Stochastic Neural Network with Kronecker Flow

Chin-Wei Huang
Mila†

Ahmed Touati
Mila†

Pascal Vincent
Facebook AI Research, Mila, CIFAR

Gintare Karolina Dziugaite
Element AI

Alexandre Lacoste
Element AI

Aaron Courville
Mila, CIFAR Fellow

Abstract

Recent advances in variational inference enable the modelling of highly structured joint distributions, but are limited in their capacity to scale to the high-dimensional setting of stochastic neural networks. This limitation motivates a need for scalable parameterizations of the noise generation process, in a manner that adequately captures the dependencies among the various parameters. In this work, we address this need and present the *Kronecker Flow*, a generalization of the Kronecker product to invertible mappings designed for stochastic neural networks. We apply our method to variational Bayesian neural networks on predictive tasks, PAC-Bayes generalization bound estimation, and approximate Thompson sampling in contextual bandits. In all setups, our methods prove to be competitive with existing methods and better than the baselines.

1 Introduction

Stochastic neural networks (SNN) are a central tool in many subfields of machine learning, including (1) Bayesian deep learning (MacKay, 1992; Blundell et al., 2015; Hernández-Lobato and Adams, 2015; Gal and Ghahramani, 2016), (2) exploration in reinforcement learning (Ian et al., 2013; Osband et al., 2016; Riquelme et al., 2018), and (3) statistical learning theory such as PAC-Bayesian learning (McAllester, 1999; Langford and Seeger, 2001; Dziugaite and Roy, 2017a). Perturbations of the network parameters induce a distribution over the model, and this intrinsic uncertainty is the subject of great interest to machine learning practitioners and theoreticians alike. For example, deep Bayesian models are often used to adequately measure uncertainty, and determine whether the model itself is inherently familiar with the unseen data. This is especially important in the context of autonomous vehicles, where decisions must be made to meet specific safety standards (McAllister et al., 2017). Conversely, the lack of confidence can be leveraged to efficiently guide exploration in reinforcement learning, via randomizing the approximate value function (Azizzadenesheli et al., 2018; Touati et al., 2018) or maximizing intrinsic rewards (Houthooft et al., 2016).

Furthermore, a considerable proportion of statistical learning theory is devoted to understanding what implies generalization, or what constitutes an appropriate measure of complexity (Bartlett et al., 2017; Arora et al., 2018; Neyshabur et al., 2017). PAC-Bayesian learning theory (McAllester, 1999) specifically explores the generalization property of a randomized prediction rule, and has been recently studied in the context of stochastic neural networks (Dziugaite and Roy, 2017a). In this particular study, the working hypothesis is that good generalization can be guaranteed on the premise that stochastic gradient descent (Robbins and Monro, 1951) finds a solution that obtains certain structural properties, such as flatness.

For computational reasons, considerable effort has been devoted to modelling uncertainty through the injection of independent noise to the network parameters (Graves, 2011; Blundell et al., 2015; Kingma et al., 2015). However, noise independence largely restricts the expressivity of the noise distribution and thus the resulting uncertainty measures are ill-calibrated (Minka et al., 2005; Turner and Sahani, 2011). Attempts have been made to correlate parameters of a neural network,

† Work done while Chin-Wei was an intern at Element AI and Ahmed at Facebook AI Research.

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

including Louizos and Welling (2017); Krueger et al. (2017); Pawlowski et al. (2017), for example, by adapting expressive non-linear invertible transformations developed in the variational inference literature (Rezende and Mohamed, 2015; Kingma et al., 2016; Huang et al., 2018), or via implicit methods (Goodfellow et al., 2014). However, these methods are limited due to their inability to scale well. Louizos and Welling (2017), for instance, resort to a specific multiplicative noise sampled from a lower dimensional space and have to use an auxiliary method to bound the entropy. Krueger et al. (2017), on the other hand, give up on injecting noise on the entire set of parameters and model the distribution of the scale and shift parameter of the pre-activations.

In attempts to address some of the challenges articulated above and efficiently model the joint distribution of a network's parameters, we take inspiration from the Kronecker product, which we notice can be thought of as left-transforming a matrix via a linear map, and then right-transforming it using another linear map, thus providing us an efficient way to correlate the weight parameters. We propose the *Kronecker Flow*, an invertible transformation-based method that generalizes the Kronecker product to its nonlinear counterparts. Our contributions are as follows.

1. We extend the idea of Kronecker product to more general invertible mappings to induce non-linear dependencies, and apply this trick to parameterizing deep stochastic neural networks.

2. We apply our method to predictive tasks and show that our methods work better on larger architectures compared to existing methods.

3. We are the first to apply flow-based methods to tighten the PAC-Bayes bound. We show that the KL divergence in the PAC-Bayes bound can be estimated with high probability, and demonstrate the generalization gap can be further reduced and explained by leveraging the structure in the parameter space.

4. Our methods prove to be competitive over other methods in approximate Thompson sampling in contextual bandit problems.

2 Background

Stochastic neural networks with parameter perturbation normally follow the stochastic process: $q(\ )$, $yjx \quad p(yjx;\ ) = f(x)$, where $$ is the parameters of the neural network f, which outputs the prediction probability vector for classification or the predicted values for regression. We let $D = f(x_i; y_i) : i \ 2 \ [m]g$ be the training set of size m [1], H be the differential entropy $H[q] = \ \mathsf{E}_q[\log q]$, $\ > 0$ be the coefficient controlling the amount of noise injected into the model and the degree of regularization, $l(y; y)$ be the loss function and $\hat{R}_D(\ ) = \frac{1}{m} \sum_{i=1}^{m} l(y_i; f\ (x_i))$ be the empirical risk.

2.1 Variational Bayesian neural networks

Variational Bayesian neural networks are a type of stochastic neural network. Bayesian inference updates our prior belief $p(\ )$ over the model parameters according to the Bayes rule $p(\ jD) \ / \ p(Dj\ )p(\ )$, by incorporating information from the training set through the likelihood function $p(Dj\ )$. Variational inference is a computational realization of Bayesian inference, which casts inference as an optimization problem, where one maximizes the variational lower bound (also known as the evidence lower bound, or the ELBO) on the log marginal likelihood:

$$\log p(D) \quad \mathsf{E}_q\ [\log p(Dj\ ) + \log p(\ )] + H(q\ (\ )); \tag{1}$$

where q is the variational approximate posterior and $p(Dj\ )$ can be decomposed into $\prod_{i=1}^{m} p(y_i jx_i;\ )$ due to conditional independence assumption. The optimal q is the true posterior, i.e. $q^*(\ ) = \frac{p(D|\ )p(\ )}{p(D)}$. In our case, we use $$ to parameterize a neural network. Prediction can be carried out via the (approximate) predictive posterior

$$\begin{aligned} p(yjx; D) &= \mathsf{E}_{\sim p(\ |D)}[p(yjx;\ )] \\ &\approx \mathsf{E}_{\sim q\ (\ )}[p(yjx;\ )] \\ &\approx \frac{1}{K} \sum_{k=1}^{K} p(yjx;\ _k) \end{aligned}$$

for $f\ _k g_{k\in[K]}$ drawn i.i.d. from $q\ (\ )$, where we use the variational distribution q to approximate $p(\ jD)$ and a Monte Carlo estimate to estimate the integral.

