BMRS: Bayesian Model Reduction for Structured Pruning

Dustin Wright, Christian Igel, and Raghavendra Selvan
Department of Computer Science, University of Copenhagen
{dw,igel,raghav}@di.ku.dk

Abstract

Modern neural networks are often massively overparameterized leading to high compute costs during training and at inference. One effective method to improve both the compute and energy efficiency of neural networks while maintaining good performance is structured pruning, where full network structures (e.g. neurons or convolutional filters) that have limited impact on the model output are removed. In this work, we propose Bayesian Model Reduction for Structured pruning (BMRS), a fully end-to-end Bayesian method of structured pruning. BMRS is based on two recent methods: Bayesian structured pruning with multiplicative noise, and Bayesian model reduction (BMR), a method which allows efficient comparison of Bayesian models under a change in prior. We present two realizations of BMRS derived from different priors which yield different structured pruning characteristics: 1) $\text{BMRS}_{\mathcal{N}}$ with the truncated log-normal prior, which offers reliable compression rates and accuracy without the need for tuning any thresholds and 2) $\text{BMRS}_{\mathcal{U}}$ with the truncated log-uniform prior that can achieve more aggressive compression based on the boundaries of truncation. Overall, we find that BMRS offers a theoretically grounded approach to structured pruning of neural networks yielding both high compression rates and accuracy. Experiments on multiple datasets and neural networks of varying complexity showed that the two BMRS methods offer a competitive performance-efficiency trade-off compared to other pruning methods.[1]

1 Introduction

Modern neural networks come with an increasing computational burden, as scale is often seen to be associated with performance [34]. The response to this has been a focus on research around the topic of neural network efficiency [3], where the goal is to reduce the computational cost of a system while maintaining other desirable metrics. As such, selecting a method to improve efficiency comes with many tradeoffs, including how to balance compute and energy consumption with accuracy [35].

Neural network pruning seeks to do this by removing parts of a network which have limited impact on its output. This comes in two primary forms: unstructured pruning, where individual weights are removed, and structured pruning, where entire neural network structures such as neurons and convolutional filters are removed [27]. Structured pruning is often desirable as unstructured pruning can result in sparse computations which are energy intensive on current hardware, while structured pruning can maintain more energy efficient dense operations [14, 17, 31]. Many ways to perform structured pruning have been proposed, but the challenge of how to appropriately balance accuracy and complexity in a principled manner has remained.

In this work, we address this challenge by proposing BMRS: **B**ayesian **M**odel **R**eduction for **S**tructured pruning. BMRS is a principled method based on combining two complementary lines

[1] Source code: https://github.com/saintslab/bmrs-structured-pruning/

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-2045

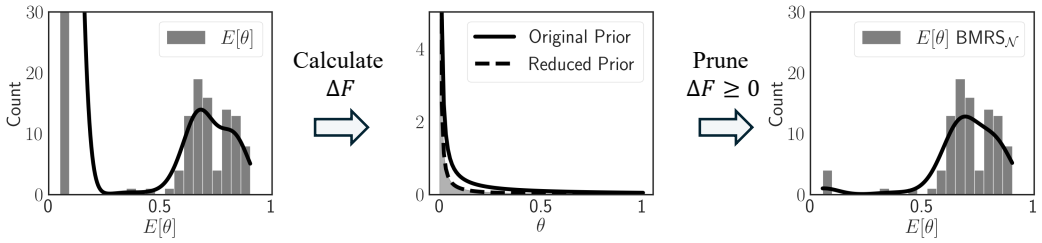


Figure 1: BMRS uses BMR to perform structured pruning under multiplicative noise by calculating the change in log-evidence of noise variables θ under a prior which would shrink them to 0.

of work: Bayesian structured pruning with multiplicative noise [30] and Bayesian model reduction (BMR) [6], a method of efficient Bayesian model comparison under a change in prior. Multiplicative noise allows one to flexibly induce sparsity at any structural level without the need to use computationally complex spike-and-slab priors [16, 29], while BMR enables principled pruning rules without the need for task-specific threshold tuning. Starting with the approach from [30], we derive two versions of BMRS using different priors which offer their own benefits. BMRS$_{\mathcal{N}}$ is based on the truncated log-normal prior and has the benefit of achieving a high compression rate without needing to tune a threshold for compression, while BMRS$_{\mathcal{U}}$ offers tunable compression by controlling the allowable precision of noise variables in the network. In sum, our contributions are:

- BMRS: a method for Bayesian structured pruning based on multiplicative noise and Bayesian model reduction;
- Derivations of pruning algorithms for two priors with theoretical motivation;
- Empirical results on a range of neural networks and datasets demonstrating high compression rates without any threshold tuning, with more extreme compression achievable via a parameter controlling allowable precision.

2 Related work

The primary goal of neural network pruning is to determine the elements of a network which can be removed with minimal impact on the output. Ideally, a pruning method ranks all elements in the order in which they can be removed and provides a criterion for truncating the resulting ordered list. Since the early works on gradient based methods for pruning [23, 12], the literature around neural network pruning has expanded rapidly, with the two main lines of work exploring pruning individual weights (unstructured pruning) and pruning full network structures (structured pruning). For a recent survey, see [27]. The closest related works to ours are those pruning methods which perform Bayesian pruning [30, 28, 10, 16, 26], and those which use Bayesian model reduction to determine what elements to remove from a neural network [5, 29].

Bayesian pruning. Bayesian structured pruning was first explored in Kingma et al. [20], where the authors demonstrate that dropout has a Bayesian interpretation as multiplicative noise with a sparsity inducing prior. The studies of [30, 28, 10] follow this work by explicitly modeling the random noise in dropout with different priors, [30] using a truncated log-uniform prior and [28, 10] using horseshoe priors. Following this, the works of [2, 15, 16, 33, 29, 26] have explored pruning of Bayesian neural networks (BNNs) with spike-and-slab priors to induce both weight sparsity and group sparsity with flat and hierarchical priors, respectively. [16] demonstrate that thresholdless pruning is achievable by placing an explicit spike-and-slab prior on the nodes of a BNN to induce group sparsity. However, this setup requires complex and carefully constructed posteriors due to the discrete nature of spike-and-slab distributions and is thus computationally inefficient [29, 16].

Bayesian model reduction. Bayesian model reduction, discussed in detail in §3.2, is an efficient method of Bayesian model comparison which allows for analytic solutions for the model evidence under a change in priors. BMR has found application across multiple scientific disciplines [9, 8, 18], and has recently been used as a method for neural network pruning [29, 5]. More specifically, [5] demonstrate the benefits of BMR-based pruning for the case of a BNN with a Gaussian prior on

the weights, and [29] demonstrate the utility of BMR for unstructured pruning of BNNs with priors inducing both weight and group sparsity.

3 Problem formulation

3.1 Structured pruning with multiplicative noise and variational inference

We approach the problem of structured pruning using sparsity inducing multiplicative noise as described in [30]. In this setting, we have a dataset consisting of N i.i.d. input-output pairs, $\mathcal{D} = \{(\mathbf{x}_j, y_j)\ \forall j = 1, \ldots, N\}$. We consider a parametric model, here a deep neural network, that maps the input data $\mathbf{x}_j$ to their output y_j using the trainable parameters $\mathbf{W}$ giving rise to the likelihood function $p(\mathcal{D}|\Theta, \mathbf{W}) = \prod_{j=1}^{N} p(y_j|\mathbf{x}_j, \Theta, \mathbf{W})$. In addition to the trainable weights, $\mathbf{W}$, the model consists of the sparsity inducing multiplicative noise given by the random variable, Θ, with prior $p(\Theta)$. This is in contrast to BNNs where the weights are random variables but aligns with the setting when using multiplicative noise for Bayesian pruning [30].

The effect of the multiplicative noise $\theta_i \in \Theta$ for a structural element in a neural network with index i, parameters $\mathbf{w}_i$, and input $\mathbf{h}_{i-1}$ is given as

$$\mathbf{h}_i = \theta_i \cdot (\mathbf{w}_i \mathbf{h}_{i-1}) \quad , \quad \theta_i \sim p(\theta_i). \tag{1}$$

We note that $\mathbf{w}_i$ could be the parameters of any structural element in the network, for example, a single neuron or an entire convolutional filter. Given this, we would like to learn the maximum likelihood estimate (MLE) $\hat{\mathbf{w}}_i$ of the weights as well as the posterior distribution over the multiplicative noise, $p(\theta_i|\mathcal{D}, \hat{\mathbf{w}}_i)$, when using a sparsity inducing prior $p(\theta_i)$ such that θ_i favors values closer to 0.

Following [30], the neural network weights are learned via gradient descent as in standard deep learning model optimization. The posterior distribution, $p(\theta|\mathcal{D}, \hat{\mathbf{w}}_i) = p(\mathcal{D}|\theta_i, \hat{\mathbf{w}}_i)p(\theta_i)/p(\mathcal{D})$, however, is intractable. We resort to a variational approximation from a tractable family of approximating distributions, $q_\phi(\theta)$, parameterized by ϕ (for the sake of brevity we do not indicate the dependence on $\mathbf{w}_i$ and omit the subscript i). The parameters ϕ are obtained by optimizing the following objective w.r.t. θ:

$$\mathrm{F}[p, q] = D_{\mathrm{KL}}[q_\phi(\theta)||p(\theta|\mathcal{D})] \overset{c}{=} D_{\mathrm{KL}}[q_\phi(\theta)||p(\theta)] - \mathbb{E}_{q_\phi}[\log p(y_j|\mathbf{x}_j, \theta, \hat{\mathbf{w}})] \tag{2}$$

This is the commonly used variational free energy (VFE) or negative evidence lower bound (ELBO) [4]. Here $\overset{c}{=}$ denotes equality up to a positive constant.

