Supplementary Materials for: The k-tied Normal Distribution: A Compact Parameterization of Gaussian Mean Field Posteriors in Bayesian Neural Networks

Jakub Swiatkowski [1 +] **Kevin Roth** [2 +] **Bastiaan S. Veeling** [3 4 +] **Linh Tran** [5 +] **Joshua V. Dillon** [4] **Jasper Snoek** [4] **Stephan Mandt** [6 +] **Tim Salimans** [4] **Rodolphe Jenatton** [4] **Sebastian Nowozin** [7 +]

Abstract

This document contains additional details for the main ICML 2020 paper.

A. Proof of the Matrix Variate Normal Parameterization

In this section of the appendix, we formally explain the connections between the k-tied Normal distribution and the matrix variate Gaussian distribution (Gupta & Nagar, 2018), referred to as $\mathcal{MN}$.

Consider positive definite matrices $\mathbf{Q} \in \mathbb{R}^{r\times r}$ and $\mathbf{P} \in \mathbb{R}^{c\times c}$ and some arbitrary matrix $\mathbf{M} \in \mathbb{R}^{r\times c}$. We have by definition that $\mathbf{W} \in \mathbb{R}^{r\times c} \sim \mathcal{MN}(\mathbf{M}, \mathbf{Q}, \mathbf{P})$ if and only if $\text{vec}(\mathbf{W}) \sim \mathcal{N}(\text{vec}(\mathbf{M}), \mathbf{P} \otimes \mathbf{Q})$, where $\text{vec}(\cdot)$ stacks the columns of a matrix and $\otimes$ is the Kronecker product

The $\mathcal{MN}$ has already been used for variational inference by Louizos & Welling (2016) and Sun et al. (2017). In particular, Louizos & Welling (2016) consider the case where both $\mathbf{P}$ and $\mathbf{Q}$ are restricted to be diagonal matrices. In that case, the resulting distribution corresponds to our k-tied Normal distribution with $k = 1$ since

$$\mathbf{P} \otimes \mathbf{Q} = \text{diag}(\mathbf{p}) \otimes \text{diag}(\mathbf{q}) = \text{diag}(\text{vec}(\mathbf{q}\mathbf{p}^\top)).$$

Importantly, we prove below that, in the case where $k \geq 2$, the k-tied Normal distribution cannot be represented as a matrix variate Gaussian distribution.

Lemma (Rank-2 matrix and Kronecker product)**.** *Let* $\mathbf{B}$ *be a rank-*2 *matrix in* $\mathbb{R}^{r\times c}_+$*. There do not exist matrices* $\mathbf{Q} \in \mathbb{R}^{r\times r}$ *and* $\mathbf{P} \in \mathbb{R}^{c\times c}$ *such that*

$$\textit{diag}(\textit{vec}(\mathbf{B})) = \mathbf{P} \otimes \mathbf{Q}.$$

Proof. Let us introduce the shorthand $\mathbf{D} = \text{diag}(\text{vec}(\mathbf{B}))$. By construction, $\mathbf{D}$ is diagonal and has its diagonal terms strictly positive (it is assumed that $\mathbf{B} \in \mathbb{R}^{r\times c}_+$, i.e., $b_{ij} > 0$ for all i, j).

We proceed by contradiction. Assume there exist $\mathbf{Q} \in \mathbb{R}^{r\times r}$ and $\mathbf{P} \in \mathbb{R}^{c\times c}$ such that $\mathbf{D} = \mathbf{P} \otimes \mathbf{Q}$.

This implies that all diagonal blocks of $\mathbf{P}\otimes\mathbf{Q}$ are themselves diagonal with strictly positive diagonal terms. Thus, $p_{jj}\mathbf{Q}$ is diagonal for all $j \in \{1, \ldots, c\}$, which implies in turn that $\mathbf{Q}$ is diagonal, with non-zero diagonal terms and $p_{jj} \neq 0$. Moreover, since the off-diagonal blocks $p_{ij}\mathbf{Q}$ for $i \neq j$ must be zero and $\mathbf{Q} \neq \mathbf{0}$, we have $p_{ij} = 0$ and $\mathbf{P}$ is also diagonal.