The prior distribution can be used to encode some form of inductive bias, such as one that is in favor of parameter values closer to some $_0$ chosen *a priori*. We choose the prior to be an isotropic Gaussian, centered at the random initialization $_0$, i.e., $p(\ ) = \mathcal{N}(\ ;\ _0;\ I)$. The entropy term ensures the variational posterior does not collapse to a point estimate. Both of them can be thought of as some form of regularizer, so we attach a coefficient $$ in front of them as a hyperparameter [2].

[1] We use the notation $[n]$ to compactly describe the set of integers $\{1; 2; : : : ; n\}$.

[2] Like the $$ parameter in Zhang et al. (2017)

2.2 PAC-Bayes generalization bound

Another use case of stochastic neural networks is to understand generalization, via PAC-Bayes bounds. The aim is to bound a divergence between the empirical risk, $\hat{L}[q] = \mathsf{E}_q[\hat{R}_D(\ )]$, and the risk measured on the *true* distribution D, $L[q] = \mathsf{E}_q[\mathsf{E}_{\mathcal{D}}[l(y; f\ (x))]]$. While this quantity is unbounded in the general case, assuming a bounded loss function l (e.g.: zero-one loss), we can obtain a probabilistic bound that holds with probability 1 over the choice of D, for > 0. More specifically, with probability 1 , $(\hat{L}[q]; L[q])$ $(D_{\mathsf{KL}}(q\|p); m;\ )$, where is a measure of complexity that scales proportionally with the Kullback–Leibler (KL) divergence and a measure of divergence (e.g.: square distance or convex functions (Germain et al., 2009)).

For instance, Dziugaite and Roy (2017a) minimize the following bound originally due to McAllester (1999) and then tightened by Langford and Seeger (2001):

Theorem 1. *Let l be the zero-one loss. For any > 0 and data distribution D, and any distribution p on the space of , with probability at least 1 over the choice of a training set D D^m, for all distributions q on the space of ,*

$$D_{\mathsf{KL}}(\hat{L}[q]\|L[q]) \quad \frac{D_{\mathsf{KL}}(q\|p) + \log \frac{m}{}}{m \quad 1}; \qquad (2)$$

where the KL on the LHS is between two Bernoulli distributions, defined by the probability of performing an error.

We refer to the above bound as the *McAllester bound*. The KL divergence on the RHS of the bound, also known as the *information gain*, tells us to what extent the posterior q is dependent on the training data. The sharper and more confident q is, and the farther away it is from the prior p, the larger the KL will be, which in turn is reflected by the larger bound on the generalization gap. This is consistent with traditional notion of bias-variance trade-off.

Alternatively, we consider the following bound due to Catoni (2007):

Theorem 2. *With the l, , D, and p as defined in Theorem 1, and with a fixed $> 1=2$, the following bound holds with probability over 1 :*

$$L[q] \quad \frac{1}{1 \quad \frac{1}{2}} \quad \hat{L}[q] + \frac{}{m} \quad D_{\mathsf{KL}}(q\|p) + \ln \frac{1}{-} \quad : \qquad (3)$$

We refer to this bound as the *Catoni bound*. We notice the linear relationship (which is also noticed by Germain et al. (2016)) between the empirical risk and the KL divergence. This allows us to make use of the linearity of expectation to perform change of variable (see the next section). We also note that the optimal in Equation 3 is always larger than 1, so the PAC-Bayes bound is actually more conservative than Bayesian inference in this sense.

2.3 Normalizing flows

Minimization of Equation 1 and 3 requires (i) computing the gradient with respect to the parameter of the (PAC-)Bayesian posterior , and (ii) computing the entropy of q. One approach to do this is via *change of variable* under an invertible mapping. Let q_0 be a random variable in R^d, and $g : \mathsf{R}^d \to \mathsf{R}^d$ be a bijection parameterized by . Let $= g(\ )$ and q be its density. Then we can rewrite the loss function as [3]

$$\mathsf{E}\left[\hat{R}_D(\ ) + \log q(\ )\right] = \mathsf{E}\left[\hat{R}_D(g(\ )) + \log q_0(\ ) \right.$$
$$\left. \log \det \frac{@g(\ )}{@}\right];$$

where we apply the change of variable (see Appendix A for the detailed derivation). The log-determinant (logdet) term ensures that we obtain a valid probability density function after g is applied, which can be a sequence of invertible mappings itself, hence referred to as the *normalizing flow* (Rezende and Mohamed, 2015). This way, the random variable and the parameters are decoupled, so that we can differentiate the integrand to have an unbiased estimate of the gradient (fixing some q_0). We let q_0 be the standard normal.

3 Kronecker Flows

We consider maximizing the ELBO and minimizing the Catoni bound via normalizing flow-based SNNs. Conventionally, mean-field approximation using factorized distributions (such as multivariate Gaussian with diagonal covariance) has been well explored in the variational inference (VI) literature (Blundell et al., 2015). We are interested in better capturing the structure in the parameter space as restricted VI methods are known to exhibit overconfidence (Minka et al., 2005; Turner and Sahani, 2011). However, the parameters of a neural network are usually very high dimensional (on the order of millions), requiring a novel way to parameterize the joint distribution over the parameters.

In its general form, neural networks can be represented by a collection of tensors i.e. $= \{W_l : l\ 2\ [L]\}$. While our method below can easily be generalized to high-dimensional tensors (such as for convolutional kernels), to simplify notation, we describe the matrix form.

[3] Since the weighting coefficient can be absorbed into the loss function l, we neglect it for simplicity now.

3.1 Linear Kronecker Flow

The matrix-variate normal (MN) distribution generalizes the multivariate normal distribution to matrix-valued random variables, such as weight matrices of a neural network (Louizos and Welling, 2016). Matrix normal is a multivariate normal distribution whose covariance matrix is a Kronecker product ( ), which allows us to model the correlation among the parameters to some degree.