The expectation $\mathbb{E}_{q_\phi}[\cdot]$ is approximated by a Monte Carlo estimator acting on minibatch samples from $\mathcal{D}$, and reparameterization allows to backpropagate gradients through stochastic variables [19, 28]. Under this reparameterization, the variational distribution, $q_\phi(\theta)$, becomes a deterministic function of the non-parametric noise $\epsilon \sim p(\epsilon)$ and the VFE is calculated as

$$\mathrm{F}[p, q] \overset{c}{=} D_{\mathrm{KL}}[q_\phi(\theta)||p(\theta)] - \sum_{(x_j, y_j) \in \mathcal{D}} \log p(y_j|x_j, \theta = f(\phi, \epsilon); \hat{\mathbf{w}}), \tag{3}$$

where f is a function that allows us to sample θ via deterministic parameters ϕ and the non-parametric stochastic variable, ϵ. Optimization of Equation 3 allows us to jointly learn $\hat{\mathbf{w}}$ and θ. The particular choice of priors and the approximating distributions to induce sparsity are discussed in §4.1.

3.2 Bayesian model reduction

Bayesian model reduction (BMR) allows one to *efficiently* compute the change in VFE (Equation 2) under a change in prior without the need to re-estimate model parameters. To perform pruning, one can start out by selecting a broad prior for the original model estimation and then pick a narrower prior (i.e. reduced prior) with the density concentrated around 0. Then, BMR can be used to determine if the VFE is greater under the reduced model, and prune those parameters for which this condition holds. We briefly describe how this is achieved in the general case, followed by the specific realization for BMRS in §4.2; for further details see [6].

Consider the likelihood function $p(\mathcal{D}|\theta)$ and a prior $p(\theta)$ on the variable θ. We can introduce a new prior $\tilde{p}(\theta)$ which shares the same likelihood as the original model (i.e. $p(\mathcal{D}|\theta) = \tilde{p}(\mathcal{D}|\theta)$) and get:

$$p(\mathcal{D}|\theta) = \frac{p(\theta|\mathcal{D})p(\mathcal{D})}{p(\theta)} = \frac{\tilde{p}(\theta|\mathcal{D})\tilde{p}(\mathcal{D})}{\tilde{p}(\theta)} \Rightarrow \tilde{p}(\theta|\mathcal{D}) = \frac{p(\mathcal{D})}{\tilde{p}(\mathcal{D})} p(\theta|\mathcal{D}) \frac{\tilde{p}(\theta)}{p(\theta)} \tag{4}$$

https://doi.org/10.52202/079017-2045

By marginalizing over θ and taking the log, we obtain the difference in log evidence as:

$$\log \tilde{p}(\mathcal{D}) - \log p(\mathcal{D}) = \log \int p(\theta|\mathcal{D}) \frac{\tilde{p}(\theta)}{p(\theta)} d\theta \approx \log \int q_\phi(\theta) \frac{\tilde{p}(\theta)}{p(\theta)} d\theta = \log \mathbb{E}_{\tilde{p}} \left[\frac{q_\phi(\theta)}{p(\theta)} \right] \quad (5)$$

More concisely, we call the change in log evidence ΔF and thus have:

$$\Delta F \triangleq \log \tilde{p}(\mathcal{D}) - \log p(\mathcal{D}) \approx \log \mathbb{E}_{\tilde{p}} \left[\frac{q_\phi(\theta)}{p(\theta)} \right] \quad (6)$$

If the new prior, $\tilde{p}(\theta)$, is selected so that θ would be removed, pruning can be performed when $\Delta F \geq 0$. Additionally, when the type of distributions between $p(\theta)$, $\tilde{p}(\theta)$, and $q_\phi(\theta)$ are the same or similar (e.g. Gaussian), ΔF can be calculated efficiently in closed form (see [7]).

We presented BMR for a general likelihood function $p(\mathcal{D}|\theta)$; it holds analogously for the likelihood function $p(\mathcal{D}|\theta, \mathbf{W})$ introduced with the multiplicative noise described in §3.1.

4 Bayesian model reduction for structured pruning (BMRS)

Our goal is to derive a principled structured pruning algorithm starting from the general formulation in § 3 which can automatically determine which structures to prune. BMRS accomplishes this by following the multiplicative noise setup in [30] with BMR used on the noise terms. Figure 1 illustrates the general approach, where ΔF is calculated for a model trained with multiplicative noise under a reduced prior, and elements of the model are removed if the new VFE is greater. We next describe the multiplicative noise layer trained using Equation 2, and then derive two variants of BMRS from Equation 6 using different reduced priors.

4.1 Multiplicative noise layer

The concept of multiplicative noise inducing sparsity in neural networks was first introduced with variational dropout, where [20] show that dropout has a Bayesian interpretation as multiplicative noise with a log-uniform prior. One can use this interpretation of dropout in order to explicitly learn dropout parameters, θ_i, as in §3, by selecting appropriate prior and variational distributions and optimizing Equation 2 directly. [30] propose to do so by using the truncated log-uniform distribution as a prior and the truncated log-normal distribution as the variational distribution. As such, the variational approximation can be performed using

$$\mathbf{h}_i = \theta_i \cdot (\mathbf{w}_i \mathbf{h}_{i-1}); \quad q_\phi(\theta_i) = \text{LogN}_{[a,b]}(\theta_i|\mu_i, \sigma_i^2); \quad p(\theta_i) = \text{LogU}_{[a,b]}(\theta_i) \quad (7)$$

with bounded support between a and b and $0 < a < b \leq 1$. We refer to [30] for details on how to learn q_ϕ, which is obtained by optimizing Equation 3.

The log-uniform distribution serves as a sparsity inducing prior as most of its density is concentrated around 0 (see panel 2 in Figure 1). Additionally, it acts as a regularizer on the floating point precision of the multiplicative noise terms [20]. In [30] this is used to perform structured pruning by removing all structures $\mathbf{h}_i$ where the signal-to-noise ratio of the noise term θ_i falls below a pre-defined threshold. We next show how to derive principled pruning algorithms based on BMR which induce sparsity while maintaining accuracy without the need for tuning pruning thresholds.

4.2 Deriving BMRS

Our goal is to use BMR in order to perform structured pruning of models trained with multiplicative noise. To do so, we must select a new prior $\tilde{p}(\theta)$ from which we can: 1) induce sparsity; 2) efficiently calculate ΔF and; 3) prune the network while maintaining good performance.

Selecting the reduced prior, $\tilde{p}(\theta)$, is straightforward when the prior and approximate posterior are the same type of distributions. For example, in a fully BNN where one assumes a prior distribution of $\mathcal{N}(\Theta|\mathbf{0}, \mathbf{I})$ over all the model weights with a mean-field variational approximation resulting in a factorisation over individual weights, $\mathcal{N}(\theta|\mu, \sigma^2)$, the three criteria above can be met when one selects a Gaussian reduced prior with slight variance around 0 i.e. $\mathcal{N}(\theta|0, \epsilon), \epsilon \approx 0$.[2] However, in the case of

[2]It can be shown that ΔF is calculable efficiently in closed form for this setup; see e.g. [7]

multiplicative noise, our prior and variational distributions are of different types and thus the selection of the reduced prior is not immediately obvious. Here, we derive and compare the characteristics of two different reduced priors: one based on a truncated log-normal distribution, which we can use to approximate a Dirac delta at 0 (BMRS$_{\mathcal{N}}$), and one based on a truncated log-uniform distribution with reduced support (BMRS$_{\mathcal{U}}$).

4.2.1 BMRS with log-normal reduced prior (BMRS$_{\mathcal{N}}$)

First, we derive ΔF when using the log-uniform distribution as the original prior, $p(\theta) = \text{LogU}_{[a,b]}(\theta)$, and the truncated log-normal distribution as the reduced prior, $\tilde{p}(\theta) = \text{LogN}_{[a,b]}(\theta|\tilde{\mu}_p, \tilde{\sigma}_p^2)$. We select a truncated log-normal distribution, as it matches the variational distribution $q_\phi(\theta)$, and the log-uniform prior, because it is a special case of the log-normal distribution when the variance goes to infinity. Because of this, we expect that ΔF will have a closed form solution, and that the computation will be efficient. We briefly present the results of the derivation here; for the full derivation see § A.1.

We can use the specific forms of $p(\theta)$ and $\tilde{p}(\theta)$ for the truncated log-uniform and truncated log-normal distributions, respectively, in Equation 6 to determine ΔF:

$$\Delta F \approx \log \mathbb{E}_{\tilde{p}}\left[\frac{q_\phi(\theta)}{p(\theta)}\right] = \log \frac{Z_{\tilde{q}}(\log b - \log a)}{Z_{\tilde{p}} Z_q} + \frac{1}{2}\log \frac{\tilde{\sigma}_q^2}{2\pi\tilde{\sigma}_p^2\sigma_q^2} - \frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2} + \frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2} - \frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right) \quad (8)$$

with $\tilde{\sigma}_q^2 = \left(\frac{1}{\sigma_q^2} + \frac{1}{\tilde{\sigma}_p^2}\right)^{-1}$ and $\tilde{\mu}_q = \tilde{\sigma}_q^2\left(\frac{\mu_q}{\sigma_q^2} + \frac{\tilde{\mu}_p}{\tilde{\sigma}_p^2}\right)$. Here, $Z_p = \Phi(\beta_p) - \Phi(\alpha_p)$; $\Phi(t) = \frac{1}{2}[1 + \text{erf}(\frac{t}{\sqrt{2}})]$ is the CDF of the standard Normal distribution, $t \sim \mathcal{N}(0,1)$; $\alpha_p = (a - \mu_p)/\sigma_p$; and $\beta_p = (b - \mu_p)/\sigma_p$.