To summarize, if there exist $\mathbf{Q} \in \mathbb{R}^{r\times r}$ and $\mathbf{P} \in \mathbb{R}^{c\times c}$ such that $\mathbf{D} = \mathbf{P} \otimes \mathbf{Q}$, then it holds that $\mathbf{D} = \text{diag}(\mathbf{p}) \otimes \text{diag}(\mathbf{q})$ with $\mathbf{p} \in \mathbb{R}^c$ and $\mathbf{q} \in \mathbb{R}^r$. This last equality can be rewritten as $b_{ij} = p_j q_i$ for all $i \in \{1, \ldots, r\}$ and $j \in \{1, \ldots, c\}$, or equivalently

$$\mathbf{B} = \mathbf{q}\mathbf{p}^\top.$$

This leads to a contradiction since $\mathbf{q}\mathbf{p}^\top$ has rank one while $\mathbf{B}$ is assumed to have rank two. □

Figure 1 provides an illustration of the difference between the k-tied Normal and the $\mathcal{MN}$ distribution.

B. He-scaled Normal Prior

We investigate whether the low-rank structure is specific to the GMFVI neural networks that use a Normal prior with a single scalar scale for all the weights. Instead of using the single scale parameter, we analyse a setting in which the Normal prior scale is set according to the scaling rules devised for neural network weights initialization (Glorot & Bengio, 2010; He et al., 2015). According to these rules, a per layer scale parameter is set according to the layer shape and activation function used. In particular, we use the scaling rule from He et al. (2015) for the models with ReLU

[+]Work done while at Google [1]University of Warsaw [2]ETH Zurich [3]University of Amsterdam [4]Google Research [5]Imperial College London [6]University of California, Irvine [7]Microsoft Research. Correspondence to: Jakub Swiatkowski <jakub.swiatkowski@mimuw.edu.pl>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

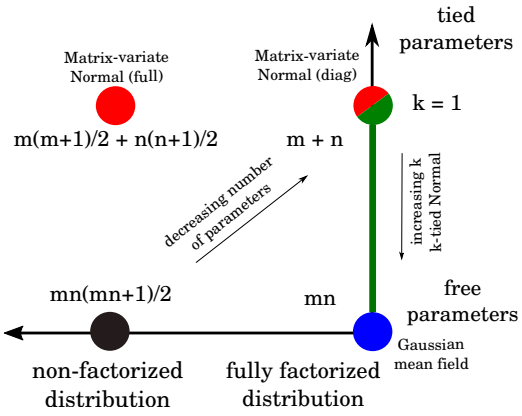


Figure 1. Illustration of the difference in modeling of the posterior covariance by the k-tied Normal distribution (green), the $\mathcal{MN}$ distribution (red), the Gaussian mean field (blue) and the dense Gaussian covariance (black) for a layer of size $m \times n$. The k-tied Normal with $k = 1$ is equivalent to $\mathcal{MN}$ with diagonal row and column covariance matrices (half-red, half-green circle). Our experiments show that the $k = 1$ fails to capture the performance of the mean field. On the other hand, while the full/non-diagonal $\mathcal{MN}$ increases the expressiveness of the posterior, it also increases the number of parameters. In contrast, the k-tied Normal distribution with $k \geq 2$ not only decreases the number of parameters, but also matches the predictive performance of the mean field.

activations (Glorot et al., 2011):

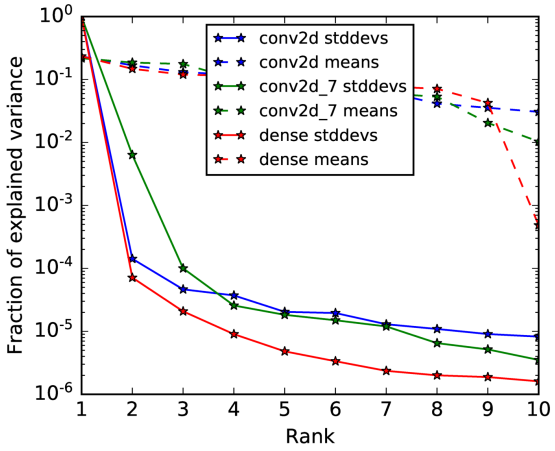
$$p(\mathbf{w}_l) = \mathcal{N}\left(0, \frac{2}{m_l}\right), \tag{1}$$

where m_l is the *fan-in* of the m'th layer.[1] However, the scaling rule proposed in He et al. (2015) does not cover the bias terms, which are initialized at zero. Therefore, for the ResNet-18 on CIFAR-10 which we take under test, we keep the prior for the biases unchanged at $\mathcal{N}(0, I)$. We rerun then the low-rank structure experiments from Section 2.3 Figure 5, but now with the He-scaled prior. Figure 2 shows the low-rank structure analysis results for the new prior. While we observe an overall drop in performance, the low-rank structure clearly remains present.