More concretely, assume $E_{ij} \overset{\text{i.i.d.}}{\sim} N(0,1)$ is an $n \times p$ random Gaussian matrix, and $A \in R^{n \times n}$, $B \in R^{p \times p}$ and $M \in R^{n \times p}$ are real-valued matrices. Then $M + AEB$ has a matrix normal distribution, as

$$\text{vec}(M + AEB) \sim N(\text{vec}(M), B^\top B \otimes AA^\top),$$

where vec is the vectorization of a matrix that concatenates all the columns. This allows us to represent the covariance matrix in a more compact manner ($n^2p^2/2$ parameters versus $n^2/2 + p^2/2$ parameters for Kronecker product).

Limitation of the Kronecker product. The Kronecker product covariance matrix is not a strict generalization of diagonal covariance matrix. To observe this, let $U = \text{diag}(u)$, $V = \text{diag}(v)$ (this is the case of Louizos and Welling (2016)), and $S = \text{diag}(s)$, where $u \in R^n_{>0}$, $v \in R^p_{>0}$, and $s \in R^{np}_{>0}$. Then $U \otimes V$ is also a diagonal matrix of size $np \times np$. Equating $U \otimes V = S$ to solve for u and v will result in np nonlinear equations with $n + p$ variables, which can be over-determined for $n, p > 2$. For example, let $n = 2, p = 3$, and $s = [1, \alpha, \alpha, \alpha, 1, 1]$ for some $\alpha > 0$. Then the nonlinear system below does not have a solution:

$$U \otimes V = S \iff \underset{(a)}{u_1 v_1 = 1} \quad \underset{(b)}{u_1 v_2 = \alpha} \quad \underset{(c)}{u_1 v_3 = \alpha} \quad \underset{(d)}{u_2 v_1 = \alpha} \quad \underset{(e)}{u_2 v_2 = \alpha} \quad \underset{(f)}{u_2 v_3 = \alpha} \quad 1$$

To see this, dividing (a) by (b) and dividing (d) by (e) yield $v_1 = v_2/\alpha$ and $v_1 = v_2$, respectively, which doesn't have a solution if $\alpha \neq 1$. This is because the Kronecker product is essentially parameter sharing, which can heavily restrict the matrix it can represent.

To remedy the above limitation, we can further decouple the reparameterization of the parameter matrix into two parts: (1) one that models the marginal variance and (2) one that models correlations. Assume $S \in R^{n \times p}_{>0}$ is a positive-valued matrix, and let $W := M + A(E \odot S)B$. Then $\text{vec}(W)$ is a Gaussian distribution with the following property, which is useful in calculating the KL divergence:

Property 1. Let W be given as above, with $\mu = E[\text{vec}(W)]$ and $\Sigma = \text{Var}(\text{vec}(W))$. Then

(P1) $\mu = \text{vec}(M)$, and
$\Sigma = (B^\top \otimes A)\,\text{diag}(\text{vec}(S^2))(B \otimes A^\top)$

(P2) $\det(\Sigma) = \det(AA^\top)^{2p} \det(B^\top B)^{2n} \prod_{ij} S_{ij}^2$

(P3) $\text{Tr}(\Sigma) = \sum_{ij} A^2 S^2 B^2{}_{ij}$

See Appendix B for the derivation and interpretation of the property. Naive implemetations of this can be inefficient and numerically unstable, as the entropy term involves computing the log-determinant of A and B, requiring the standard automatic differentiation libraries to resort to singular value decomposition when the matrix is near-singular. Thus, we choose to parameterize A and B as lower triangular matrices[4] with ones on the diagonal, leaving the uncertainty to be modeled by S. This means $\det(\Sigma) = \prod_{ij} S_{ij}^2$.

Simulation. To validate the limited expressiveness of kronecker product, we randomly initialize a target density p to be a multivariate Gaussian with mean zero, and covariance being the square of a random standard Gaussian matrix. We choose the dimensionality d of the Gaussian such that it can be decomposed into a product of integers, and parameterizeq using independent Gaussian (dubbed Diag), the Kronecker product with diagonal A and B (K-Diag), and the Kronecker product with elementwise scaling (K-Linear). We minimize $D_{KL}(q||p)$; see Figure 1 for the results. We also conduct the same experiment with 3D tensors (instead of matrices). We see that K-Diag consistently underperforms when compared to Diag, which indicates parameter sharing does restrict the family of distributions it can represent, and K-Linear is consistently better as it captures some correlation.

3.2 Nonlinear Kronecker Flow

In this section, we generalize the Kronecker product to more general non-linear mappings. In Appendix C, we make a connection to non-decreasing triangle maps (Villani, 2008) that are general enough to model any probability distributions.

First, notice that left-multiplying E by A amounts to introducing linear correlation among the n rows of E, applied to each of the p columns. Likewise, right-multiplying E by B amounts to correlating column entries of each row of E. Inspired by this, we consider applying an invertible mapping to each row of the random weight matrix, and another invertible mapping to each column of the matrix. We call this the Kronecker Flow [5].

[4] This is achieved by masking.

[5] To differentiate this from K-Linear from the previous section, we refer to using non-linear g as K-Nonlinear.

(a) Random Gaussian matrix

(b) Random 3D Gaussian tensor

Figure 1: Minimizing KL divergence between q and a randomly initialized distribution p. X-axis indicates the shape of the random matrix/tensor, sorted according to the dimensionality. The shaded area is the error bar with 0.1-standard deviation away from the mean performance, averaged across 25 trials.

Specifically, let $g_A : \mathbb{R}^n \to \mathbb{R}^n$ and $g_B : \mathbb{R}^p \to \mathbb{R}^p$ be invertible mappings. We define the matrix-matrix function $G : \mathbb{R}^{n \times p} \to \mathbb{R}^{n \times p}$ as $G_B(G_A(E^\top)^\top)$, with the following batch-operations (for $i \in [n]$ and $j \in [p]$):

$$G_A(E^\top)_{j:} := g_A(E_{:j}) \qquad G_B(E)_{i:} := g_B(E_{i:})$$

It is easy to verify that G is invertible. Due to the partial dependency of G_A and G_B, the Jacobians of the vectorized forms (after proper permutation) are block-diagonal, so we have

$$\det \frac{\partial \mathrm{vec}(G(E))}{\partial \mathrm{vec}(E)} = \prod_{j \in [p]} \det \frac{\partial g_A(E_{:j})}{\partial E_{:j}} \prod_{i \in [n]} \det \frac{\partial g_B(G_A(E^\top)_{:i})}{\partial G_A(E^\top)_{:i}}.$$

In practice, we use the volume preserving version of RealNVP (Dinh et al., 2016) and inverse autoregressive flow (IAF) (Kingma et al., 2016) to parameterize g_A and g_B for our experiments[6]. The K-Linear from the previous section can be thought of as using a linear map as g_A and g_B.

