by variational expectation maximization. Other work shows that the amortization gab can be significant [186, 5, 138]. Hence in order to exclude the gap as a reason for failure, we decided to model local variables. We did run experiments though with ResNets as encoders without success either. We believe the core problem in probabilistic pose estimation is the number of local optima. This makes simple SGD a somewhat poor choice, because we rely on finding the global minimum.

Remarks to the chosen observation model

The Gaussian noise model is a common but surely oversimplified model [190]. A better model would be the Poisson distribution. The Gaussian is a good approximation to it given a high rate parameter meaning if there is a reasonable high count of radiation hitting the sensors. This is a good assumption for most methods of the field, but can actually be a poor model in some cases of cryo electron microscopy. An example of an elaborate model is presented in Vulović et al. [219].

9.4 Appendix Chapter 5

Model samples

Message reconstructions for the bandwidth-limited channel can be seen in Figure 9.16.

Rate-distortion perspective

Originally, information theory would study how a message can be communicated over a noisy channel to a receiver without errors. It is often a more realistic scenario, though, to think of the receiver to

Figure 9.16: *First row:* Message samples from the source distribution. *Other rows from top to bottom:* Samples of the reconstructed message at all considered bandwidths. The top row has least information.

tolerate a certain amount of distortion. Intuitively, the more we allow for distortion of a message the smaller the number of bits we need to communicate. Rate-distortion theory is a major field of information theory that studies how these modifications to the original set-up effect fundamental theorems such as data compression or transmission. The limitations of the classical view become clear when considering continuous random variables. Continuous random variables require infinite precision to represent exactly. Hence it is not possible to send finite rate codes. Assume X to be the continuous random variable to be represented by $X'(X)$. Say we are given R bits to send X. $X'(X)$ than can take 2^R values. The goal of rate-distortion coding is to distribute these 2^R codepoints such that a minimal distortion, measured by a distortion function d,

$$d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+, \tag{9.22}$$

where $\mathcal{X}$ is the source alphabet, is being achieved.

Source Coding in a rate-distortion sense

Source coding in the context of rate-distortion theory entails two steps: quantization $X'(X)$ and traditional source encoding $Y(X') = Y(X)$. In both steps the goal is to keep the loss of information minimal given a rate that shall be achieved, $I(X;Y) \leq R$ where

$$I(X;Y) = \mathbb{E}_{P(X,Y)}[\log P(X,Y) - \log P(X)P(Y)], \tag{9.23}$$

is the mutual information. The goal is to keep this bound tight. However, computing the mutual information is hard since we do not have

access to the true data density. Following Alemi et al. [5], we instead find a variational approximation,

$$H - D \leq I(X;Y) \leq R \tag{9.24}$$

with

$$D := \mathbb{E}_{P(X)}[\mathbb{E}_{E^S(Y'|X)}[\log D^S(\hat{X}|Y')]] \tag{9.25}$$
$$R := \mathbb{E}_{P(X)}[\mathbb{E}_{E^S(Y'|X)}[\log E^S(Y'|X) - \log M(Y)]], \tag{9.26}$$

where they introduce $M(Y)$ as the variational approximation to $P(Y) = \mathbb{E}_{P(X)}[E^S(Y'|X)]$. This reestablishes that the data entropy bounds feasible (compression rate R, distortion D) pairs: $H(X) \leq R + D$. This ties together with the result from optimal coding that the source entropy bounds the optimal code length. Hinton and Van Camp [80] show that via bits-back coding this code length (rate) can actually be achieved. This argument has further been hardened by Townsend et al. [207] who design an actual compression algorithm in this manner.