As can be seen from Equation 8, the calculation for ΔF can be performed directly using only the statistics of the priors and variational distribution. In order for Equation 8 to induce sparsity, we must select a $\tilde{\mu}_p$ and $\tilde{\sigma}_p^2$ that effectively collapse θ to 0. To achieve this, we can approximate a Dirac delta at 0 by selecting $\tilde{\mu}_p$ to be close to 0 (e.g., the lower bound of truncation a), and $\tilde{\sigma}_p^2$ to be sufficiently small. We will demonstrate in §5 that this reduced prior results in high sparsity while maintaining performance without any need for tuning pruning thresholds.

4.2.2 BMRS with log-uniform reduced prior (BMRS$_{\mathcal{U}}$)

Next, we derive the change in VFE, ΔF, when using a truncated log-uniform distribution as the original prior, $p(\theta) = \text{LogU}_{[a,b]}(\theta)$, and a truncated log-uniform distribution with reduced support as the reduced prior, $\tilde{p}(\theta) = \text{LogU}_{[a',b']}(\theta)$. We select a reduced truncated log-uniform distribution for the same reasons as the truncated log-normal: we expect that ΔF will have an efficiently calculable closed form, given that the priors are of the same type and are a special case of the variational distribution. The PDF of the reduced truncated log-uniform distribution is given as follows:

$$\tilde{p}(\theta) = \text{LogU}_{[a',b']}(\theta) = \begin{cases} \left(\theta \log \frac{b'}{a'}\right)^{-1}, & a < a' \leq \theta \leq b' < b \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

Using this, we can directly solve the integral under the expectation given in Equation 6 for ΔF (full details in § A.2):

$$\exp \Delta F \approx \mathbb{E}_{\tilde{p}}\left[\frac{q_\phi(\theta)}{p(\theta)}\right] = \int_a^b \text{LogU}_{[a',b']}(\theta)\frac{q_\phi(\theta)}{\text{LogU}_{[a,b]}(\theta)}d\theta = \frac{\log \frac{b}{a}}{\log \frac{b'}{a'}} q_\phi(a' \leq \theta_i \leq b') \quad (10)$$

where $q_\phi(a' \leq \theta_i \leq b')$ is the CDF of the variational distribution evaluated between a' and b'. We know from Equation 5 that the VFE under $\tilde{p}(\theta)$ is greater when $\exp \Delta F \geq 1$. Plugging this in:

$$1 \leq \frac{\log \frac{b}{a}}{\log \frac{b'}{a'}} q_\phi(a' \leq \theta_i \leq b') \Rightarrow \frac{\log \frac{b'}{a'}}{\log \frac{b}{a}} \leq q_\phi(a' \leq \theta \leq b') \quad (11)$$

Here, the left hand side of the inequality is the CDF of the truncated log-uniform distribution between a' and b'. In other words, when the new prior is a log-uniform distribution with reduced support,

https://doi.org/10.52202/079017-2045

Algorithm 1: Training and pruning with BMRS

input : dataset $\mathcal{D}$; neural network with deterministic weights $\mathbf{W}$ and variational parameters ϕ; original prior p(Θ); reduced prior $\tilde{p}(\Theta)$; number of training epochs e_T; number of fine-tuning epochs e_F; number of pruning epochs P

$j \leftarrow 0$
while $j < e_T$ and $\mathbf{W}, \phi$ not converged **do**
  Train ϕ and $\mathbf{W}$ on $\mathcal{D}$ using Equation 3
  if $j \bmod P = 0$ **then**
    forall θ_i in Θ **do**
      dF $\leftarrow \Delta F(p(\theta_i), \tilde{p}(\theta_i), q_\phi(\theta_i))$
      if dF ≥ 0 **then**
        $\phi \leftarrow \phi \setminus \phi_i$
        $\mathbf{W} \leftarrow \mathbf{W} \setminus \mathbf{w}_i$
  $j \rightarrow j + 1$
Fine tune $\mathbf{W}$ and ϕ on $\mathcal{D}$ for e_F epochs

the BMR pruning criterion amounts to a comparison between the CDF of the original prior and the variational distribution along the interval $[a', b']$. Additionally, Equation 11 shows that this is generalizable to any variational distribution with support broader than the reduced prior.

BMRS$_\mathcal{U}$ pruning and connection to floating point precision. To see how BMRS$_\mathcal{U}$ can be used for pruning, we first briefly summarize the relationship between the log-uniform distribution and floating point numbers. Floating point numbers are commonly encoded in binary as a combination of a sign bit s, a set of exponent bits e, and a set of mantissa bits m denoting the fractional part of a real-number: $r = s \cdot (m/2^{p-1}) \cdot 2^e$, where p determines the precision of the encoding. As discussed in [11], the mantissae of "naturally observed" floating point numbers (e.g., natural constants) tend to follow a log-uniform distribution and repeated multiplications/divisions on a digital computer transform a broad class of distributions towards a log-uniform distribution

$$m \sim (m \log B)^{-1}, \quad 1/B \leq m \leq 1, \tag{12}$$

where B is the base of the number system. In the case where the mantissa uses p bits, there are 2^p representable fractional numbers so $B = 2^p$. As such, p determines the smallest fractional value which can be represented. We can use this to have the reduced prior cover a finite range of high precision values which are acceptable to prune. To accomplish this, we use a reduced log-uniform prior of the following form by selecting two integers p_1, p_2 where $0 \leq p_1 < p_2$:

$$\tilde{p}(\theta) = \begin{cases} \left(\theta \log \frac{2^{p_2}}{2^{p_1}}\right)^{-1}, & 1/2^{p_2} \leq \theta \leq 1/2^{p_1} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Thus we can reduce the prior to a range of precision between p_1 and p_2 by selecting $a' = 1/2^{p_2}$ and $b' = 1/2^{p_1}$. Equation 11 then has a natural interpretation as comparing the probability of drawing a random mantissa from the interval $[1/2^{p_2}, 1/2^{p_1}]$ to the probability of drawing a value within that interval from the variational distribution. If we select p_2 to be the limit of the precision of values in the number system used ($p_2 = 23$ for single-point precision), then we can interpret p_1 as determining the prunable range of precision of all variables Θ. The accuracy-complexity tradeoff inherent in the selection of $\tilde{p}(\theta)$ is then controlled through p_1, the desired cutoff of the precision of the network.

4.3 Training and pruning

The details on how to train a network and use BMRS for pruning are given in Algorithm 1. We train a model for a fixed number of epochs (or until convergence) and perform pruning every P epochs. This lends itself to either post-training pruning, where the network is fully trained followed by pruning and fine-tuning, or continuous pruning, where pruning is performed during model training. In our experiments, we explore both of these setups and contrast BMRS with alternative pruning methods.

5 Experiments

We demonstrate the pruning behavior of BMRS through several experiments with neural networks of varying complexity measured as the number of trainable parameters. We use the following datasets (full details in Appendix B): MNIST [22], Fashion-MNIST [37], CIFAR10 [21], and TinyImagenet. For MNIST, Fashion-MNIST, and CIFAR10 we experiment with both a multi-layer perceptron (MLP) and a small CNN (Lenet5 [24]). Pruning layers are applied after each fully connected layer for the MLP, and for each convolutional filter and fully connected layer for Lenet5. For CIFAR10 and TinyImagenet, we further experiment with a pretrained ResNet-50 [13] and a pretrained vision transformer (ViT) [36]. For ResNet-50, we apply pruning layers after each layer of batch normalization, and for ViT we apply pruning layers to the output of each transformer block. For the multiplicative noise layers, we set the left bound of truncation to be $\log a = -20$ and the right bound of truncation to be $\log b = 0$. Hyperparameters for the MLPs and Lenet5 are tuned on a model with no pruning performed and kept the same for each variant (see Appendix C). We perform experiments using the following model variants and baselines which cover both Bayesian pruning criteria (e.g. ours, SNR, and $\mathbb{E}_{q_\phi}[\theta]$) as well as magnitude pruning (L2):