4 Concentration of empirical KL with normalizing flows

In their study, Dziugaite and Roy (2017a) use independent Gaussian for q to minimize the McAllester bound, so they can compute the KL between Gaussians analytically. This is no longer feasible when we use more flexible families for q, such as normalizing flows. Moreover, a Monte Carlo estimate might result in underestimating the bound after inverting the KL between Bernoullis on the LHS of Equation 2 (which is a concave function; see Appendix A of Reeb et al. (2018) for an illustration). This necessitates a high probability bound on the concentration of the empirical estimate.

In Section 2.3, we have established $D_{KL}(q \| p)$ can be written in the following form

$$\mathbb{E}\left[\log N(\epsilon; 0, I) - \log \det \frac{\partial g(\epsilon)}{\partial \epsilon} - \log N(g(\epsilon); 0, I)\right],$$

where both q_0 and p are standard Gaussian (the mean and variance can be absorbed into the invertible mapping g if this is not the case).

The first term in the KL can be computed analytically. The second term usually can be almost surely bounded (e.g. using Block neural autoregressive flows) so that we can use Hoeffding-type concentration or it can simply be made zero using e.g. volume preserving flows. The challenge now lies in the third term, which has a quadratic form $\frac{1}{2}\|g(\epsilon)\|^2$, neglecting the normalizing constant.

Now assume g is a L_0-Lipschitz [7]. Let $\tilde{g}(\epsilon) = \sqrt{\frac{1}{2}}\|g(\epsilon)\|$. Then $\tilde{g}$ is $L_0\sqrt{2}$-Lipschitz:

[6] We also experimented with Block NAF by De Cao et al. (2019), but did not include it in the manuscript: its performance was similar to IAF, but it is much slower to sample from (we adapted the open implementation from De Cao et al. (2019)).

[7] The following flows are all Lipschitz (with Lipschitz activation functions): volume preserving version of Dinh et al. (2016); Kingma et al. (2016), Berg et al. (2018), Behrmann et al. (2018), De Cao et al. (2019), etc.

$$g(\ _1) \quad g(\ _2) = \frac{1}{\sqrt{2}} jj g(\ _1) jj \ jj \ g(\ _2) jj$$
$$\frac{1}{\sqrt{2}} jj g(\ _1) \quad g(\ _2) jj \quad \frac{L_0}{\sqrt{2}} jj \ _1 \quad _2 jj :$$

This is key in deriving a tail bound on g^2, as Lipschitz functions of canonical Gaussian random variables are sub-Gaussians, meaning they have a tail that decays faster than a Gaussian random variable. The following theorem provides a concentration bound for the empirical average of g^2 similar to that of a Chi-square random variable, as g^2 (square of a sub-Gaussian) is sub-exponential.

Theorem 3. Let g be defined as above with a Lipschitz constant $L = L_0 = \sqrt{2}$. Let $g^2 = \frac{1}{K} \sum_{k=1}^{K} g_k^2$. Then the following concentration bound holds

$$P(g^2 \quad E[g^2] > \ ) \quad \exp \quad \frac{K \ ^2}{2(4C^2 + C \ )} \ ;$$

where $C = \ 6L^2 + \frac{L}{\sqrt{\log 2}} (\sqrt{d} + jj g \ ^1(0) jj) \ ^2$.

Note that in practice the empirical KL that we use is inversely scaled by the size of the training set m (see Equation 2), so the Lipschitz constant can be made small in practice to dominate the dimensionality.

5 Experiments

We evaluate our proposed method in the context of two prediction tasks (Section 5.1), PAC-Bayes bound minimization (Section 5.2) and contextual bandit (Section 5.3). For the two prediction tasks, we use the MNIST handwritten digit dataset (Lecun et al., 1998) and CIFAR-10 (Krizhevsky, 2009). See Appendix E for a detailed description.

5.1 Classification

One benefit of Bayesian neural networks compared to the regular ones is that the trade-off between the prior and the likelihood is a form of regularization. In this section, we evaluate the generalization performance of our method applied to Bayesian neural networks. We consider two architectures: LeNet-5 (Lecun et al., 1998) and a modified version VGG-16 (Simonyan and Zisserman, 2014) proposed by Zhang et al. (2017).

We first compare to the multiplicative normalizing flow (MNFG) proposed by Louizos and Welling (2017), applying our method to LeNet-5 (see Table 1). Our Diag matches the performance of their FFG (fully factorized Gaussian). K-Diag outperforms Diag in this case, perhaps due to the smaller number of parameters which makes it easier to optimize. K-Nonlinear yields the best generalization error in this case. On the CIFAR-5 experiment (we take the first 5 classes of CIFAR-10), our methods are on par with MNFG.

Second, we compare with the noisy K-FAC proposed by Zhang et al. (2017), applying our methods to the larger architecture VGG-16 (see Table 2). Noisy K-FAC applies an approximate natural gradient method. Despite this advantage, our methods (K-Linear and K-Nonlinear) have simiar prediction accuracy in the regular setup. We also include the results of data augmentation with horizontal flip and random crop where K-Nonlinear outperforms all the other methods.

5.2 PAC Bayes bound minimization

For the PAC-Bayes bound estimation, we minimize Equation 3. We follow the recipe of Dziugaite and Roy (2017a). We upper bound the zero-one loss by cross-entropy divided by $\log jYj$ (where jYj is the number of classes) to make the upper bound tight. We set the prior to be $N(\ _0; \ I)$, where $\ _0$ is the initial value of the parameters, and apply a union bound to tune the prior variance $\ $. We also tune the $\ $ coefficient as a parameter during training [8], and report the McAllester bound for comparison (since it is the tightest). For more details, see Dziugaite and Roy (2017a) for reference.

We test with a multi-layer perceptron with 1 or 2 hidden layers with 600 neurons and LeNet-5, evaluated on the MNIST dataset (see Table 3). For further clarification, we follow the steps of Dziugaite and Roy (2017a) by minimizing the McAllester bound, using Pinsker's inequality to bound the inverse of the Bernoulli KL (which we call the Pinsker bound). Since this bound has a square root in the complexity term, we can only use the Gaussian family with an analytic form of the KL. The result we have is slightly looser than Dziugaite and Roy (2017a) since we have a 10-class problem and they deal with a binary version of MNIST. We see that the bound can indeed be improved by capturing the correlation among the parameters. We then compare to minimizing the Catoni bound, which is slightly looser since the linear relationship between the empirical risk and the KL term penalizes the latter more when the KL is larger. However, by modelling the non-linear dependencies, K-Nonlinear clearly outperforms the other methods (even compared to the ones minimizing the Pinsker bound). This indicates there exists a considerable amount of structure in the parameter space that may explain the gap between the test error and the generalization bound.