C. KL Annealing with Adam

We verify that when using KL annealing with Adam the posterior standard deviation parameters do not converge prematurely, but rather continue being optimized after the KL is at its full contribution. Figure 3 illustrates this on the example of the ResNet-18 CIFAR-10 model trained the standard GMFVI. Furthermore, for the MLP, CNN and LSTM models, we observed their posterior standard deviations at convergence to have large values compared to the prior standard deviation value (>50% of the prior value), showing that we are modeling substantial uncertainty.

[1]For a dense layer the fan-in is the number of input dimensions, for a 2D Convolutional layer with a kernel of size $k \times k$ and d input channels the fan-in is $m_l = k^2 d$.



Method	-ELBO ↓	NLL ↓	Accuracy ↑
Mean-field	1.379±0.0096	0.6384±0.0096	79.0±0.41
1-tied	5.428±0.018	1.485±0.0056	57.0±0.50
2-tied	1.448±0.0097	0.648±0.0079	78.8±0.41
3-tied	1.411±0.0097	0.646±0.0079	78.9±0.41

Figure 2. The post-training low rank structure is still present in the posterior standard deviation parameters of the ELBO-converged standard GMFVI ResNet-18 CIFAR-10 model when using the He-scaled prior. Approximations to these parameters with ranks higher than 1 result in performance close to that when not using the approximation. We report mean and SEM for predictions made using an ensemble of 100 weights samples. The SEM is measured across the test examples.

D. Experimental Details

In this section we provide additional information on the experimental setup used in the main paper. In particular, we describe the details of the models and datasets, the utilized standard Gaussian Mean Field Variational Inference (GMFVI) training procedure, the low-rank structure analysis of the GMFVI trained posteriors and the proposed k-tied Normal posterior training procedure.

D.1. Models and datasets

To confirm the validity of our results, we performe the experiments on a range of models and datasets with different data types, architecture types and sizes. Below we describe their details.

MLP MNIST Multilayer perceptron (MLP) model with three dense layers and ReLu activations trained on the MNIST dataset (LeCun & Cortes, 2010). The three layers have sizes of 400, 400 and 10 hidden units. We preprocess

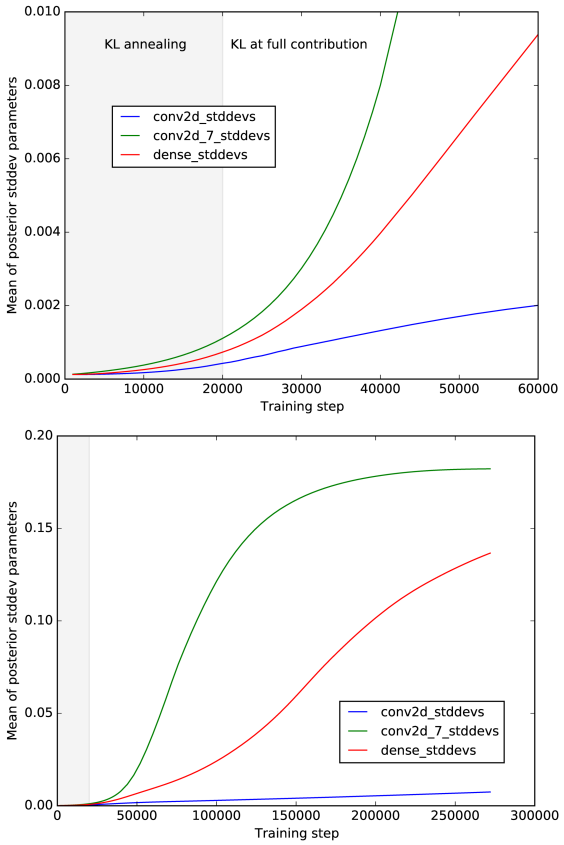


Figure 3. Change in the mean of posterior standard deviation parameters for selected layers of the standard GMFVI ResNet-18 CIFAR-10 model over the course of training. KL is annealed over the first 50 epochs linearly from 0 to 1 (gray area). Top: posterior standard deviation parameters continue being optimized when the KL is at its full contribution. Bottom: the posterior standard deviations reach large values after 700 epochs showing that we are modeling substantial uncertainty.

the images to be have values in range $[-1, 1]$. We use the last 10,000 examples of the training set as a validation set.

