Minimal Random Code Learning with Mean-KL Parameterization

Jihao Andreas Lin [1] **Gergely Flamich** [1] **José Miguel Hernández-Lobato** [1]

Abstract

This paper studies the qualitative behavior and robustness of two variants of *Minimal Random Code Learning* (MIRACLE) used to compress variational Bayesian neural networks. MIRACLE implements a powerful, *conditionally Gaussian* variational approximation for the weight posterior $Q_{\mathbf{w}}$ and uses relative entropy coding to compress a weight sample from the posterior using a Gaussian coding distribution $P_{\mathbf{w}}$. To achieve the desired compression rate, $D_{\mathrm{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}]$ must be constrained, which requires a computationally expensive annealing procedure under the conventional mean-variance (Mean-Var) parameterization for $Q_{\mathbf{w}}$. Instead, we parameterize $Q_{\mathbf{w}}$ by its mean and KL divergence from $P_{\mathbf{w}}$ to constrain the compression cost to the desired value by construction. We demonstrate that variational training with Mean-KL parameterization converges twice as fast and maintains predictive performance after compression. Furthermore, we show that Mean-KL leads to more meaningful variational distributions with heavier tails and compressed weight samples which are more robust to pruning.

1. Introduction

With the ever-growing size of neural network architectures, such as large language models (e.g. BERT, Kenton & Toutanova, 2019), it is now a key challenge to ensure their memory and energy efficiency. While there is a large literature on model compression, almost all works rely on some form of quantization scheme. In this paper, we consider an alternative method to quantization, namely Minimal Random Code Learning (MIRACLE, Havasi et al., 2019), which has recently demonstrated state-of-the-art performance for neural network compression. The MIRACLE framework employs a powerful, conditionally Gaussian variational distribution $Q_{\mathbf{w}}$ over the weights $\mathbf{w}$ of a neural network and uses relative entropy coding (REC, Flamich et al., 2020) with a Gaussian coding distribution $P_{\mathbf{w}}$ to encode a random weight sample from $Q_{\mathbf{w}}$. The average coding cost of encoding a weight sample is $D_{\mathrm{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}]$, which needs to be carefully controlled in a practical compression scheme. To this end, we propose to use Mean-KL parameterization for Gaussians (Flamich et al., 2022) to parameterize $Q_{\mathbf{w}}$, allowing explicit control over $D_{\mathrm{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}]$ by construction. We demonstrate that Mean-KL leads to many practical benefits over the conventional mean-variance (Mean-Var) parameterization used by Havasi et al. 2019, which requires a computationally expensive annealing procedure to control the coding cost. In particular, we show that, compared to Mean-Var parameterization, variational training converges in half the number of iterations using Mean-KL parameterization while maintaining predictive performance after compression. Furthermore, we illustrate that the resulting variational distribution exhibits more meaningful shapes with heavy tails, which makes the compressed weight sample more robust against zero pruning.

2. Background

Minimal Random Code Learning Havasi et al. 2019 consider a setting akin to the β-VAE (Higgins et al., 2017) to encode neural network weights with a limited information budget C. To this end, let $\mathcal{X}, \mathcal{Y}$ and $\mathcal{W}$ be the input, output and weight spaces, respectively, let $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ be a dataset and let $h : \mathcal{X} \times \mathcal{W} \to \mathcal{Y}$ be a neural network with input $\mathbf{x}$ and weights $\mathbf{w}$. To control the information content of the weights, let $P_{\mathbf{w}}$ be the *coding distribution* and $Q_{\mathbf{w}}$ be the *variational distribution* over $\mathbf{w}$. In this setting, Hinton & Van Camp 1993 show that the information content of the weights is $D_{\mathrm{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}]$. Further, let $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ be a *distortion function*. MIRACLE minimizes

$$\mathbb{E}_{\mathbf{w} \sim Q_{\mathbf{w}}} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \Delta(\mathbf{y}, h(\mathbf{x}, \mathbf{w})) + \beta D_{\mathrm{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}] \quad (1)$$

with respect to $Q_{\mathbf{w}}$ to minimize distortion within the given information budget of $D_{\mathrm{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}] = C$ nats. During optimization, β is dynamically adapted to anneal the KL divergence, such that the constraint is eventually satisfied.