[8] We are allowed to do so since we treat Equation 3 as an optimization objective, rather than report it as a bound. We report the McAllester bound, which holds for any q, even if it depends on $\ $.

Table 1: Test error with LeNet (%) on MNIST and the first 5 classes of CIFAR-10. First 3 columns are from Louizos and Welling (2017). K-Diag on CIFAR-5 diverged, so we did not include the result.

Dataset	L2	FFG	MNFG	Diag	K-Diag	K-Linear	K-Nonlinear
MNIST	0.6	0.9	0.7	0.92	0.67	0.70	0.60
CIFAR-5	24	22	16	19.0	-	16.8	17.4

Table 2: Test error with modified version of VGG16 (%) on CIFAR10. First 4 columns are from Zhang et al. (2017). R means regular training and D means training with data augmentation.

Setup	SGD	KFAC	BBB	Noisy-KFAC	Diag	K-Diag	K-Linear	K-Nonlinear
R	18.21	17.61	17.18	14.48	17.71	16.71	14.65	14.74
D	11.65	11.11	11.69	10.65	10.69	13.65	11.35	9.88

Table 3: PAC-Bayes bound estimation: We minimize the Pinsker bound (an upper bound on the McAllester bound) and the Catani bound using different flows, and estimate the McAllester bound at inference time using Newton's method.

Bound	Pinsker Bound				Catoni Bound						Catoni Bound			
Flow	Diag		K-Linear		Diag		K-Linear		K-Nonlinear		D	K-D	K-L	K-N
L 1	1	2	1	2	1	2	1	2	1	2	LeNet-5			
$\hat{L}[q]$	6.62	6.00	6.09	5.90	8.04	7.66	8.10	8.33	5.96	5.90	2.12	2.95	2.00	2.01
L[q]	6.66	6.12	5.98	5.96	7.78	7.70	7.98	8.26	5.83	5.76	2.31	2.87	1.91	2.14
bound	23.77	25.94	21.69	25.33	24.11	26.41	22.88	26.43	20.41	22.53	10.83	12.96	10.09	10.03
KL	5968	7829	5292	7554	5001	6555	4334	5996	4725	5921	3177	3477	2913	2873

Table 4: Cumulative regret incurred by different algorithms on the bandit benchmarks described in Riquelme et al. (2018). Values reported are the mean over 3 independent trials with standard error of the mean, normalized with respect to the performance of the uniform policy.

Bandit	SGD		fBNN		Diag		K-Diag		K-Linear		K-Nonlinear	
Mushroom	4.06	0.71	3.91	0.89	2.16	0.29	2.41	0.73	1.85	0.15	3.47	0.47
Statlog	1.29	0.20	0.73	0.01	1.01	0.01	0.84	0.06	0.81	0.01	0.79	0.04
Covertype	30.01	0.21	32.03	0.40	28.42	0.30	29.19	0.16	28.13	0.12	28.06	0.15
Financial	6.08	0.47	7.27	1.09	7.43	0.57	5.88	0.25	5.88	0.35	5.78	0.28
Jester	56.24	1.93	59.70	2.48	59.34	2.26	57.17	1.81	57.66	2.11	57.96	2.58
Adult	79.31	0.47	84.45	0.82	76.32	0.09	77.28	0.01	75.94	0.12	77.30	0.26

We also notice that, despite the linear relationship, the Catoni bound focuses more on the complexity term than the ELBO. For example, the empirical risks of LeNet-5 in Table 3 are much higher compared to the test loss of Table 1. The reasons are: (1) the optimal in Equation 3 is larger than 1 (depending on the relative value of the KL), and (2) to properly upper bound the zero-one loss, we scale down the cross-entropy loss by $\log|\mathcal{Y}|$ during optimization. This means a learning algorithm based on a tight PAC-Bayes risk bound cannot over t by much; see Dziugaite and Roy (2017b) for a recent demonstration. A smaller value of could bring down the risk and empirical risk, resulting in a looser bound but usually better test performance. This trade-o between test set performance and the tightness of the bound is a general issue for generalization bounds.

One reason for the interest in PAC-Bayes bounds is that their optimization leads to training algorithms with generalization guarantees. The bounds, however, are considerably looser than held-out estimates.[9] Our work produces much tighter bounds by building exible families of distributions on neural network weight matrices.

5.3 Contextual bandit

Uncertainty modeling lies at the heart of the exploration-exploitation dilemma in sequential decision-making. In order to maximize its collected cumulative rewards, an agent should trade o exploring di erent actions and gaining more knowledge about the reward estimate vs. exploiting the current estimate and allocating resources to the actions that are likely rewarding. Thompson sampling (TS) (Thompson, 1933) is one the popular approaches that deals with the latter trade-o by maintaining posterior distribution over reward models and randomizing actions on the basis of their probability of being optimal.

In this section, we investigate the e ectiveness of our proposed method for performing an approximate Thompson sampling in the particular setting of contextual bandits. In the latter setting, at each time $t = 1 \dots T$, the agent sees a d-dimensional context X_t, selects one of the k available actions, a_t, and earns a reward r_t generated by the environment. The agent aims to minimize its cumulative regret de ned as $R = E[\sum_{t=1}^{T} r_t^* - r_t]$ where r_t^* is the highest expected reward given the context X_t and the expectation is over the randomness of both environment and the agent's choice of actions.

[9] A recent work by Rivasplata et al. (2019) shows PAC-Bayes bound can potentially be made tight; however, we could not reproduce the results and the best bound using Gaussian prior was around 40% in their setting.

We compare di erent methods on a range of real-world bandit problems introduced by Riquelme et al. (2018), including Mushroom (Linco , 1989), Statlog (Asuncion and Newman, 2007), Covertype (Blackard and Dean, 1999), Financial (Riquelme et al., 2018), Jester (Goldberg et al., 2001), and Adult (Kohavi, 1996). We train the models every 50 time steps for 200 iterations using a batch-size of 512. We ran each experiment with 3 di erent random seeds and we report the means and standard errors of cumulative regret normalized with respect to the uniform baseline in the table 4. We include the functional variational Bayesian neural networks (fBNN), recently introduced by Sun et al. (2019) as a baseline, and we use their open sourced implementation of fBNN in the bandit setting. From table 4, we see that across the 6 bandit problems, our proposed method (K-Linear and K-Nonlinear) provides competitive and consistent results. They outperform other baselines in 4 problems out of 6.