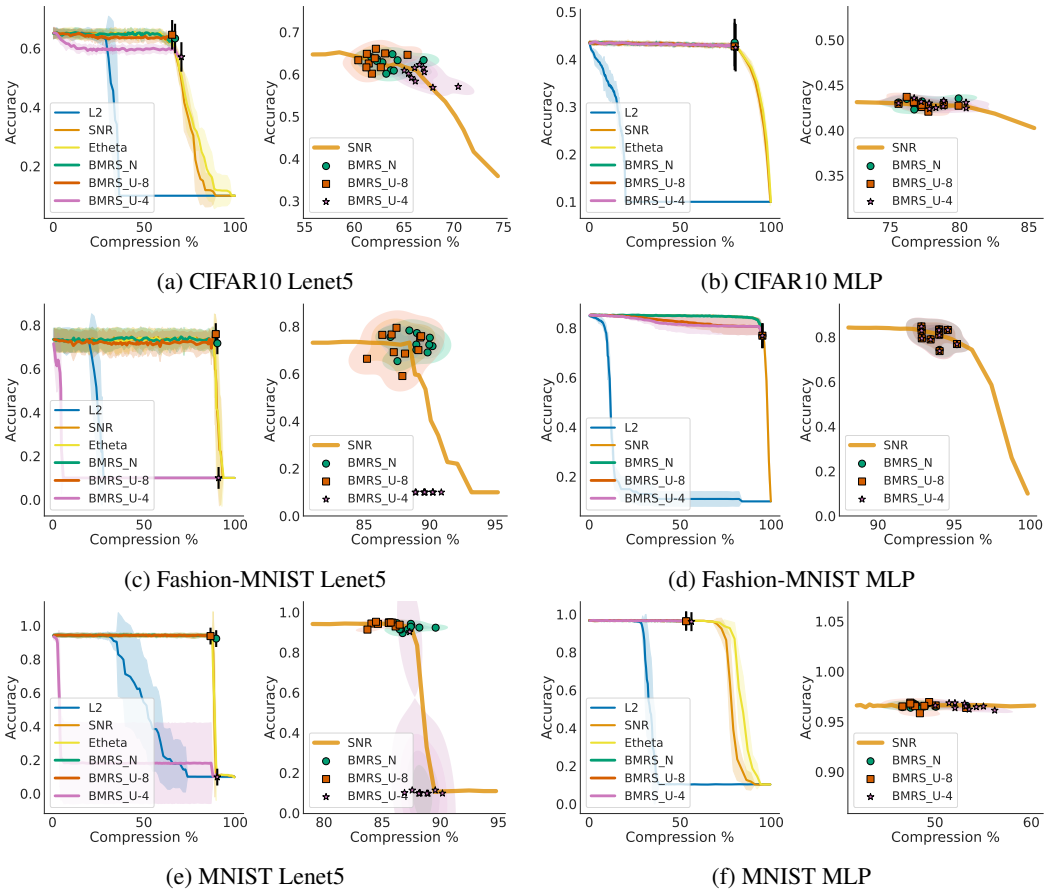
- **None:** A baseline with no compression and no multiplicative noise.
- **L2:** A magnitude pruning baseline based on the L2 Norm of weight vectors (matrices in the case of convolutional filters) at the input of each neuron to be pruned [25]. We set the pruning threshold to the compression rate achieved by BMRS$_{\mathcal{N}}$ using the same settings of a given experiment.
- $\mathbb{E}_{q_\phi}[\theta]$: The expected value of noise variables θ. For continuous pruning, we use a set threshold of 0.1.
- **SNR:** The signal-to-noise ratio $\mathbb{E}_{q_\phi}[\theta] \big/ \sqrt{\text{Var}[\theta]}$ as used in [30]. For continuous pruning, we use a set threshold of 1 as recommended in [30].
- **BMRS$_{\mathcal{N}}$:** BMRS using the log-normal prior from Equation 6. In order to reduce the prior to 0, we set $\tilde{\mu}_p$ to the left bound of truncation (a), and $\tilde{\sigma}_p^2$ to 10^{-12}.
- **BMRS$_{\mathcal{U}}$-p_1:** BMRS using the log-uniform prior from Equation 11. In our experiments, we set a' to be the limit of the precision of single-point floats (p_2 = 23 so $a' = 1/2^{23}$) and b' to either p_1 = 8-bit precision ($b' = 1/2^8$) or p_1 = 4-bit ($b' = 1/2^4$).

Post-training pruning. We first look at the behavior of BMRS when used in the post-training setting. To do so, we first train a model on a given dataset, then use each method to rank the neurons based on their pruning function (L2 norm, signal-to-noise ratio, or ΔF). To observe the accuracy at different compression rates, neurons are progressively removed based on their rank, and the model is fine-tuned for one epoch before measuring the test accuracy. For BMRS methods, we additionally stop pruning once $\Delta F < 0$ for a given structure. The plots of accuracy vs. compression for 10 different random seeds are given in Figure 2 (Further experiments given in Appendix E).

First, we find that BMRS generally stops compressing near the knee point of the trade-off curve – a preferred solution of a Pareto front if there is no a priori preferences – in all settings except for BMRS$_{\mathcal{U}}$-4 which only does so in 4 out of 6 settings. Notably, BMRS$_{\mathcal{N}}$ accomplishes this with no need to tune additional thresholds as is common in pruning literature. To further visualize this, the right plot in each subfigure shows a scatter plot of the accuracy at the maximum compression rate (pruning all neurons where $\Delta F \geq 0$) along with the curve of accuracy vs. compression for SNR pruning near the knee point. We can see that the density of points for BMRS is concentrated near the optimal point in all cases except for the MLP on MNIST, indicating the robustness of the proposed methods.

We additionally observe much similarity in the curves for BMRS and SNR pruning, suggesting that they may be performing similar functions. To further investigate this, we look at the Spearman rank correlation coefficient [32] of the neurons based on their respective functions in Figure 3 (plots for additional datasets in Appendix E). We see that BMRS$_{\mathcal{N}}$ tends to have a high correlation with SNR, suggesting that it learns a qualitatively similar function with the benefit of providing a threshold for compression. BMRS$_{\mathcal{U}}$, on the other hand, tends to have very low or even negative correlation. This, combined with the more rapidly declining accuracy for a given level of compression, suggests that BMRS$_{\mathcal{U}}$ is only apt for determining a single split into elements to keep and to remove, but does not provide an accurate ranking of the elements..

https://doi.org/10.52202/079017-2045

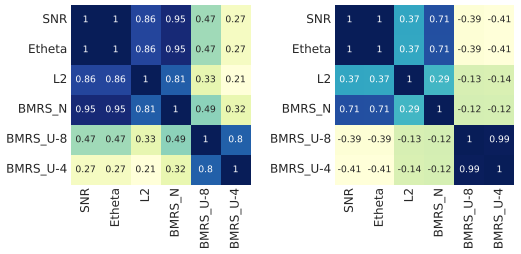


(a) CIFAR10 Lenet5 (b) CIFAR10 MLP

(c) Fashion-MNIST Lenet5 (d) Fashion-MNIST MLP

(e) MNIST Lenet5 (f) MNIST MLP

Figure 2: Accuracy vs. compression for post-training pruning on CIFAR10, Fashion-MNIST, and MNIST. The left plot in each subfigure shows the average accuracy across 10 seeds, shading shows the standard deviation. For BMRS, we mark the maximum compression rate based on when $\Delta F \geq 0$. The right plot in each subfigure shows a scatter plot and kernel density estimation of accuracy vs. compression of BMRS compared to SNR accuracy. BMRS$_{\mathcal{N}}$ and BMRS$_{\mathcal{U}}$-8 consistently stop pruning near the knee point, a preferred trade-off solution.



(a) CIFAR10 Lenet5 (b) CIFAR10 MLP

Figure 3: Average Spearman's rank correlation between the ranks of neurons for pruning when using different methods on CIFAR10 (plots for additional datasets are given in Appendix E).

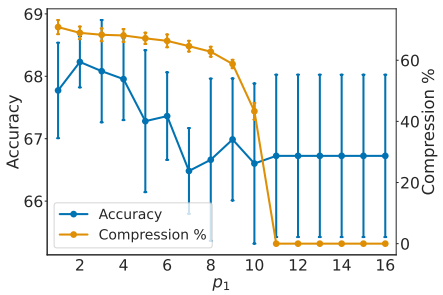


Figure 4: Accuracy and compression rate vs. p_1 for BMRS$_{\mathcal{U}}$ on CIFAR10 with Lenet5. Results are averaged across 10 seeds with standard deviation indicated by the error bars.

Continuous pruning. Next, we experiment with continuous pruning, where neurons are pruned continuously throughout training based on either a provided pruning threshold (SNR, $\mathbb{E}_{q_\phi}[\theta]$) or ΔF (BMRS). For SNR, we prune a neuron when its SNR falls below 1 (as in [30]), and for $\mathbb{E}_{q_\phi}[\theta]$ we set a threshold of 0.1. For L2 pruning, we perform post-training pruning based on the compression rate

Table 1: Parameter compression % and accuracy for different baseline methods and settings of the proposed method. Standard deviations over ten runs are included. Best accuracy for the compression methods is given in bold.

Pruning Method	MNIST		Fash-MNIST		CIFAR10	
	Comp. (%)	Acc.	Comp. (%)	Acc.	Comp. (%)	Acc.
			MLP			
None	0.00 ± 0.00	97.43 ± 0.14	0.00 ± 0.00	88.17 ± 0.20	0.00 ± 0.00	44.94 ± 0.40
L2	43.11 ± 2.06	10.39 ± 0.32	87.86 ± 2.27	18.23 ± 10.22	42.89 ± 2.64	10.00 ± 0.00
$E[\theta]$	52.08 ± 1.71	96.88 ± 0.15	91.76 ± 0.81	85.59 ± 0.26	77.99 ± 1.54	43.39 ± 0.46
SNR	58.57 ± 2.01	96.92 ± 0.08	99.83 ± 0.00	10.00 ± 0.00	75.93 ± 1.26	43.97 ± 0.46
BMRS$_{\mathcal{N}}$	48.86 ± 1.32	96.95 ± 0.19	93.20 ± 0.66	84.99 ± 0.35	76.36 ± 1.08	43.59 ± 0.29
BMRS$_{\mathcal{U}}$-8	48.73 ± 1.90	96.93 ± 0.16	93.02 ± 0.81	85.01 ± 0.32	77.17 ± 0.98	43.45 ± 0.42
BMRS$_{\mathcal{U}}$-4	54.47 ± 1.74	$\mathbf{96.99 \pm 0.13}$	91.57 ± 0.71	$\mathbf{85.79 \pm 0.34}$	76.63 ± 0.94	$\mathbf{44.06 \pm 0.40}$
			Lenet5			
None	0.00 ± 0.00	99.07 ± 0.09	0.00 ± 0.00	89.16 ± 0.27	0.00 ± 0.00	67.62 ± 0.77
L2	83.42 ± 1.92	11.35 ± 0.00	83.62 ± 1.69	10.00 ± 0.00	52.29 ± 2.18	10.00 ± 0.00
$E[\theta]$	88.29 ± 1.00	51.30 ± 41.12	89.71 ± 0.56	50.93 ± 33.45	66.19 ± 1.36	65.83 ± 0.90
SNR	92.66 ± 5.77	62.70 ± 41.93	98.47 ± 3.45	17.01 ± 21.03	70.29 ± 2.02	$\mathbf{67.68 \pm 0.52}$
BMRS$_{\mathcal{N}}$	86.90 ± 1.15	95.59 ± 0.94	88.02 ± 1.00	77.90 ± 2.44	62.87 ± 1.64	66.14 ± 0.70
BMRS$_{\mathcal{U}}$-8	86.11 ± 1.37	95.27 ± 1.02	87.61 ± 0.72	77.23 ± 3.49	62.54 ± 1.49	66.28 ± 1.07
BMRS$_{\mathcal{U}}$-4	87.58 ± 1.01	$\mathbf{96.66 \pm 0.59}$	88.72 ± 0.73	$\mathbf{81.10 \pm 1.50}$	68.07 ± 1.95	67.66 ± 0.59

achieved by BMRS$_{\mathcal{N}}$. Neurons are pruned after every epoch during training, followed by 10 epochs of fine-tuning at the very end of training.