In this paper, we encode the samples using minimal ran-

[1]University of Cambridge. Correspondence to: Jihao Andreas Lin <jal232@cam.ac.uk>.

Accepted at *ICML Neural Compression Workshop 2023*, Honolulu, Hawaii, USA.

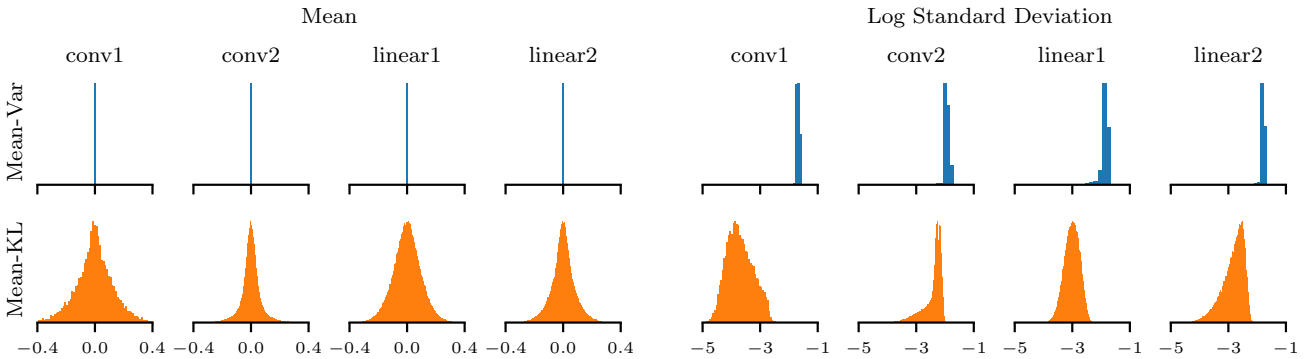

Figure 1. Layerwise histograms of variational mean and log standard deviation for Mean-Var (blue) versus Mean-KL (orange) parameterizations. Mean-Var struggles to learn meaningful distributions: means are concentrated at zero and standard deviations are clustered at high values. Mean-KL learns more reasonable distributions with heavier tails and a broader range of values.

dom coding (MRC, Havasi et al., 2019) for simplicity, though more sophisticated approaches, such as A* coding (Flamich et al., 2022) or greedy Poisson rejection sampling (Flamich, 2023), have been invented. Given a suitable $Q_{\mathbf{w}}$, a random sample from $Q_{\mathbf{w}}$ is compressed by first drawing $K = \exp(D_{\mathrm{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}])$ samples from $P_{\mathbf{w}}$. These K samples are then used to construct a discrete distribution whose probability mass function is defined by the importance weights $r_k = \frac{dQ_{\mathbf{w}}}{dP_{\mathbf{w}}}(\mathbf{w}_k)$, where $\frac{dQ_{\mathbf{w}}}{dP_{\mathbf{w}}}$ is the Radon-Nikodym derivative, i.e. the density ratio, of $Q_{\mathbf{w}}$ with respect to $P_{\mathbf{w}}$. The compressed weight sample is represented by an index $k_* \sim Q_k$. Since $0 \leq k_* < K$, it is always possible to encode k_* using $D_{\mathrm{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}] = C$ nats. The weight sample can be decoded by drawing the ${k_*}^{\text{th}}$ sample from $P_{\mathbf{w}}$ using a shared random number generator with a shared random seed. Due to the exponential scaling, simulating K samples is intractable if $\mathbf{w}$ has many dimensions. Havasi et al. 2019 solve this issue by partitioning $\mathbf{w}$ dimensionwise into smaller blocks with local information budgets C_{block}, such that K is feasible.