6 Conclusion

In this work, we present the Kronecker Flow, a ow-based method to induce complex distribution inspired by the Kronecker product. Our methods scale to larger architectures such as VGG-16 since it takes advantage of the shape of the parameters. We demonstrate our methods work better than vanilla Kronecker product with diagonal matrices on multiple setups, including classi cation and approximate Thompson sampling in contexual bandit, and prove to be competitive with existing methods in the Bayesian neural network literature. We are also the rst to apply ow-based methods to obtain a tighter numerical generalization bound. Our work shows that the dependencies among network parameters constitute a non-negligible portion of the gap between risk and PAC-Bayes generalization bound.

Acknowledgement

CWH would like to thank Kris Sankaran for pointing to the TIS inequality for Gaussian concentration, which is a key component in deriving the tail bound on Lipschitz ows. Special thanks to Salem Lahlou for discussion on deriving the concentration inequality, and to Guillaume Rabusseau for discussion on Kronecker factorization.

References

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. (2018). Stronger generalization bounds for deep nets via a compression approach. arXiv preprint arXiv:1802.05296.

Asuncion, A. and Newman, D. (2007). Uci machine learning repository.

Azizzadenesheli, K., Brunskill, E., and Anandkumar, A. (2018). E cient exploration through bayesian deep q-networks. CoRR, abs/1802.04412.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. In Advances in Neural Information Processing Systems, pages 6240 6249.

Behrmann, J., Duvenaud, D., and Jacobsen, J.-H. (2018). Invertible residual networks. arXiv preprint arXiv:1811.00995.

Berg, R. v. d., Hasenclever, L., Tomczak, J. M., and Welling, M. (2018). Sylvester normalizing ows for variational inference. arXiv preprint arXiv:1803.05649.

Blackard, J. A. and Dean, D. J. (1999). Comparative accuracies of arti cial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. Computers and electronics in agriculture, 24(3):131 151.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In Proceedings of The 32nd International Conference on Machine Learning, pages 1613 1622.

Bogachev, V. I., Kolesnikov, A. V., and Medvedev, K. V. (2005). Triangular transformations of measures. Sbornik: Mathematics, 196(3):309 335.

Boucheron, S., Lugosi, G., and Massart, P. (2013). Concentration inequalities: A nonasymptotic theory of independence. Oxford university press.

Catoni, O. (2007). Pac-bayesian supervised classi cation: The thermodynamics of statistical learning. Lecture Notes-Monograph Series, 56:i 163.

De Cao, N., Titov, I., and Aziz, W. (2019). Block neural autoregressive ow. arXiv preprint arXiv:1904.04676.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. arXiv preprint arXiv:1605.08803.

Dziugaite, G. K. and Roy, D. M. (2017a). Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In Proceedings of the 33rd Conference on Uncertainty in Arti cial Intelligence .

Dziugaite, G. K. and Roy, D. M. (2017b). Entropy-sgd optimizes the prior of a pac-bayes bound: Generalization properties of entropy-sgd and data-dependent priors. arXiv preprint arXiv:1712.09376 .

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning, pages 1050 1059.

Germain, P., Bach, F., Lacoste, A., and Lacoste-Julien, S. (2016). Pac-bayesian theory meets bayesian inference. In Advances in Neural Information Processing Systems, pages 1884 1892.

Germain, P., Lacasse, A., Laviolette, F., and Marchand, M. (2009). Pac-bayesian learning of linear classi ers. In Proceedings of the 26th Annual International Conference on Machine Learning, pages 353 360. ACM.

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative ltering algorithm. information retrieval , 4(2):133 151.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems.

Graves, A. (2011). Practical variational inference for neural networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, Advances in Neural Information Processing Systems 24, pages 2348 2356. Curran Associates, Inc.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In International Conference on Machine Learning, pages 1861 1869.

Houthooft, R., Chen, X., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). Vime: Variational information maximizing exploration. In Advances in Neural Information Processing Systems 29.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. (2018). Neural autoregressive ows. In International Conference on Machine Learning.

Hyvärinen, A. and Pajunen, P. (1999). Nonlinear independent component analysis: Existence and uniqueness results. Neural Networks, 12(3).

Ian, O., Benjamin, V. R., and Daniel, R. (2013). (more) e cient reinforcement learning via posterior sampling. In Proceedings of the 26th International Conference on Neural Information Processing Systems, USA. Curran Associates Inc.

Jaini, P., Selby, K. A., and Yu, Y. (2019). Sum-of-squares polynomial ow. In International Conference on Machine Learning.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive ow. In Advances in Neural Information Processing Systems.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization

trick. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc.

Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.

Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., and Courville, A. (2017). Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*.

Langford, J. and Seeger, M. (2001). Bounds for averaging classifiers.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11).

Lincoff, G. H. (1989). The audubon society field guide to north american mushrooms. Technical report.

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

Louizos, C. and Welling, M. (2017). Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*.

MacKay, D. J. (1992). A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.

McAllester, D. A. (1999). Pac-bayesian model averaging. In *COLT*, volume 99, pages 164–170. Citeseer.

McAllister, R., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., and Weller, A. (2017). Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 4745–4753. AAAI Press.

Minka, T. et al. (2005). Divergence measures and message passing. Technical report, Technical report, Microsoft Research.

Müller, T., McWilliams, B., Rousselle, F., Gross, M., and Novák, J. (2018). Neural importance sampling. *arXiv preprint arXiv:1808.03856*.

Neyshabur, B., Bhojanapalli, S., Mcallester, D., and Srebro, N. (2017). Exploring generalization in deep learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5947–5956. Curran Associates, Inc.

Osband, I., Van Roy, B., and Wen, Z. (2016). Generalization and exploration via randomized value functions. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2377–2386. JMLR.org.

Pawlowski, N., Brock, A., Lee, M. C. H., Rajchl, M., and Glocker, B. (2017). Implicit Weight Uncertainty in Neural Networks. *arXiv e-prints*.

Reeb, D., Doerr, A., Gerwinn, S., and Rakitsch, B. (2018). Learning gaussian processes by minimizing pac-bayesian generalization bounds. In *Advances in Neural Information Processing Systems*, pages 3337–3347.

Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*.

Riquelme, C., Tucker, G., and Snoek, J. R. (2018). Deep bayesian bandits showdown.

Rivasplata, O., Tankasali, V. M., and Szepesvari, C. (2019). Pac-bayes with backprop. *arXiv preprint arXiv:1908.07380*.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sun, S., Zhang, G., Shi, J., and Grosse, R. (2019). Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*.

Touati, A., Satija, H., Romoff, J., Pineau, J., and Vincent, P. (2018). Randomized value functions via multiplicative normalizing flows. *arXiv preprint arXiv:1806.02315*.