Joint Source-Channel Coding in a rate-distortion sense

In the previous section we discussed how relaxing the requirement to sending a message exactly, to sending a message under a certain distortion, effects source coding. The optimal code length would thus be $R(D)$ bits/symbol rather than $H(X) \geq R(D)$ in the error-free scenario. We can connect this information to Shannon's channel coding theorem. We know that the channel capacity C restricts the number of bits that can be send. Thus there exists a solution for a maximum distortion communication system only if $R(D) < C$. When in the previous section R helps to describe the number of bits that represent a random variable X, we can similarly find a variational approximation to $I(X, Z)$ the amount of bits representing X after passing the channel,

$$T := \mathbb{E}_{P(X)E(Y|X)}\left[\mathbb{E}_{C(Z|Y)}[\log C(Z|Y) - \log N(Z)]\right], \tag{9.27}$$

where equivalent to the discussion in the previous section $N(Z)$ is the variational approximation to $P(Z) = \mathbb{E}_{P(X)E(Y|X)}[C(Z|Y)]$. We shall refer to T as the transmission rate.

Relaxing the Binary Channel

Relaxing the Bernoulli: Binary Concrete Distribution

Learning of systems with stochastic nodes $P_\theta(X)$ in Machine Learning is often synonymous with optimizing an objective function $\mathcal{L}(\theta,\phi) = E_{X\sim P_\theta(X)}[f_\theta(X)]$ w.r.t the parameters θ,ϕ via some gradient descent based scheme. The challenge lies in computing the parameters θ that belong to the stochastic node. A popular approach to this problem is the application of the so called reparameterization trick [106, 171] in which a stochastic node $P_\theta(X)$ with parameter dependency θ is turned into a stochastic node $Q(Z)$ without parameter dependentcy and a determined function $g_\theta(\cdot)$.

$$\mathcal{L}(\theta,\phi) = E_{X\sim P_\theta(X)}[f_\theta(X)] = E_{Z\sim Q(Z)}[f_\theta(g_\theta(Z))] \tag{9.28}$$

with $x = g_\theta(x)$. The remodelled stochastic node now allows for gradient based stochastic optimization via Monte Carlo sampling[3],

$$\nabla_\theta \mathcal{L}(\theta,\phi) = E_{Z\sim Q(Z)}[f'_\theta(g_\theta(Z))\nabla_\theta g_\theta(Z)]. \tag{9.29}$$

[3] If there is no possibility of reparametization, one can retain to the score-function estimator, also known as REINFORCE or likelihood-ratio estimator, which allows to compute the gradient via Monte Carlo sampling. This however leads to higher variance gradients.

For example, consider sampling from the Gaussian distribution, $X \sim \mathcal{N}(X|\mu,\sigma)$ can be replaced by sampling from a standard Gaussian $Z \sim \mathcal{N}(Z|0,1)$ and applying $x = g_{\{\mu,\sigma\}}(z) = \mu + \sigma z$. Reparameterizing a discrete distribution such as the Bernoulli $\mathcal{B}(p)$ is not as straight forward. Maddison et al. [136] propose reparameterization with Gumbel-Max trick. Specifically, the reparameterization is based on a logistic random variable $L \sim Logistic(L)$ as parameter-free stochastic node, the relaxed Bernoulli sample can than be attained by

$$\begin{aligned} Y &= (L + \log(\alpha)/T) \\ X &= \sigma(Y) \end{aligned} \tag{9.30}$$

where α corresponds to the location parameter p in the Bernoulli distribution, T the temperature adjusts the amount of relaxiation and σ is the sigmoid function. The density corresponding to this sampling procedure is given by,

$$p_{\alpha,T}(x) = \frac{T\alpha x^{-T-1}(1-x)^{-T-1}}{(\alpha x^{-T}(1-x)^{-T})^2}. \tag{9.31}$$

We shall call this the Binary Concrete distribution or relaxed Bernoulli distribution $\mathcal{B}_T(\alpha)$. It has several desirable properties.

1. $P(X > 0.5) = \frac{\alpha}{1+\alpha}$
2. $P(\lim_{T\to 0} X = 1) = \frac{\alpha}{1+\alpha}$

3. If $T \leq 1$ than $p_{\alpha,T}(x)$ is log-convex in x.

One problem, however, we may often be faced with is computing the log-likelihood of such a stochastic node. For example when computing the KL-divergence of a variational auto-encoder. Due to the saturation of the sigmoid function computing the log-likelihood empirically may lead to underflow issues. This is why it has been proposed to compute the log-likelihood based on the samples Y before applying the sigmoid function since this is an inevitable function. The corresponding log-likelihood is given by,

$$\begin{aligned} \log g_{\alpha,T}(y) = \log T - Ty + \log \alpha \\ - 2\log(1 + \exp(-Ty + \log \alpha)). \end{aligned} \tag{9.32}$$

As an alternative we may clip the log-likelihood. This variant is easier to apply when there is no direct access to the stochastic node, we shall see what this means precisely in the next section.