Refining Mean-Field Posteriors An important choice in practice is the variational family over which we optimize Equation (1). Since we are interested in studying the behavior of samples using MIRACLE, we also adopt the variational family suggested by Havasi et al. (2019). Concretely, assume that we have already partitioned the weight vector as $\mathbf{w} = w_{1:B} = \mathbf{w}_1 \oplus \mathbf{w}_2 \oplus \ldots \oplus \mathbf{w}_B$, where B denotes the number of blocks, and $\oplus$ denotes vector concatenation. To begin, we use a mean-field Gaussian variational approximation, i.e. we parameterize the means $\mu_{1:B} = \mu_1 \oplus \ldots \oplus \mu_B$ and marginal variances $\sigma^2_{1:B} = \sigma^2_1 \oplus \ldots \oplus \sigma^2_B$ (Mean-Var). Once variational training converges, we compress the first block $\mathbf{w}_1$, resulting in a sample $\tilde{\mathbf{w}}_1$. Keeping $\tilde{\mathbf{w}}_1$ fixed, we resume optimization to *fine-tune* the remaining means $\mu_{2:B}$ and variances $\sigma^2_{2:B}$. We repeat this process B times in total, where at step b, $\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_{b-1}$ are fixed, means $\mu_{b:B}$ and variances $\sigma^2_{b:B}$ are optimized, and a random sample from block b is encoded. Note that the variational posterior $Q_{\mathbf{w}_{b:B}|\tilde{\mathbf{w}}_{1:b-1}}$ at step b is only *factorized conditionally* on the weight samples in the first $b-1$ blocks, which results in a much better variational approximation.

Mean-KL Parameterization for Gaussians Flamich et al. 2022 show that, given a univariate Gaussian coding distribution $P_w = \mathcal{N}(w|\nu, \rho^2)$ with mean ν and variance ρ^2, a variational distribution $Q_w = \mathcal{N}(w|\mu, \sigma^2)$ can be uniquely parameterized by mean μ and $D_{\mathrm{KL}}[Q_w \| P_w] = \kappa$ if

$$|\mu - \nu| < \rho\sqrt{2\kappa} \tag{2}$$

is satisfied. The variance σ^2 of Q_w can be recovered via

$$\sigma^2 = -\rho^2 W\left(-\exp(z^2 - 2\kappa - 1)\right), \tag{3}$$

where $z = (\mu - \nu)/\rho$ and W is the principal branch of the Lambert W function (Corless et al., 1996), defined by the relation $W(x)e^{W(x)} = x$ (see Appendix B for details).

3. Mean-KL Parameterization for MIRACLE

Recognizing that the main goal of minimizing Equation (1) combined with KL annealing is to solve

$$\arg\min_{Q_{\mathbf{w}}} \quad \mathbb{E}_{\mathbf{w}\sim Q_{\mathbf{w}}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \Delta(\mathbf{y}, h(\mathbf{x}, \mathbf{w})), \tag{4}$$

$$\text{subject to} \quad D_{\mathrm{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}] = C, \tag{5}$$

we propose to use Mean-KL parameterization (Flamich et al., 2022) to enforce the $D_{\mathrm{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}] = C$ constraint mathematically instead of performing computationally expensive KL annealing. To this end, the total information budget $C = \kappa$ must be distributed to each weight, resulting in local information budgets κ_w. Thus, in Mean-KL parameterization, each weight has a mean parameter μ_w and a local information budget κ_w, matching the number of parameters for the conventional Mean-Var parameterization, albeit with one fewer degree of freedom because $\sum_{w\in\mathbf{w}} \kappa_w = \kappa$.

In practice, we introduce an *information quota* parameter γ_w per weight, which satisfies $\sum_{w\in\mathbf{w}} \gamma_w = 1$ and defines

the relative share of the total information budget assigned to w, that is $\kappa_w = \gamma_w \kappa$. The constraint on the information quota parameters is implemented using a softmax function. To ensure that $|\mu_w - \nu| < \rho\sqrt{2\kappa_w}$ (Equation (2)), we define

$$\mu_w = \nu + \rho\sqrt{2\kappa_w}\tanh(\tau_w), \tag{6}$$

as suggested by Flamich et al. (2022), leaving τ_w and γ_w as trainable parameters. In combination with blockwise partitioning of $\mathbf{w}$, each block has its own constraint and κ is simply replaced by κ_{block}. When drawing samples from $Q_{\mathbf{w}}$ or evaluating the density of $Q_{\mathbf{w}}$, we convert τ_w and γ_w to μ_w and ${\sigma_w}^2$ using Equation (6) and Equation (3), respectively, followed by the same computations as with conventional Mean-Var parameterization.