We compare the raw performance of each variant using an MLP and Lenet5 on MNIST, Fashion-MNIST, and CIFAR10 in Table 1. First, we note that using the L2 norm with the same compression rate as BMRS$_{\mathcal{N}}$ results in a degenerate model; the accuracy degrades to random in all settings. Additionally, we see that using the SNR as a pruning criterion with the recommended threshold of 1 from [30] is also inconsistent, resulting in large drops in performance for 3 out of 6 settings. BMRS$_{\mathcal{N}}$ and BMRS$_{\mathcal{U}}$ result in both high compression rate and high performance in all settings. BMRS$_{\mathcal{N}}$ accomplishes this without the need for tuning any pruning thresholds, in one case yielding a higher compression rate than BMRS$_{\mathcal{U}}$-4 while keeping the accuracy high. BMRS$_{\mathcal{U}}$-4 results in both the highest compression rate among the three BMRS variants in 4 out of 6 settings, and the highest accuracy in 5 out of 6 settings, with the caveat of needing to select p_1 as a hyperparameter. To explore the effect of this hyperparameter further, we plot accuracy and compression vs. p_1 for Lenet5 trained on CIFAR10 in Figure 4. We see that compression rapidly increases after $p_1 = 11$, continuing until $p_1 = 1$. Additionally, we find that accuracy also steadily increases with a higher compression rate, indicating that BMRS$_{\mathcal{U}}$ reduces complexity while increasing the generalization capacity of the model.

Finally, a comparison of ResNet-50 and ViT on CIFAR10 and TinyImagenet is given in Table 2. Here, SNR and the three BMRS variants achieve similar accuracies at different compression rates. BMRS$_{\mathcal{N}}$ achieves a modest compression rate compared to SNR with a threshold of 1 for each case. BMRS$_{\mathcal{U}}$-4 yields a higher compression rate than BMRS$_{\mathcal{N}}$ and BMRS$_{\mathcal{U}}$-8 in all settings. As such, we show that BMRS$_{\mathcal{N}}$ is capable of achieving high compression with no threshold tuning, while a more extreme compression rate is possible by selecting p_1 for BMRS$_{\mathcal{U}}$.

6 Discussion and conclusion

Our experimental results demonstrate the pruning characteristics of BMRS in two settings: continuous pruning and post-training pruning. One of the benefits of BMRS, and BMRS$_{\mathcal{N}}$ in particular, is that no threshold tuning is needed, making it particularly useful for the continuous pruning setting. Here, the compression rate improves as the model converges, allowing one to gradually increase the compression rate without sacrificing accuracy. The choice of BMRS$_{\mathcal{N}}$ vs. BMRS$_{\mathcal{U}}$ is then dependent on the problem, where more complex scenarios (e.g., over-parameterized models) are suitable for BMRS$_{\mathcal{U}}$ if a higher compression rate is desired. In this case, a hyperparameter search over p_1 in the range $[\epsilon, 23]$ can be done to select BMRS$_{\mathcal{U}}$ (see Figure 4 for an example), or one can simply use BMRS$_{\mathcal{N}}$ to achieve good compression with no hyperparameter tuning. Additionally, we expect that the more over-specified the model is, the lower we can set p_1 and maintain accuracy (i.e., lower bit-rate).

https://doi.org/10.52202/079017-2045

Table 2: Parameter compression % and accuracy for different baseline methods and settings of the proposed method. Standard deviations over three runs are included. Best accuracy for the compression methods is given in bold.

	CIFAR10		TinyImagenet	
Pruning Method	**Comp. %**	**Acc.**	**Comp. %**	**Acc.**
	Res50-Pretrained			
None	0.00 ± 0.00	90.65 ± 0.05	0.00 ± 0.00	53.01 ± 0.35
L2	63.79 ± 4.21	10.00 ± 0.00	55.71 ± 1.08	0.50 ± 0.00
$E[\theta]$	89.85 ± 0.09	16.27 ± 1.86	83.59 ± 1.50	21.11 ± 6.47
SNR	91.73 ± 0.16	89.24 ± 0.40	77.85 ± 0.14	50.54 ± 0.59
BMRS$_{\mathcal{N}}$	87.10 ± 1.55	$\mathbf{89.62 \pm 0.41}$	74.98 ± 0.16	50.56 ± 0.33
BMRS$_{\mathcal{U}}$-8	88.18 ± 0.22	89.29 ± 0.20	74.99 ± 0.04	$\mathbf{50.84 \pm 0.34}$
BMRS$_{\mathcal{U}}$-4	89.85 ± 0.05	89.26 ± 0.28	76.12 ± 0.13	50.82 ± 0.50
	Vision Transformer			
None	0.00 ± 0.00	94.80 ± 0.17	0.00 ± 0.00	63.14 ± 0.42
L2	57.74 ± 0.26	54.36 ± 1.84	47.47 ± 0.12	7.08 ± 0.57
$E[\theta]$	68.18 ± 0.09	10.00 ± 0.00	57.08 ± 0.18	0.50 ± 0.00
SNR	73.03 ± 0.03	94.78 ± 0.10	62.75 ± 0.04	64.60 ± 0.07
BMRS$_{\mathcal{N}}$	57.74 ± 0.25	94.60 ± 0.01	47.48 ± 0.12	65.00 ± 0.21
BMRS$_{\mathcal{U}}$-8	58.14 ± 0.11	94.69 ± 0.04	47.34 ± 0.14	$\mathbf{65.14 \pm 0.13}$
BMRS$_{\mathcal{U}}$-4	67.16 ± 0.21	$\mathbf{94.83 \pm 0.25}$	56.13 ± 0.14	65.13 ± 0.10

Limitations: We note a few of the limitations of BMRS. First, while multiplicative noise pruning allows for the flexible application of pruning at different structural levels, BNNs may offer more aggressive compression rates as they apply sparsity inducing priors at multiple hierarchical levels [2, 15, 16, 33, 29]. BMR based approaches may be derived for such networks; as of this work and as far we we know, it has only been successfully applied in practice to models with flat priors for unstructured pruning [5, 29]. Additionally, multiplicative noise creates an overhead of additional parameters ϕ which increase the training time and storage requirements. Third, more exhaustive baseline pruning criteria could be used for comparison in the future, for example classic methods based on the Hessian or gradient of weights (e.g. see Figure 5 in Appendix E) [23, 12]. Fourth, we apply multiplicative noise to linear layers and convolutional filters, while it could be useful to explore pruning more complex structures in the future. Finally, while structured pruning can reduce the inference time and energy consumption of neural networks, improvements in efficiency have been shown to have potential negative consequences in terms of energy consumption and carbon emissions based on how efficiency can affect how a model is used in practice [35].

Conclusions: In this work we presented BMRS, an efficiently calculable method for threshold-free structured pruning of neural networks. We derived two versions of BMRS: BMRS$_{\mathcal{N}}$ based on the truncated log-normal prior, and BMRS$_{\mathcal{U}}$ based on a reduced truncated log-uniform prior. BMRS offers several key features over existing work: by basing the method off of the approach of multiplicative noise [30], the structured pruning aspect is flexible as it is not dependent on assuming any prior over individual weights and can be easily applied at any structural level [16, 2]. Additionally, the prior and variational posterior in the multiplicative noise approach lend themselves to the derivation of BMRS using multiple reduced priors which have different pruning properties, allowing for flexibility in the compression rate when desired and threshold free pruning otherwise. Finally, our experimental results demonstrate the competitive compression and accuracy of BMRS compared to baseline compression methods on multiple networks of varying complexity and across multiple datasets. The methods presented here based on BMR could form a template for developing more aggressive pruning schemes by incorporating more complex hierarchical priors on both structures and individual weights, as well as for studying limits on the number of neural network structures required to solve a given dataset.

Acknowledgments DW, CI and RS are partly funded by European Union's Horizon Europe Research and Innovation Programme under grant agreements No. 101070284 and No. 101070408. DW is also partly supported by a Danish Data Science Academy postdoctoral fellowship (grant: 2023-1425).

References

[1] L. F. W. Anthony, B. Kanding, and R. Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. In *ICML Workshop on Challenges in Deploying and Monitoring Machine Learning Systems*, 2020.

[2] J. Bai, Q. Song, and G. Cheng. Efficient variational inference for sparse deep learning with theoretical guarantee. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] B. R. Bartoldson, B. Kailkhura, and D. Blalock. Compute-efficient deep learning: Algorithmic trends and opportunities. *Journal of Machine Learning Research*, 24:122–1, 2023.

[4] M. J. Beal. *Variational algorithms for approximate Bayesian inference*. University of London, University College London (United Kingdom), 2003.

[5] J. Beckers, B. Van Erp, Z. Zhao, K. Kondrashov, and B. De Vries. Principled pruning of bayesian neural networks through variational free energy minimization. *IEEE Open Journal of Signal Processing*, 2024.

[6] K. Friston, T. Parr, and P. Zeidman. Bayesian model reduction. arXiv:1805.07092 [stat.ME], 2018.

[7] K. J. Friston and W. D. Penny. Post hoc Bayesian model selection. *NeuroImage*, 56(4): 2089–2099, 2011.

[8] K. J. Friston, V. Litvak, A. Oswal, A. Razi, K. E. Stephan, B. C. M. van Wijk, G. Ziegler, and P. Zeidman. Bayesian model reduction and empirical bayes for group (DCM) studies. *NeuroImage*, 128:413–431, 2016.