Turner, R. E. and Sahani, M. (2011). Two problems with variational expectation maximisation for time-series models. *Bayesian Time series models*, 1(3.1):3–1.

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

Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. (2017). Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390*.

A Law of the unconscious statistician

Let $(\ ; F; \mathrm{P})$ be our probability space. Let $\ 2 \ \mathrm{R}^d$ be a random variable following the (Lebesgue) density $q_0(\ ) = \frac{\mathrm{d} \ \mathbb{P}}{\mathrm{d}}$ and $\ _*\mathrm{P}$ being its pushforward measure, and write $\ = g \ (\ ) \ 2 \ \mathrm{R}^d$ with $q \ = \frac{\mathrm{d}(g \ \circ \ ) \ \mathbb{P}}{\mathrm{d}}$ being its density and $(g \ \ )_*\mathrm{P}$ being its pushforward measure, and $A = \hat{R}_D(\ ) + \log q \ (\ ) \ 2 \ \mathrm{R}$. Then

$$
\begin{aligned}
\mathrm{E}[A] = \int_{\mathbb{R}^d} A \ \mathrm{d}(g \ \ )_*\mathrm{P} &= \int_{\mathbb{R}^d} \hat{R}_D(\ ) + \log q \ (\ ) \ q \ (\ ) \mathrm{d} \\
= \int_{\mathbb{R}^d} A \ \ \mathrm{d} \ _*\mathrm{P} &= \int_{\mathbb{R}^d} \hat{R}_D(g \ (\ )) + \log q \ (g \ (\ )) \ q_0(\ ) \mathrm{d} \\
&= \int_{\mathbb{R}^d} \hat{R}_D(g \ (\ )) + \log q_0(\ ) \ \ \log \ \det \frac{@g \ (\ )}{@} \ \ q_0(\ ) \mathrm{d}
\end{aligned}
$$

B Derivation and interpretation of Property 1

We first derive Property 1 algebraically, and give an interpretation that can be genralized to higher dimensional tensor operation. Recall that we have the following givens:

Assume $E_{ij} \overset{\text{i.i.d.}}{} N(0; 1)$ is a $n \ \ p$ random Gaussian matrix.

Assume $A \ 2 \ \mathrm{R}^{n \times n}$ and $B \ 2 \ \mathrm{R}^{p \times p}$.

$S \ 2 \ \mathrm{R}^{n \times p}_{>0}$.

$M \ 2 \ \mathrm{R}^{n \times p}$.

If we rescale E elementwise by S before inducing the column-wise and row-wise correlation, we have: (superscript is Hadamard power)

$$
\begin{aligned}
:= \mathrm{Var}(\mathrm{vec}(M \ + A(E \ \ S)B)) &= \mathrm{Var}((B^\top \ \ A)\mathrm{vec}(E \ \ S)) \\
&= (B^\top \ \ A) \, \mathrm{diag}(\mathrm{vec}(S^2))(B^\top \ \ A)^\top \\
&= (B^\top \ \ A) \, \mathrm{diag}(\mathrm{vec}(S^2))(B \ \ A^\top)
\end{aligned}
$$

If S is a matrix of ones, the RHS equals $(B^\top \ \ A)(B \ \ A^\top) = (B^\top B) \ \ (A A^\top)$, which is the covariance of the matrix normal.

Generally, S might not be a matrix of ones. But we can still compute the determinant and trace of the covariance matrix (useful in computing KL):

$$
\det(\ ) = \det(A)^{2p} \det(B)^{2n} \prod_{ij} S_{ij}^2
$$

$$
\begin{aligned}
\mathrm{Tr}(\ ) &= \sum_{ii} \\
&= \sum_i \sum_j (B^\top \ \ A)_{ij} \mathrm{vec}(S^2)_j (B \ \ A^\top)_{ji} \\
&= \sum_i \sum_j (B^{2\top} \ \ A^2)_{ij} \mathrm{vec}(S^2)_j \\
&= \sum_i (B^{2\top} \ \ A^2) \mathrm{vec}(S^2)_{\ i} \\
&= \sum_{ij} A^2 S^2 B^2 \ _{ij}
\end{aligned}
$$

Interpretation of the determinant and trace. The determinant measures the change in volume due to the linear map. Since each operation (elementwise multiplication with S, left-multiplication with A, and right-multiplication with B) is an invertible map, the determinant of the composition is a product of determinants. After elementwise multiplication with S (hence $\prod_{ij} S_{ij}$)[10], we apply the same linear map A to the columns of $(E \quad S)$; in vector form, this corresponds to left-multiplication with a block diagonal of p A's, hence $\det(A)^p$. The same reasoning explains $\det(B)^n$.

The trace of the covariance can be written as $\mathrm{Tr}(\quad) = \sum_{ij} \mathrm{Var}(W_{ij})$, i.e. the sum of marginal variances. Each of the W_{ij} is a linear combination of the entries of E, which have unit variance and are uncorrelated, so by the additive property of variance of sum of uncorrelated random variables and the quadratic scaling property of variance, $\mathrm{Var}(W_{ij}) = A_{i:}^2 S^2 B_{:j}^2$.

C Connection to triangular maps.

Much of the recent work on normalizing flows has been dedicated to inverse autoregressive transformations (Kingma et al., 2016; Huang et al., 2018; Müller et al., 2018; De Cao et al., 2019; Jaini et al., 2019), as they are general enough to induce any density function (Hyvärinen and Pajunen, 1999; Bogachev et al., 2005; Villani, 2008). When such transformations are used for g_A and g_B, the overall transformation G is also a triangle map, since $G(E)_{ij}$ depends on $E_{i'j'}$ for $i' \quad i$ and $j' \quad j$. Such a function has some "blind spots" similar to the ones discovered by Van den Oord et al. (2016). One avenue for improvement is to design a transformation that increase the connectivity. Another avenue for improvement is to condition each (row-wise or column-wise) transformation on a learnable embedding of the row/column, such that each row/column is transformed by a slightly different function than another [11].

D Tail bound of empirical KL

We begin with some preliminaries and lemmas in Section D.1, and prove the main result in Section D.2.

D.1 Basic tail bounds and Bernstein inequality

The tools developed in this section is to translate the coefficients (such as variance) of sub-Gaussian random variables and sub-exponential random variables. We start with the definition of sub-Gaussians:

Definition 1. *We write* $X \quad \mathrm{subN}(L^2)$ *if* X *is a random variable satisfying*

$$\mathrm{P}(|X| > t) \quad 2\exp\left(\frac{t^2}{2L^2}\right)$$

We write $(\,)$ as the Gamma function: $(z) = \int_0^\infty e^{-u} u^{z-1} \mathrm{d}u$. Note that for positive integers z, $(z) = (z \quad 1)!$. The following lemma gives an upper-bound on the moments of a sub-Gaussian.