4. Experiments

We empirically demonstrate advantages of Mean-KL compared to conventional Mean-Var parameterization: We show that variational training with Mean-KL parameterization converges faster than Mean-Var while maintaining predictive performance, we illustrate that Mean-KL leads to more meaningful distributions with heavier tails, and we demonstrate that these more meaningful distributions translate to improved robustness when pruning weights to zero.

Training Dynamics and Predictive Performance We adopt the experimental setup of Havasi et al. 2019 and train a LeNet-5 on MNIST. The distortion function Δ is the cross-entropy, which is commonly used as a loss function in image classification. Matching Havasi et al. 2019, we used a local information budget of $C_{\text{block}} = \kappa_{\text{block}} = 20$ bits. We varied the block size between 20, 30, and 40. For both parameterizations, we used Adam with a learning rate of 0.001 and a mini-batch size of 200. For KL divergence annealing with Mean-Var, we used $\epsilon_{\beta_0} = 10^{-8}$ and $\epsilon_{\beta} = 5 \times 10^{-5}$, as suggested by Havasi et al. 2019. See Appendix C for further implementation details.

Figure 2 illustrates how Mean-Var spends most of the optimization on minimizing and annealing the KL divergence to the desired coding cost, whereas for Mean-KL, the whole optimization process focuses on minimizing cross entropy, given that the parameterization already constrains the KL divergence to the desired coding cost. Crucially, KL divergence annealing with Mean-Var takes a tremendous amount of time while minimizing cross entropy with Mean-KL converges in just *half* the number of iterations. Table 1 shows that Mean-KL maintains predictive performance comparable to Mean-Var across different compression ratios, being slightly better in the low compression ratio setting and slightly worse in the high compression ratio settings, albeit within standard error.

Table 1. MNIST classification error after compression (lower is better). Mean ± standard error over 10 seeds.

Block Size	Ratio	Mean-Var	Mean-KL
20	555x	0.82 ± 0.07 %	0.77 ± 0.05 %
30	833x	0.79 ± 0.05 %	0.87 ± 0.08 %
40	1111x	0.87 ± 0.07 %	0.96 ± 0.08 %
Optimizer Iterations		200,000	100,000

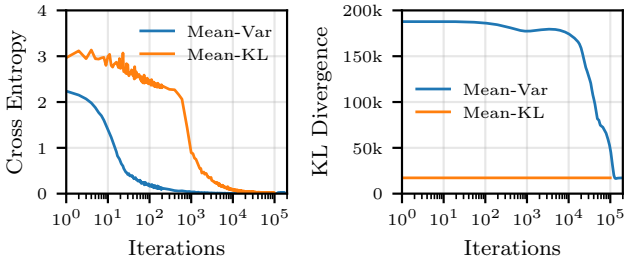

Figure 2. Training dynamics of Mean-Var and Mean-KL parameterizations. Mean-Var requires a large amount of iterations to anneal the KL divergence to the desired coding cost. Mean-KL constrains $D_{\text{KL}}[Q_{\mathbf{w}} \| P_{\mathbf{w}}]$ to the desired value and focuses on minimizing cross entropy, converging in half the number of iterations.

Visualizing Variational Posteriors To qualitatively investigate the variational posterior distributions, we plot layerwise histograms of learned parameters after the compressed weight sample has been generated. For purposes of comparison, both Mean-Var and Mean-KL parameters have been have been converted to mean and log standard deviation.

Figure 1 reveals striking differences between layerwise Mean-Var and Mean-KL parameter distributions. In terms of the means, Mean-Var parameters collapse to sharp peaks at zero for all layers without any visible tails. In contrast, Mean-KL mean parameters manifest much wider, symmetric distributions centered around zero with heavier tails, resembling shapes akin to Laplace, Gaussian or Student's t-distributions. In terms of the log standard deviation, similarly, Mean-Var parameters form peaked distributions around a particular value with virtually no tails. The distributions of Mean-KL log standard deviations is more spread out, forming distinct shapes for each layer. In general, Mean-Var standard deviations seem to be higher than Mean-KL standard deviations. Furthermore, despite resulting in similar predictive performance, the stark differences in distributional shapes suggest potential qualitative differences between the learned variational posteriors.