LeNet CNN CIFAR-100 LeNet convolutional neural network (CNN) model (LeCun et al., 1998) with two convolutional layers followed by two dense layers, all interleaved with ReLu activations. The two convolutional layers have 32 and 64 output filters respectively, each produced by kernels of size 3×3. The two dense layers have sizes of 512 and 100 hidden units. We train this network on the CIFAR-100 dataset (Krizhevsky et al., 2009). We preprocess the images to have values in range $[0, 1]$. We use the last 10,000 examples of the training set as a validation set.

LSTM IMDB Long short-term memory (LSTM) model (Hochreiter & Schmidhuber, 1997) that consists of an embedding and an LSTM cell, followed by a dense layer with a single unit. The LSTM cell consists of two dense weight matrices, namely the kernel and the recurrent kernel. The embedding and the LSTM cell have both 128-dimensional output space. More precisely, we adopt the publicly available LSTM Keras (Chollet et al., 2015) example[2], except that we set the dropout rate to zero. We train this model on the IMDB text sentiment classification dataset (Maas et al., 2011), in which we use the last 5,000 examples of the training set as a validation set.

ResNet-18 CIFAR-10 ResNet-18 model (He et al., 2016) trained on the CIFAR-10 dataset (Krizhevsky et al., 2009). We adopt the ResNet-18 implementation[3] from the Tensorflow Probability (Dillon et al., 2017) repository. We train/evaluate this model on the train/test split of 50,000 and 10,000 images, respectively, from the CIFAR-10 dataset available in Tensorflow Datasets[4].

D.2. GMFVI training

We train all the above models using GMFVI. We split the discussion of the details of the GMFVI training procedure into two parts. First, we describe the setup for the MLP, CNN and LSTM models, for which we prepare our own GMFVI implementations. Second, we explain the setup for the GMFVI training of the ResNet-18 model, for which we use the implementation available in the Tensorflow Probability repository as mentioned above.

MLP, CNN and LSTM In the MLP and the CNN models, we approximate the posterior using GMFVI for all the weights (both kernel and bias weights). For the LSTM model, we approximate the posterior using GMFVI only for the kernel weights, while for the bias weights we use a point estimate. For all the three models, we use the standard reparametrization trick estimator (Kingma & Welling, 2013). We initialize the GMFVI posterior means using the standard He initialization (He et al., 2015) and the GMFVI posterior standard deviations using samples from $\mathcal{N}(0.01, 0.001)$. Furthermore, we use a Normal prior $\mathcal{N}(0, \sigma_p I)$ with a single scalar standard deviation hyper-parameter σ_p for all the layers. We select σ_p for each of the models separately from a set of $\{0.2, 0.3\}$ based on the validation data set performance.

We optimize the variational parameters using an Adam optimizer (Kingma & Ba, 2014). We pick the optimal learning rate for each model from the set of $\{0.0001, 0.0003, 0.001, 0.003\}$ also based on the validation data set performance. We choose the batch size of 1024 for the MLP and CNN models, and the batch size of 128 for

[2]See: https://github.com/keras-team/keras/blob/master/examples/imdb_lstm.py.

[3]See: https://github.com/tensorflow/probability/blob/master/tensorflow_probability/examples/cifar10_bnn.py.

[4]See: https://www.tensorflow.org/datasets/catalog/cifar10.

the LSTM model. We train all the models until the ELBO convergence.

To implement the MLP and CNN models we use the `tfp.layers` module from the Tensorflow Probability, while to implement the LSTM model we use the `LSTMCellReparameterization`[5] class from the Edward2 Layers module (Tran et al., 2019).

ResNet-18 The specific details of the GMFVI training of the ResNet-18 model can be found in the previously linked implementation from the Tensorflow Probability repository. Here, we describe the most important and distinctive aspects of this implementation.

The ResNet-18 model approximates the posterior using GMFVI only for the kernel weights, while for the bias weights it uses a point estimate. The model uses the Flipout estimator (Wen et al., 2018) and a constraint on the maximum value of the GMFVI posterior standard deviations of 0.2. The GMFVI posterior means are initialized using samples from $N(0, 0.1)$, while the GMFVI posterior log standard deviations are initialized using samples from $\mathcal{N}(-9.0, 0.1)$. Furthermore, the model uses a Normal prior $\mathcal{N}(0, I)$ for all of its layers.

The variational parameters are trained using the Adam optimizer with a learning rate of 0.0001 and a batch size of 128. The model is trained for 700 epochs. The contribution of the D_{KL} term in the negative Evidence Lower Bound (ELBO) equation is *annealed* linearly from zero to its full contribution over the first 50 epochs (Sønderby et al., 2016).