[9] K. J. Friston, M. Lin, C. D. Frith, G. Pezzulo, J. A. Hobson, and S. Ondobaka. Active inference, curiosity and insight. *Neural Computation*, 29(10):2633–2683, 2017.

[10] S. Ghosh, J. Yao, and F. Doshi-Velez. Model selection in bayesian neural networks via horseshoe priors. *Journal of Machine Learning Research*, 20:182:1–182:46, 2019.

[11] R. W. Hamming. On the distribution of numbers. *The Bell System Technical Journal*, 49(8): 1609–1625, 1970.

[12] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 164–171. Morgan Kaufmann, 1992.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[14] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau. Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. *Journal of Machine Learning Research*, 21(1):10039–10081, 2020.

[15] A. Hubin and G. Storvik. Variational inference for Bayesian neural networks under model and parameter uncertainty. arXiv:2305.00934 [stat.ML], 2023.

[16] S. R. Jantre, S. Bhattacharya, and T. Maiti. Layer adaptive node selection in bayesian neural networks: Statistical guarantees and implementation details. *Neural Networks*, 167:309–330, 2023.

[17] Y. Jeon and J. Kim. Constructing fast network through deconstruction of convolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5955–5965, 2018.

[18] S. J. Kiebel, M. I. Garrido, R. J. Moran, and K. J. Friston. Dynamic causal modelling for EEG and MEG. *Cognitive Neurodynamics*, 2:121–136, 2008.

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representation (ICLR)*, 2014.

 https://doi.org/10.52202/079017-2045

[20] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2575–2583, 2015.

[21] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[22] Y. LeCun. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`, 1998.

[23] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 598–605. Morgan Kaufmann, 1989.

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

[25] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representation (ICLR)*, 2017.

[26] J. Li, Z. Miao, Q. Qiu, and R. Zhang. Training bayesian neural networks with sparse subspace variational inference. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024.

[27] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[28] C. Louizos, K. Ullrich, and M. Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3288–3298, 2017.

[29] D. Markovic, K. J. Friston, and S. J. Kiebel. Bayesian sparsification for deep neural networks with Bayesian model reduction. arXiv:2309.12095 [stat.ML], 2023.

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

[31] H. Peng, Q. Cao, J. Dodge, M. E. Peters, J. Fernandez, T. Sherborne, K. Lo, S. Skjonsberg, E. Strubell, D. Plessas, I. B. end Evan Pete Walsh, N. A. Smith, and H. Hajishirzi. Efficiency Pentathlon: A standardized arena for efficiency evaluation. arXiv:2307.09701 [cs.CL], 2023.

[32] P. Sedgwick. Spearman's rank correlation coefficient. *BMJ*, 349, 2014.

[33] Y. Sun, Q. Song, and F. Liang. Learning sparse deep neural networks with a spike-and-slab prior. *Statistics & Probability Letters*, 180:109246, 2022.

[34] N. Thompson, K. Greenewald, K. Lee, and G. F. Manso. The Computational Limits of Deep Learning. In *Workshop on Computing within Limits*, 2023.

[35] D. Wright, C. Igel, G. Samuel, and R. Selvan. Efficiency is not enough: A critical perspective of environmentally sustainable AI. arXiv:2309.02065 [cs.LG], 2023.

[36] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, Z. Yan, M. Tomizuka, J. Gonzalez, K. Keutzer, and P. Vajda. Visual transformers: Token-based image representation and processing for computer vision. arXiv:2006.03677 [cs.CV], 2020.

[37] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747 [cs.LG], 2017.

A Derivations

We use the following notation and distributions:

$$\Phi(x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \quad \text{(CDF of } \mathcal{N}(0,1) \text{ evaluated at } x)$$

$$\alpha_p = \frac{a - \mu_p}{\sigma_p}$$

$$\beta_p = \frac{b - \mu_p}{\sigma_p}$$

$$Z_p = \Phi(\beta_p) - \Phi(\alpha_p)$$

$$\text{LogN}_{[a,b]}(\theta|\mu_p, \sigma_p^2) = \begin{cases} \frac{1}{Z_p \theta \sqrt{2\pi\sigma_p^2}} \exp\left\{-\frac{1}{2}\frac{(\log\theta - \mu_p)^2}{\sigma_p^2}\right\} & a \le \theta \le b \\ 0, & \text{otherwise} \end{cases}$$

$$\text{LogU}_{[a,b]}(\theta) = \begin{cases} \left(\theta \log\frac{b}{a}\right)^{-1}, & a \le \theta \le b \\ 0, & \text{otherwise} \end{cases}$$

$$q_\phi(\theta) = \text{LogN}_{[a,b]}(\theta|\mu_q, \sigma_q^2)$$

$$\tilde{p}(\theta) = \text{LogN}_{[a,b]}(\theta|\tilde{\mu}_p, \tilde{\sigma}_p^2)$$

$$p(\theta) = \text{LogU}_{[a,b]}(\theta)$$

A.1 BMRS$_\mathcal{N}$

We start by finding $q_\phi(\theta)\frac{\tilde{p}(\theta)}{p(\theta)}$.

1. First we look at

$$q_\phi(\theta)\tilde{p}(\theta) = \frac{1}{\theta^2 2\pi Z_q \tilde{Z}_p \sqrt{\sigma_q^2 \tilde{\sigma}_p^2}} \exp\left\{-\frac{1}{2}\left(\frac{(\log\theta - \mu_q)^2}{\sigma_q^2} + \frac{(\log\theta - \tilde{\mu}_p)^2}{\tilde{\sigma}_p^2}\right)\right\}.$$

The exponent can be rewritten as

$$-\frac{1}{2}\left(\frac{(\log\theta - \mu_q)^2}{\sigma_q^2} + \frac{(\log\theta - \tilde{\mu}_p)^2}{\tilde{\sigma}_p^2}\right) = -\frac{1}{2}\frac{\tilde{\sigma}_p^2(\log^2\theta - 2\log\theta\mu_q + \mu_q^2) + \sigma_q^2(\log^2\theta - 2\log\theta\tilde{\mu}_p + \tilde{\mu}_p^2)}{\sigma_q^2\tilde{\sigma}_p^2} =$$

$$-\frac{1}{2}\frac{(\sigma_q^2 + \tilde{\sigma}_p^2)\left(\log\theta - \frac{\sigma_q^2\tilde{\sigma}_p^2}{\sigma_q^2+\tilde{\sigma}_p^2}\left(\frac{\mu_q}{\sigma_q^2} + \frac{\tilde{\mu_p}}{\tilde{\sigma}_p^2}\right)\right)^2 + \tilde{\sigma}_p^2\mu_q^2 + \sigma_q^2\tilde{\mu}_p^2 - (\sigma_q^2 + \tilde{\sigma}_p^2)\left(\frac{\sigma_q^2\tilde{\sigma}_p^2}{\sigma_q^2+\tilde{\sigma}_p^2}\left(\frac{\mu_q}{\sigma_q^2} + \frac{\tilde{\mu}_p}{\tilde{\sigma}_p^2}\right)\right)^2}{\sigma_q^2\tilde{\sigma}_p^2}.$$

Defining

$$\tilde{\sigma}_q^2 := \frac{\sigma_q^2\tilde{\sigma}_p^2}{\sigma_q^2 + \tilde{\sigma}_p^2} = \left(\frac{1}{\sigma_q^2} + \frac{1}{\tilde{\sigma}_p^2}\right)^{-1} \text{ and } \tilde{\mu}_q := \tilde{\sigma}_q^2\left(\frac{\mu_q}{\sigma_q^2} + \frac{\tilde{\mu}_p}{\tilde{\sigma}_p^2}\right)$$

we get

$$q_\phi(\theta)\tilde{p}(\theta) = \frac{1}{\theta^2 2\pi Z_q \tilde{Z}_p \sqrt{\sigma_q^2\tilde{\sigma}_p^2}} \exp\left\{-\frac{1}{2}\frac{(\log\theta - \tilde{\mu}_q)^2}{\tilde{\sigma}_q^2}\right\} \exp\left\{-\frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2} + \frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2} - \frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right)\right\}$$

$$= \frac{\theta\tilde{Z}_q\sqrt{2\pi\tilde{\sigma}_q^2}}{\theta^2 2\pi Z_q \tilde{Z}_p \sqrt{\sigma_q^2\tilde{\sigma}_p^2}} \frac{1}{\theta\tilde{Z}_q\sqrt{2\pi\tilde{\sigma}_q^2}} \exp\left\{-\frac{1}{2}\frac{(\log\theta - \tilde{\mu}_q)^2}{\tilde{\sigma}_q^2}\right\} \exp\left\{-\frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2} + \frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2} - \frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right)\right\}$$

$$= \frac{\tilde{Z}_q\sqrt{\tilde{\sigma}_q^2}}{\theta Z_q \tilde{Z}_p \sqrt{2\pi\sigma_q^2\tilde{\sigma}_p^2}} \exp\left\{-\frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2} + \frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2} - \frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right)\right\} \text{LogN}_{[a,b]}(\theta|\tilde{\mu}_q, \tilde{\sigma}_q^2).$$

https://doi.org/10.52202/079017-2045

2. Then we divide out $p(\theta)$:

$$q_\phi(\theta)\frac{\tilde{p}(\theta)}{p(\theta)} = \frac{(\log b - \log a)\tilde{Z}_q\sqrt{\tilde{\sigma}_q^2}}{Z_q\tilde{Z}_p\sqrt{2\pi\sigma_q^2\tilde{\sigma}_p^2}}\exp\left\{-\frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2}+\frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2}-\frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right)\right\}\text{LogN}_{[a,b]}(\theta|\tilde{\mu}_q,\tilde{\sigma}_q^2)$$