Robustness to Pruning To study potential qualitative differences between variational posteriors learned using Mean-Var and Mean-KL parameterizations, we analyze the robustness of the compressed weight sample by setting certain weights to zero using three different strategies:

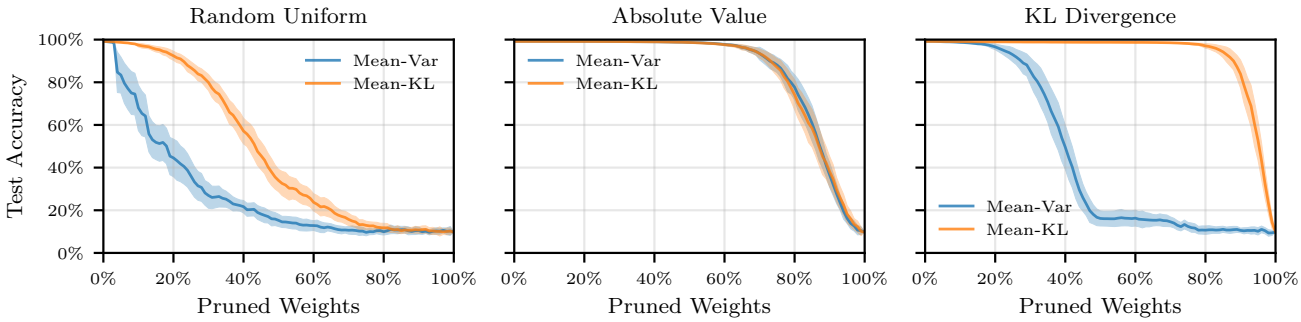

Figure 3. Predictive performance of compressed weight samples from Mean-Var and Mean-KL parameterizations when exposed to pruning via setting weights to zero by selecting the pruned weights uniformly at random (left), based on the smallest absolute values (middle) or based on minimizing KL divergence to a Dirac delta centered at zero (right). Mean $\pm$ standard error over block sizes 20, 30, and 40 with 10 random seeds per block size.

1. Random Uniform: Select pruned weights uniformly at random. This strategy reflects a general notion of robustness due to the uninformed nature of this strategy.

2. Absolute Value: Set the weight with smallest absolute value to zero. This strategy is a simple yet competitive pruning baseline (Blalock et al., 2020), which only depends on the compressed weight sample itself. If the same sample was generated by two different distributions it would still be pruned in the same way.

3. KL Divergence: Prune the weight which minimizes the KL divergence from the variational posterior to a Dirac delta at zero, $\arg\min_i D_{\mathrm{KL}}[\delta_w \| Q_{w_i}]$. For a Gaussian variational posterior with diagonal covariance matrix, this is equivalent to finding the weight with maximal density at zero (see Appendix A for details). This strategy depends on the variational posterior, implying that the same compressed sample would be pruned differently if it was generated by two different distributions.

Figure 3 illustrates how the test accuracy changes as more weights in the compressed sample are pruned to zero. With Random Uniform pruning, Mean-Var test accuracy quickly drops off, already losing more than half the performance after about 20% of the weights have been pruned, and diminishing to performance equal to guessing uniformly at random after roughly 70% of the weights have been set to zero. Mean-KL performance also reduces rapidly, albeit more gracefully. After setting 30% of all weights to zero, a test accuracy of 80% is maintained. Performance equal to guessing is reached after more than 80% of the weights have been pruned. This suggests a general notion of improved robustness of the compressed sample produced by Mean-KL compared to Mean-Var.

With Absolute Value pruning, Mean-Var and Mean-KL perform nearly identical. Both parameterizations roughly maintain full predictive performance until 50% of the weights have been pruned and decay towards random guessing as more weights are set to zero. In particular, this pruning strategy does not depend on the variational posterior and is only informed by the compressed weight sample itself, demonstrating that both parameterizations produce compressed samples which are generally capable of maintaining performance to some degree under pruning.