D.3. Low-rank structure analysis

After training the above models using GMFVI, we investigate the low-rank structure in their trained variational posteriors. For the MLP, CNN and LSTM models, we investigate the low-rank structure of their dense layers only. For the ResNet-18 model, we investigate both its dense and convolutional layers.

To investigate the low-rank structure in the GMFVI posterior of a dense layer, we inspect a spectrum of the posterior mean and standard deviation matrices. In particular, for both the posterior mean and standard deviation matrices, we consider the fraction of the variance explained by the top singular values from their SVD decomposition (see Figure 3 in the main paper). Furthermore, we explore the impact on predictive performance of approximating the reshaped diagonal matrices with their low-rank approximations using only the components corresponding to the top singular values (see Table 2 in the main paper). Note that such low-rank approximations may contain values below zero. This has to be addressed when approximating the matrices of the posterior standard deviations, which can contain only positive values. Therefore, we use a lower bound of zero for the values of the approximations to the posterior standard deviations.

To investigate the low-rank structure in a GMFVI posterior of a convolutional layer, we need to add a few more steps compared to those for a dense layer. In particular, weights of the convolutional layers considered here are 4-dimensional, instead of 2-dimensional as in the dense layer. Therefore, before performing the SVD decomposition, as for the dense layers, we first reshape the 4-dimensional weight tensor from the convolutional layer into a 2-dimensional weight matrix. More precisely, we flatten all dimensions of the weight tensor except for the last dimension (e.g., a weight tensor of shape $[3, 3, 512, 512]$ is reshaped to $[3 \cdot 3 \cdot 512, 512]$). Figure 4 contains example visualizations of the resulting flattened 2-dimensional matrices[6]. Given the 2-dimensional form of the weight tensor, we can investigate the low-rank structure in the convolutional layers as for the dense layers. As noted already in Figure 5 in the main paper, we observe the same strong low-rank structure behaviour in the flattened convolutional layers as in the dense layers. Interestingly, the low-rank structure is the most visible in the final convolutional layers, which also contain the highest number of parameters, see Figure 5.

Importantly, note that after performing the low-rank approximation in this 2-dimensional space, we can reshape the resulting 2-dimensional low-rank matrices back into the 4-dimensional form of a convolutional layer. Table 1 shows that such a low-rank approximation of the convolutional layers of the analyzed ResNet-18 model can be performed without a loss in the model's predictive performance, while significantly reducing the total number of model parameters.

D.4. k-tied Normal posterior training

To exploit the low-rank structure observation, we propose the k-tied Normal posterior, as discussed in Section 3. We study the properties of the k-tied Normal posterior applied to the MLP, CNN and LSTM models. We use the k-tied Normal variational posterior for all the dense layers of the analyzed models. Namely, we use the k-tied Normal variational posterior for all the three layers of the MLP model, for the two dense layers of the CNN model and for the LSTM cell's kernel and recurrent kernel.

We initialize the parameters u_{ik} and v_{jk} of the k-tied Normal distribution so that after the outer-product operation

[5] See: https://github.com/google/edward2/blob/master/edward2/tensorflow/layers/recurrent.py.

[6] After this specific reshape operation, all the weights corresponding to a single output filter are contained in a single column of the resulting weight matrix.

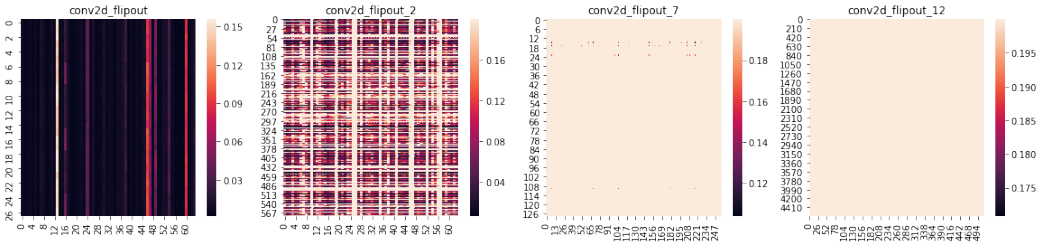


Figure 4. Heat maps of the partially flattened posterior standard deviation tensors for the selected convolutional layers of the ResNet-18 GMFVI BNN trained on CIFAR-10. The partially flattened posterior standard deviation tensors of the convolutional layers display similar low-rank patterns that we observe for the dense layers.