3. Now we can find $\tilde{q}(\theta)$ and ΔF using Equation 6. Start with $\tilde{q}(\theta)$:

$$\begin{aligned}
\tilde{q}(\theta) &= \frac{q_\phi(\theta)\frac{\tilde{p}(\theta)}{p(\theta)}}{\exp\Delta F} \\
&= \frac{q_\phi(\theta)\frac{\tilde{p}(\theta)}{p(\theta)}}{\int q_\phi(\theta)\frac{\tilde{p}(\theta)}{p(\theta)}d\theta} \\
&= \frac{\frac{(\log b-\log a)\tilde{Z}_q\sqrt{\tilde{\sigma}_q^2}}{Z_q\tilde{Z}_p\sqrt{2\pi\sigma_q^2\tilde{\sigma}_p^2}}\exp\left\{-\frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2}+\frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2}-\frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right)\right\}\text{LogN}_{[a,b]}(\theta|\tilde{\mu}_q,\tilde{\sigma}_q^2)}{\int\frac{(\log b-\log a)\tilde{Z}_q\sqrt{\tilde{\sigma}_q^2}}{Z_q\tilde{Z}_p\sqrt{2\pi\sigma_q^2\tilde{\sigma}_p^2}}\exp\left\{-\frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2}+\frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2}-\frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right)\right\}\text{LogN}_{[a,b]}(\theta|\tilde{\mu}_q,\tilde{\sigma}_q^2)d\theta} \\
&= \frac{\frac{(\log b-\log a)\tilde{Z}_q\sqrt{\tilde{\sigma}_q^2}}{Z_q\tilde{Z}_p\sqrt{2\pi\sigma_q^2\tilde{\sigma}_p^2}}\exp\left\{-\frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2}+\frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2}-\frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right)\right\}\text{LogN}_{[a,b]}(\theta|\tilde{\mu}_q,\tilde{\sigma}_q^2)}{\frac{(\log b-\log a)\tilde{Z}_q\sqrt{\tilde{\sigma}_q^2}}{Z_q\tilde{Z}_p\sqrt{2\pi\sigma_q^2\tilde{\sigma}_p^2}}\exp\left\{-\frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2}+\frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2}-\frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right)\right\}\int\text{LogN}_{[a,b]}(\theta|\tilde{\mu}_q,\tilde{\sigma}_q^2)d\theta} \\
&= \text{LogN}_{[a,b]}(\theta|\tilde{\mu}_q,\tilde{\sigma}_q^2)
\end{aligned}$$

4. Finally we get ΔF:

$$\begin{aligned}
\Delta F &= \log\frac{q_\phi(\theta)\frac{\tilde{p}(\theta)}{p(\theta)}}{\tilde{q}(\theta)} \\
&= \log\frac{(\log b-\log a)\tilde{Z}_q\sqrt{\tilde{\sigma}_q^2}}{Z_q\tilde{Z}_p\sqrt{2\pi\sigma_q^2\tilde{\sigma}_p^2}}\exp\left\{-\frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2}+\frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2}-\frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right)\right\} \\
&= \log\frac{\tilde{Z}_q(\log b-\log a)}{Z_q\tilde{Z}_p}+\frac{1}{2}\log\frac{\tilde{\sigma}_q^2}{2\pi\sigma_q^2\tilde{\sigma}_p^2}-\frac{1}{2}\left(\frac{\mu_q^2}{\sigma_q^2}+\frac{\tilde{\mu}_p^2}{\tilde{\sigma}_p^2}-\frac{\tilde{\mu}_q^2}{\tilde{\sigma}_q^2}\right)
\end{aligned}$$

A.2 BMRS$_\mathcal{U}$

The PDF of the reduced truncated log-uniform distribution is given as follows:

$$\tilde{p}(\theta) = \text{LogU}_{[a',b']}(\theta) = \begin{cases}\left(\theta\log\frac{b'}{a'}\right)^{-1}, & a\le a'<b'\le b\\ 0, & \text{otherwise}\end{cases} \tag{14}$$

Using this, we can directly solve the integral under the expectation given in Equation 6 for ΔF

$$\begin{aligned}
\exp\Delta F = \mathbb{E}_{\tilde{p}}\left[\frac{q_\phi(\theta)}{p(\theta)}\right] &= \int_a^b \text{LogU}_{[a',b']}(\theta)\frac{q_\phi(\theta)}{\text{LogU}_{[a,b]}(\theta)}d\theta = \int_a^b\frac{\theta\log\frac{b}{a}}{\theta\log\frac{b'}{a'}}q_\phi(\theta)d\theta \\
&= \int_a^{a'}0d\theta+\int_{a'}^{b'}\frac{\log\frac{b}{a}}{\log\frac{b'}{a'}}q_\phi(\theta)d\theta+\int_{b'}^b 0d\theta = \frac{\log\frac{b}{a}}{\log\frac{b'}{a'}}q_\phi(a'\le\theta_i\le b')
\end{aligned}$$

Plugging this in to Equation 5 where $\exp\Delta F\ge 1$:

$$1\le\frac{\log\frac{b}{a}}{\log\frac{b'}{a'}}q_\phi(a'\le\theta_i\le b')\Rightarrow\frac{\log\frac{b'}{a'}}{\log\frac{b}{a}}\le q_\phi(a'\le\theta\le b') \tag{15}$$

B Dataset details

MNIST MNIST [22] is a classic image classification dataset consisting of 70,000 28x28 black and white images of handwritten digits (10 classes). We use the original 10,000 image test set for testing and split the 60,000 image train set into 80% training and 20% validation images.

Fashion-MNIST Fashion-MNIST [37] is a modernized version of MNIST using images of different articles of clothing as opposed to handwritten digits. The dataset statistics are the same as MNIST: 28x28 greyscale images, 60,000 training images, 10,000 test images, 10 classes. Similar to MNIST, we split the training set into 80% training and 20% validation images.

CIFAR10 CIFAR10 [21] is an image classification dataset of 32x32 color images with 10 classes. There are 50,000 training images and 10,000 test images. Again, we split the training set to 80% training and 20% validation.

TinyImagenet TinyImagenet is a reduced version of ImageNet consisting of 110,000 64x64 color images in 200 classes. We use the 10,000 image validation split for testing, and split the 100,000 image train set into 80% training and 20% validation images.

C Model details

For each model and dataset we use the Adam optimizer with no weight decay. We train for 50 epochs for each experiment with an MLP and Lenet5, and for 100 epochs for each experiment with Resnet50 and ViT. Further details about each model are given as follows:

C.1 MLP

We use a multilayer perceptron (MLP) for several experiments, with different network sizes based on a hyperparameter sweep for each dataset. Multiplicative noise for pruning is applied to every neuron in the network. We sweep through the following hyperparmeters:

- Number of layers = {1,3,5,7,9}
- Hidden dimension = {10, 30, 50, 100, 150}
- Batch size = {16, 32, 64, 128}
- Learning rate [0.0001, 0.1].

The final network settings for each dataset are given as follows:

MNIST: Number of layers: 7; Hidden dimension: 100; Batch size: 128; Learning rate: $8.5{\cdot}10^{-4}$.

Fashion-MNIST Number of layers: 1; Hidden dimension: 150; Batch size: 128; Learning rate: $1.5{\cdot}10^{-3}$.

CIFAR10 Number of layers: 5; Hidden dimension: 150; Batch size: 32; Learning rate: $6.8{\cdot}10^{-4}$.

C.2 Lenet5

Lenet5 [24] is an early CNN architecture consisting of 2 convolutional layers with 6 and 16 filters per channel, respectively, each followed by a ReLU activation and max pooling layer, followed by 3 linear layers. Multiplicative noise is applied to each convolutional filter map, as well as each neuron the the linear layers. We use the same architecture for each experiments and tune hyperparameters based on the dataset. We sweep through the following hyperparameters:

- Batch size = {16, 32, 64, 128}
- Learning rate [0.0001, 0.1].

The final settings for each dataset are given as follows:

MNIST: Batch size 128; Learning rate $1.4{\cdot}10^{-3}$.

https://doi.org/10.52202/079017-2045

Table 3: Training and inference runtimes in milliseconds. Each runtime is averaged across 1000 forward passes of a batch of 32 images. Inference runtimes are *before* pruning (i.e., with the full network).

	MNIST		Fash-MNIST		CIFAR10	
Pruning Method	Train	Inf.	Train	Inf.	Train	Inf.
MLP						
None	2.41	0.73	2.43	0.73	4.61	5.30
$E[\theta]$	19.76	1.17	20.00	1.14	21.21	5.39
SNR	20.00	1.14	19.99	1.15	22.00	4.94
BMRS$_{\mathcal{N}}$	20.02	1.14	20.14	1.15	21.76	5.57
BMRS$_{\mathcal{U}}$	19.99	1.16	20.11	1.16	22.14	5.15
Lenet5						
None	2.31	0.69	2.24	0.69	4.73	5.15
$E[\theta]$	11.45	0.94	11.34	0.94	12.11	4.90
SNR	11.18	0.93	11.42	0.94	15.05	4.95
BMRS$_{\mathcal{N}}$	11.39	0.94	11.47	0.94	13.34	5.07
BMRS$_{\mathcal{U}}$	11.44	0.94	11.08	0.93	12.86	4.43

Fashion-MNIST Batch size 32; Learning rate $1.4{\cdot}10^{-3}$.

CIFAR10 Batch size 64; Learning rate $1{\cdot}10^{-3}$.

C.3 Resnet50

Resnet50 [13] is a deep 50-layer CNN which uses residual connections to stabilize optimization and improve accuracy. We start with a model pretrained on ImageNet-1k,[3] then fine-tuned on the downstream dataset with pruning layers added to each output layer after batch normalization. We use a learning rate of 6.8e-4, a batch size of 32, and train for 100 epochs for both CIFAR10 and TinyImagenet.