Finally, both parameterizations perform drastically different under KL Divergence pruning. While Mean-Var test accuracy quickly falls off almost to random guessing after only 50% of the weights have been set to zero, Mean-KL maintains close to 90% test accuracy after pruning 90% of the weights, even outperforming the competitive Absolute Value baseline. Since this pruning strategy is informed by the variational posterior, the results strongly suggest that, compared to Mean-Var, Mean-KL parameterization leads to a superior variational posterior which produces more robust compressed samples. Given that this pruning strategy outperforms the competitive baseline, this property is also not a mere peculiarity but could potentially be leveraged to design more robust algorithms.

5. Conclusion

We demonstrated that MIRACLE with Mean-KL parameterization bypasses the need for time-consuming KL annealing, leading to training convergence after half the number of optimization steps while maintaining predictive performance. Furthermore, Mean-KL parameterization produces more meaningful variational posterior distributions with heavy tails, whereas standard Mean-Var parameterization produces distributions which are sharply peaked at particular values. We illustrated that these qualitative differences result in different properties when exposed to pruning, suggesting that compressed weight samples from Mean-KL are more robust than samples from Mean-Var. Future work should investigate whether faster convergence properties are scalable to larger models and pioneer Mean-KL parameterization for Bayesian neural networks independent of compression. Explicitly utilizing Mean-KL's robustness to design pruning or compression algorithms comprises another possible avenue.

References

Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. What is the State of Neural Network Pruning? In *Proceedings of Machine Learning and Systems*, 2020.

Chen, W., Wilson, J. T., Tyree, S., Weinberger, K. Q., and Chen, Y. Compressing Neural Networks with the Hashing Trick. In *International Conference on Machine Learning*, 2015.

Corless, R. M., Gonnet, G. H., Hare, D. E., Jeffrey, D. J., and Knuth, D. E. On the Lambert W Function. *Advances in Computational Mathematics*, 1996.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., and Saurous, R. A. TensorFlow Distributions. In *arXiv:1711.10604*, 2017.

Flamich, G. Greedy Poisson Rejection Sampling. In *arXiv:2305.15313*, 2023.

Flamich, G., Havasi, M., and Hernández-Lobato, J. M. Compressing Images by Encoding their Latent Representations with Relative Entropy Coding. In *Advances in Neural Information Processing Systems*, 2020.

Flamich, G., Markou, S., and Hernández-Lobato, J. M. Fast Relative Entropy Coding with A* Coding. In *International Conference on Machine Learning*, 2022.

Havasi, M., Peharz, R., and Hernández-Lobato, J. M. Minimal Random Code Learning: Getting Bits Back from Compressed Model Parameters. In *International Conference on Learning Representations*, 2019.

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

Hinton, G. E. and Van Camp, D. Keeping Neural Networks Simple by Minimizing the Description Length of the Weights. In *Conference on Computational Learning Theory*, 1993.

Kenton, J. D. M.-W. C. and Toutanova, L. K. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, 2019.

Winitzki, S. Uniform Approximations for Transcendental Functions. In *Computational Science and Its Applications*, 2003.

A. KL Divergence Pruning

Given a variational posterior $Q_{\mathbf{w}}$ as multivariate Gaussian distribution $\mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with diagonal covariance $\boldsymbol{\Sigma} = \operatorname{diag}(\boldsymbol{\sigma}^2)$, we want to select the dimension i which minimizes the KL divergence to a Dirac delta centered at zero, that is $D_{\mathrm{KL}}[\delta_{\mathbf{w}} \| Q_{\mathbf{w}}]$. Because the distribution of $\mathbf{w}$ is mean-field factorized, it suffices to consider individual dimensions independ of each other. To this end, let $Q_{w_i} = \mathcal{N}(w_i|\mu_i, \sigma_i^2)$ and $P_{w_i} = P_w = \mathcal{N}(w|\nu, \rho^2)$, then