Binary Channel

The symmetric binary channel is a discrete channel with an input and output alphabet of size 2, $\mathcal{Y} \in \{0,1\}^D$, $\mathcal{Z} \in \{0,1\}^D$. The channel can be realized with Bernoulli noise on each input pixel,

$$Z_i = \frac{(2W_i - 1)}{2 \cdot (2Y_i - 1)} + \frac{1}{2} \qquad W_i \sim P_W(W_i) = \mathcal{B}(W_i|p) \tag{9.33}$$

where $\mathcal{B}(p)$ is a Bernoulli with p the likelihood of keeping an input bit. The channel is called a symmetric channel because the probability of changing a bit does not depend on its state.

Following, we relax the channel as defined above to allow for training of differntiable communication models. For this we will utilize the relaxed Bernoulli distribution as described in the previous section. As for the Gaussian Channel we assume that that Y_i can be constructed from a learned Bernoulli itself with $Y_i \sim \mathcal{B}_{T_{Y_i}}(Y_i|\alpha_{Y_i})$. In order to compute (6.8), we need to evaluate the channel density given its input $C(Z|Y)$. Since Z depends deterministically on W_i and Y_i, the channel density equals the noise density with transformed input argument $C(Z|Y) = P_W(((2Z-1)(2Y-1))/2 + 1)$. For Bernoulli noise, such as in the original channel formulation, the system could thus not learn. We thus propose to also adapt the noise to be a relaxed Bernoulli with probability density as in Eq. (19).

Finally, we may restrict our channel to a specific SNR. This can be

computed to be

$$s = \frac{2pp_{Y_i} + 0.5 - p - p_{Y_i}}{-2p_{Y_i}p - 0.5 - p - p_{Y_i}}. \tag{9.34}$$

To train neural models, we will use the relaxed Bernoulli for both Y and W, however the SNR is computed assuming both as Bernoulli distributions. This assumption is only correct for $T \to 0$. Initial experimental results, with the relaxed binary channel have been showing that optimization with this parameterisation is somewhat challenging. Our finding supports a finding in [30] that focus on VIMCO instead of the re-parameterisation trick.

Sensitivity of the communication systems to the hyper-parameter β

In optimizing communication systems, β is perhaps the most important hyper parameter. This is why we present the complete set of results for experiment one in Figure 9.17. For the source VAE β trades compression rate vs distortion. At maximum compression, the channel source distribution would be emulated perfectly and thus the channel AE input distribution. However, this scenario would also eliminate the mutual information between X and Y. Thus a balance must be found by tuning β.

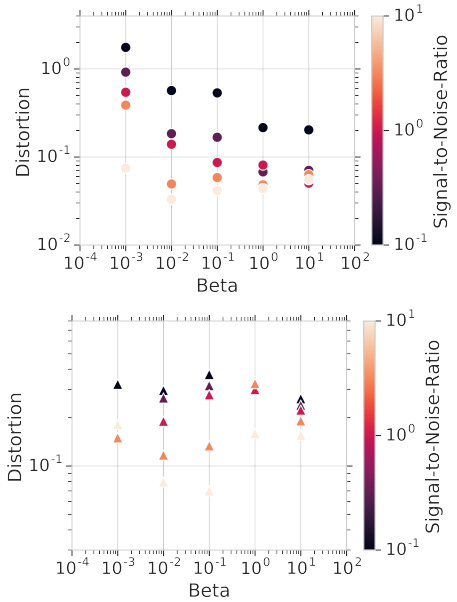

Figure 9.17: The results in Figure 6.4 show the distortion vs the SNR for an optimized β. Here we present all results. Note that, the joint model is more sensitive to changes in β in the range we have chosen. Top: Joint model Bottom: Seperate Model