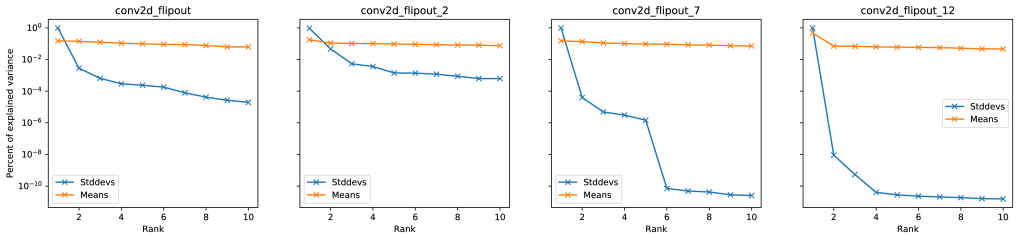


Figure 5. Fraction of variance explained per each singular value from SVD of partially flattened tensors of posterior means and posterior standard deviations for different convolutional layers of the ResNet-18 GMFVI BNN trained on CIFAR-10. Posterior standard deviations clearly display strong low-rank structure, with most of the variance contained in the top few singular values, while this is not the case for posterior means. Interestingly, the low-rank structure is the most visible for the final convolutional layers, which also contain the highest number of parameters.

the respective standard deviations σ_{ij} have the same mean values as we obtain when using the standard GMFVI posterior parametrization. More precisely, we initialize the parameters u_{ik} and v_{jk} so that after the outer-product operation the respective σ_{ij} standard deviations have means at 0.01 before transforming to log-domain. This means that in the log domain the parameters u_{ik} and v_{jk} are initialized as $0.5(\log(0.01) - \log(k))$. We also add white noise $\mathcal{N}(0, 0.1)$ to the values of u_{ik} and v_{jk} in the log domain to break symmetry.

During training of the models with the k-tied Normal posterior, we linearly anneal the contribution of the D_{KL} term of the ELBO loss. We select the best linear coefficient for the annealing from $\{5 \times 10^{-5}, 5 \times 10^{-6}\}$ (per batch) and increase the effective contribution every 100 batches in a step-wise manner. In particular, we anneal the D_{KL} term to obtain the predictive performance results for all the models in Figure 6 in the main paper. However, we do not perform the annealing in the Signal-to-Noise ratio (SNR) and negative ELBO convergence speed experiments in the same Figure 6. In these two cases, KL annealing would occlude the values of interest, which show the clear impact of the k-tied Normal posterior.

References

Chollet, F. et al. Keras, 2015.

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. *arXiv preprint arXiv:1711.10604*, 2017.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

Gupta, A. K. and Nagar, D. K. *Matrix variate distributions*. Chapman and Hall/CRC, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE*

Method	-ELBO ↓	NLL ↓	Accuracy ↑	#Params ↓	%Params ↓
Mean-field	$122.61_{\pm 0.012}$	$0.495_{\pm 0.0080}$	$83.5_{\pm 0.37}$	9,814,026	100.0
1-tied	$122.57_{\pm 0.012}$	$0.658_{\pm 0.0069}$	$81.7_{\pm 0.39}$	4,929,711	50.2
2-tied	$122.77_{\pm 0.012}$	$0.503_{\pm 0.0080}$	$83.2_{\pm 0.37}$	4,946,964	50.4
3-tied	$122.67_{\pm 0.012}$	$0.501_{\pm 0.0079}$	$83.2_{\pm 0.37}$	4,964,217	50.6

Table 1. Impact of the low-rank approximation of the GMFVI-trained posterior standard deviations of a ResNet-18 model on the model's predictive performance. We report mean and SEM of each metric across 100 weights samples. The low-rank approximations with ranks higher than one achieve predictive performance close to that when not using any approximations, while significantly reducing the number of model parameters.

conference on computer vision and pattern recognition, pp. 770–778, 2016.

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

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

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

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

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

Louizos, C. and Welling, M. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pp. 1708–1716, 2016.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pp. 142–150. Association for Computational Linguistics, 2011.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. How to train deep variational autoencoders and probabilistic ladder networks. In *33rd International Conference on Machine Learning (ICML 2016)*, 2016.

Sun, S., Chen, C., and Carin, L. Learning structured weight uncertainty in bayesian neural networks. In *Artificial Intelligence and Statistics*, pp. 1283–1292, 2017.

Tran, D., Dusenberry, M. W., Hafner, D., and van der Wilk, M. Bayesian Layers: A module for neural network uncertainty. In *Neural Information Processing Systems*, 2019.

Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.