$$D_{\mathrm{KL}}[P_w \| Q_{w_i}] = \log \frac{\sigma_i}{\rho} + \frac{\rho^2 + (\nu - \mu_i)^2}{2{\sigma_i}^2} - \frac{1}{2}, \tag{7}$$

which can be simplified if we are only interested in finding the minimizer because $\log \rho$ and $\frac{1}{2}$ are constant with respect to i,

$$\arg\min_i \; D_{\mathrm{KL}}[P_w \| Q_{w_i}] = \arg\min_i \; \log \sigma_i + \frac{\rho^2 + (\nu - \mu_i)^2}{2{\sigma_i}^2}. \tag{8}$$

Now, to let $P_w \to \delta_w$, we first set $\nu = 0$ and let $\rho \to 0$, yielding

$$\arg\min_i \; D_{\mathrm{KL}}[\delta_w \| Q_{w_i}] = \arg\min_i \; \log \sigma_i + \frac{{\mu_i}^2}{2{\sigma_i}^2} = \arg\max_i \; \log \mathcal{N}(0|\mu_i, \sigma_i), \tag{9}$$

such that choosing the dimension i by minimizing $\log(\sigma_i) + {\mu_i}^2/2{\sigma_i}^2$ will prune the weight whose marginal distribution has the lowest KL divergence to a Dirac delta centered at zero or, equivalently, has the highest log density at zero.

B. Padé Approximation to the Lambert W Function

Since the Lambert W function, defiend by $W(x)e^{W(x)} = x$, cannot be expressed using elementary functions, it has to be implemented by, for example, numerical or analytical approximations. We considered three different approximations to the principal branch of the Lambert W function: Winitzki's approximation for real $x > 0$ (Winitzki 2003, (38)), Halley's method for numerical root-finding with cubic rate of convergence, and a Padé approximation of order [3/2]. Winitzki's approximation for real $x > 0$ is used as initialization for Halley's method in the implementation of TensorFlow Probability (Dillon et al., 2017), however we experienced that the former by itself is not accurate enough and that the latter can be slow and exhibit numerical issues. Instead, we used a Padé approximation of order [3/2], given by

$$W(x) \approx \frac{\frac{13}{720}t(x)^3 + \frac{257}{720}t(x)^2 + \frac{1}{6}t(x) - 1}{\frac{103}{720}t(x)^2 + \frac{5}{6}t(x) + 2}, \tag{10}$$

$$\text{where} \qquad t(x) = \sqrt{2ex + 2}, \tag{11}$$

which was fast and accurate. We did not consider Winitzki's approximation for $-e^{-1} \leq x \leq 1$ (Winitzki 2003, (39)).

C. Implementation Details

Our implementation uses PyTorch (Paszke et al., 2019) and follows Havasi et al. 2019 closely. The LeNet-5 model consists of two convolutional layers and two linear layers, which are applied sequentially. The first convolutional layer has 1 input channel, 20 output channels, a kernel size of 5x5, a stride of 1, and no padding. It is followed by a ReLU activation and a 2D max pooling layer with a kernel size of 2 and a stride of 2. The second convolutional layer has 20 input channel, 50 output channels, and also a kernel size of 5x5, a stride of 1, and no padding. It is also followed by a ReLU activation and a 2D max pooling layer with a kernel size of 2 and a stride of 2. The first linear layer has 800 input features, matching the flattened outputs from the previous layer, 500 output features, and it is followed by a ReLU activation. The second linear layer has 500 input features and 10 output features, matching the number of classes in the MNIST dataset. It is followed by a softmax layer to produce class probabilities. Additionally, weight hashing (Chen et al., 2015) is used in the second convolutional layer and the first linear layer to reduce the effective number of weights by a factor of 2x and 64x respectively. The layerwise log standard deviation parameters of the coding distribution were initialized to -2. For Mean-Var parameters, the means were initialized using PyTorch's default initialization and the log standard deviations were initialized to -10. For Mean-KL parameters, τ_w was initialized by passing PyTorch's default initialization through the analytical inverse of Equation (6) and γ_w was initialized to 0. After initial variational training, we perform 100 fine-tuning steps in-between compressing blocks.