C.4 Vision Transformer (ViT)

Vision Transformer (ViT) [36] is a transformer model tailored for image data based on tokenizing an image as 16x16 image patches. We use a ViT which is pretrained on ImageNet-21k (14M images and 21,843 classes) as well as ImageNet-1k.[4] We add multiplicative noise to the output layer of each transformer block for pruning. We use a learning rate of 6.8e-4, a batch size of 32, and train for 100 epochs for both CIFAR10 and TinyImagenet.

D Compute resources

All experiments were run on a shared cluster. Requested jobs consisted of 16GB of RAM and 4 Intel Xeon Silver 4110 CPUs. We used a single NVIDIA Titan X GPU with 24GB of RAM for all experiments, though utilization was generally much lower due the the average size of each network. Runtimes for each experiment ranged from approx. 7 minutes for Lenet5 on MNIST with no pruning layers to approx. 44 hours for ViT on TinyImagenet with multiplicative noise trained for 100 epochs. The training of models in this work over the course of the entire project (prototyping, experimentation, etc.) is estimated to have used 3773.785 kWh of electricity contributing to 599.892 kg of CO2eq (as measured by carbontracker [1]; this is equivalent to 5580.395 km travelled by car).

We additionally benchmark the training and inference runtimes of each model *before* pruning in Table 3 and Table 4 (i.e., the inference runtimes are without removing structures).

[3] `https://pytorch.org/vision/stable/models.html`

[4] `https://huggingface.co/google/vit-base-patch16-224`

Table 4: Training and inference runtimes in milliseconds. Each runtime is averaged across 1000 forward passes of a batch of 32 images. Inference runtimes are *before* pruning (i.e., with the full network).

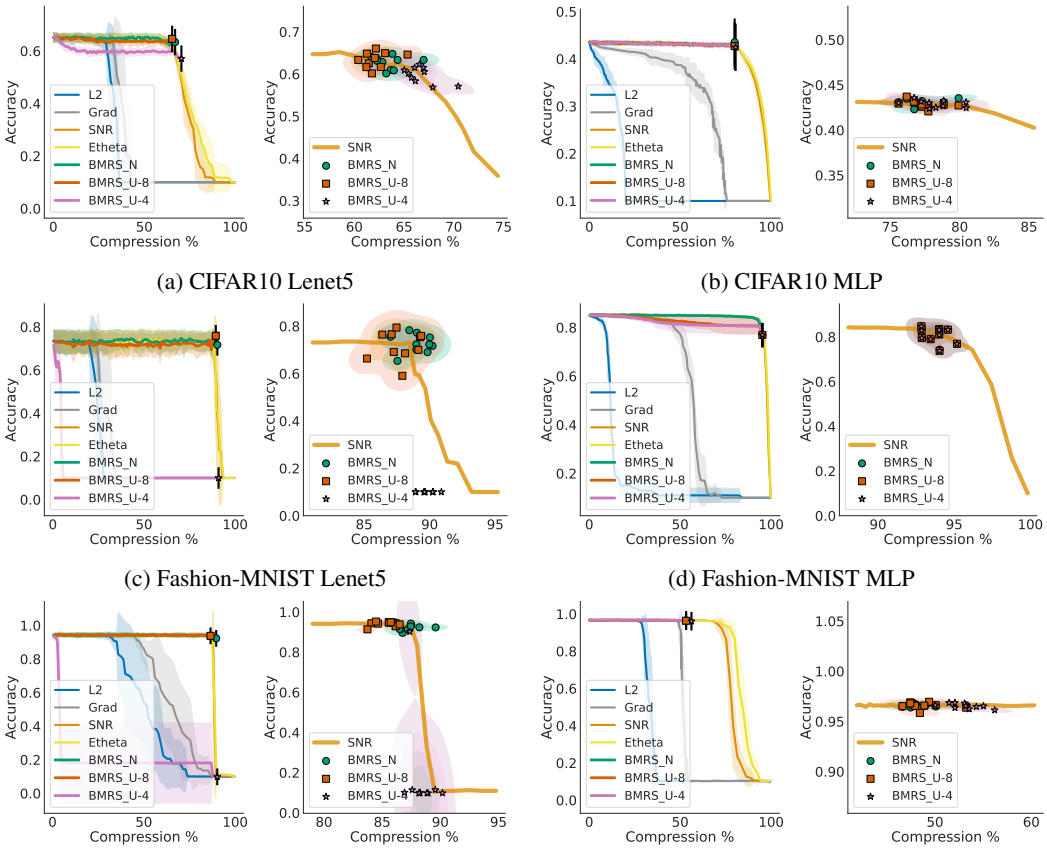
	CIFAR10		TinyImagenet	
Pruning Method	Train	Inf.	Train	Inf.
Res50-Pretrained				
None	47.78	34.30	74.89	59.11
$E[\theta]$	329.11	35.24	346.80	64.02
SNR	328.29	33.97	346.56	64.00
BMRS$_{\mathcal{N}}$	329.84	32.99	345.63	64.01
BMRS$_{\mathcal{U}}$-4	329.81	34.85	346.78	64.06
Vision Transformer				
None	254.50	87.15	252.91	86.70
$E[\theta]$	343.51	88.80	351.19	87.49
SNR	343.50	88.92	350.93	87.55
BMRS$_{\mathcal{N}}$	343.55	88.74	350.99	87.55
BMRS$_{\mathcal{U}}$-4	343.60	88.56	351.39	87.47

E Additional plots

An additional experiment for post-training pruning including gradient-based thersholding for pruning is given in Figure 5. For gradient based pruning, we use two thresholds: one for the L2-Norm (magnitude) of structures, and one for the L2-Norm of the gradients of weights in a given structure.

The neuron rank correlations for Fashion-MNIST are given in Figure 6. and for MNIST in Figure 7.

https://doi.org/10.52202/079017-2045

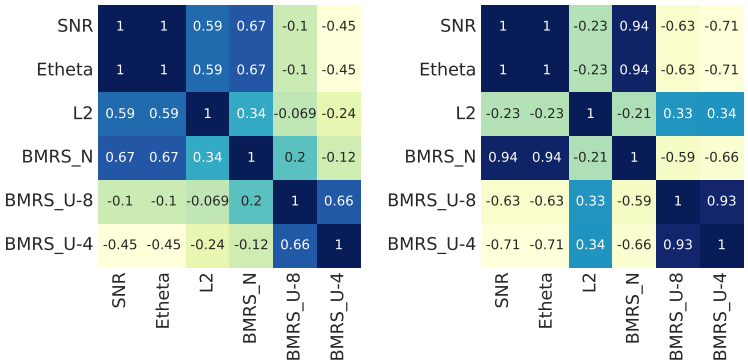


(a) CIFAR10 Lenet5 (b) CIFAR10 MLP

(c) Fashion-MNIST Lenet5 (d) Fashion-MNIST MLP

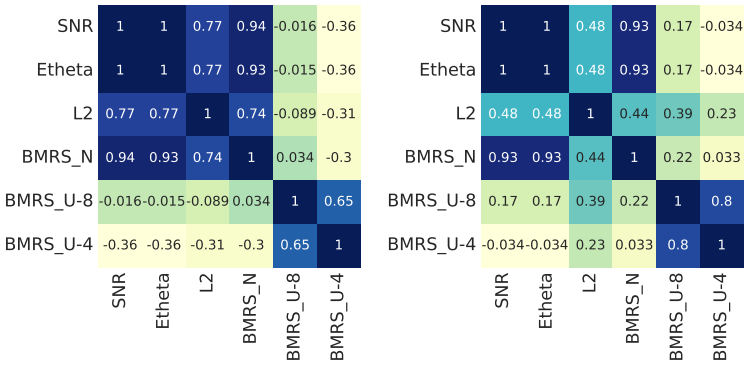
(e) MNIST Lenet5 (f) MNIST MLP

Figure 5: Additional accuracy vs. compression results for post-training pruning including gradient-based pruning.



(a) Fashion-MNIST Lenet5 (b) Fashion-MNIST MLP

Figure 6: Average correlation between the ranks of neurons for pruning when using different methods on Fashion-MNIST.



(a) MNIST Lenet5 (b) MNIST MLP

Figure 7: Average correlation between the ranks of neurons for pruning when using different methods on MNIST.

https://doi.org/10.52202/079017-2045

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract accurately reflects the main claims of the paper; see the last paragraph of §1 for the primary contributions for comparison.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed after §6 in a paragraph marked "Limitations".

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide detailed derivations and describe assumptions for each new theoretical result in §4.2 and Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The full methodology is given in §3 and §4; the experimental setting is given in §5; the full dataset and model details are given in Appendix B and Appendix C, respectively.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

https://doi.org/10.52202/079017-2045

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The datasets we use are all open source, and details are given in Appendix B. The code is made available and can be downloaded/viewed at `https://github.com/saintslab/bmrs-structured-pruning/`.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setting is given in §5; the full dataset and model details are given in Appendix B and Appendix C, respectively.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All of our experiments are run across multiple seeds. All plots include error bars (areas of standard deviation or confidence intervals). All tables include the mean and standard deviation across multiple seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Information on the compute infrastructure and resources are given in Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We conform to the NeurIPS code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We point out in the limitations that efficiency can also have potential negative environmental aspects due to e.g. the rebound effect.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

https://doi.org/10.52202/079017-2045

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: As we propose a general method for neural network pruning, we do not see where safeguards could be put in place or that there is a high risk of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We clearly mark where any non-original code is used (see e.g. modules/utils.py in the code).

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide links to the code in the paper and have documented the code for ease of use/reproducibility.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not perform crowdsourced experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not do crowdsourcing or involve human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

https://doi.org/10.52202/079017-2045

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.