Lemma 4. *For* $X \quad \mathrm{subN}(L^2)$*, for any integer* $p \quad 1$, $\mathrm{E}[|X|^p] \quad (2L^2)^{p/2} p \ (p/2)$.

Proof. Since $|X|^p$ is non-negative, similar to Lemma 6, we have

$$\mathrm{E}[|X|^p] = \int_0^\infty \mathrm{P}(|X|^p \quad s)\mathrm{d}s = \int_0^\infty \mathrm{P}(|X| \quad t) p t^{p-1} \mathrm{d}t$$
$$2p \int_0^\infty e^{-t^2/2L^2} t^{p-1} \mathrm{d}t = \quad p(2L^2)^{p/2} \int_0^\infty e^{-u} u^{p/2-1} \mathrm{d}u = p(2L^2)^{p/2} \ (p/2)$$

where we let $s = t^p$ and $u = t^2/2L^2$. □

The following definition is the main tool for translating the coefficients.

[10] The power 2 comes from the fact that we are looking at the determinant of the covariance. Direct computation of the likelihood involves $\frac{1}{2}\log\det(\quad)$, which is equivalent to the log-determinant of the invertible map.

[11] We try this idea in the preliminary stage of the project, but find it harder to optimize. This is potentially due to the extra parameters that have to be learned.

Definition 2. *Let X be a random variable. For integer $k \geq 1$, define the $\|\cdot\|_k$-Orlicz norm as*

$$\|X\|_k := \inf\{t > 0 : \mathrm{E}[\exp(|X|^k/t^k)] \leq 2\}$$

i.e, the smallest constant $t > 0$ for which the super-exponential moment of X^k/t^k is bounded by 2. The Orlicz norm is infinity if there's no finite t for which $\mathrm{E}[\exp(|X|^k/t^k)]$ exists.

It is easy to verify that the Orlicz norm is indeed a norm. We call $\|\cdot\|_2$ the sub-Gaussian norm, and $\|\cdot\|_1$ the sub-exponential norm. Note that $\|X^2\|_1 = \|X\|_2^2$.

The following lemma upper bounds the sub-Gaussian norm by its variance.

Lemma 5. *If $X \in \mathrm{subN}(L^2)$, $\|X\|_2 \leq 6L^2$.*

Proof. By power series expansion of the exponential function,

$$\mathrm{E}[e^{cX^2}] = 1 + \sum_{p=1}^{\infty} \frac{c^p \mathrm{E}[X^{2p}]}{p!} \leq 1 + \sum_{p=1}^{\infty} \frac{c^p}{p!} 2(2L^2)^p p! = 1 + 2\sum_{p=1}^{\infty} (2cL^2)^p$$

where we used Lemma 4 for the inequality. The RHS converges and is equal to 2 if $c = 1/6L^2$. Thus, $\|X\|_2 \leq 6L^2$. □

The following lemma gives an upper bound on the moments of sub-exponential random variables.

Lemma 6. *If for some $C > 0$, $\mathrm{E}[\exp(|X|/C)] \leq 2$, then $\mathrm{E}[|X|^p] \leq 2C^p p!$.*

Proof. By Markov's inequality,

$$\mathrm{P}(|X| > t) \leq \frac{\mathrm{E}[\exp(|X|/C)]}{\exp(t/C)} \leq 2e^{-t/C}$$

For $p \in \mathbb{Z}^+$, since $|X|^p$ is non-negative,

$$\mathrm{E}[|X|^p] = \int_0^{\infty} \mathrm{P}(|X|^p \geq s)ds = \int_0^{\infty} \mathrm{P}(|X| \geq t)pt^{p-1}dt$$
$$\leq 2p\int_0^{\infty} e^{-t/C} t^{p-1}dt = 2pC^p \int_0^{\infty} e^{-u} u^{p-1}du = 2pC^p\,\Gamma(p) = 2C^p p!$$

where we let $s = t^p$ and $u = t/C$. □

Finally, we derive a concentration bound for sub-exponential random variables.

Theorem 7. *(Bernstein's inequality for sub-exponential random variables) Let $(X_i)_{i\in[n]}$ be independent real-valued random variables satisfying $\mathrm{E}[\exp(|X|/C)] \leq 2$ for some $C > 0$, with mean $\mu_X = \mathrm{E}[X]$, and let $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$. Then, for any $\epsilon > 0$, the following concentration bound holds:*

$$\mathrm{P}(\bar{X} - \mu_X > \epsilon) \leq \exp\left(-\frac{n\epsilon^2}{2(4C^2 + C\epsilon)}\right)$$

Proof. Let $\alpha = 4nC^2$ and $c = C$. Then by Lemma 6, $\sum_{i=1}^n \mathrm{E}[X_i^2] \leq n \cdot 4C^2 = \alpha$ and for integers $p > 2$: $\sum_{i=1}^n \mathrm{E}[|X_i|^p] \leq 2nC^p p! = \alpha C^{p-2} p!/2 = \alpha c^{p-2} p!/2$. Then by Corollary 2.11 of Boucheron et al. (2013), we have

$$\mathrm{P}(\bar{X} - \mu_X > \epsilon) = \mathrm{P}\left(\sum_{i=1}^n (X_i - \mu_X) > n\epsilon\right) \leq \exp\left(-\frac{n\epsilon^2}{2(4C^2 + C\epsilon)}\right)$$

□

D.2 Proof of Theorem 3

Since $g(\ ) := g(\ )\quad \mathsf{E}[g(\ )]$ is L-Lipschitz, according to Theorem 5.5 and 5.6 of Boucheron et al. (2013), $g\quad \mathrm{sub}N(L^2)$. And we have that

$$jjg^2jj_{\ 1} = jjgjj^2_{\ 2} = jjg + \mathsf{E}[g]jj^2_{\ 2} \qquad jjgjj_{\ 2} + \not{p}\frac{\mathsf{E}[g]}{\log 2}^{\ 2}$$

due to triangle inequality of the norm. Now since g is L-Lipschitz, its expectation can be bounded by

$$\mathsf{E}[g] = \mathsf{E}\ \not{p}\frac{1}{2}jjg(\ )\quad 0jj \qquad L\mathsf{E}\ jj \qquad g^{-1}(0)jj \qquad L(\mathsf{E}[jj\ jj] + jjg^{-1}(0)jj)$$

Since $\quad$ is standard-normally distributed, $jj\ jj$ follows the chi distribution with d degrees of freedom, which has an expectation that can be upper-bounded using Gautschi's inequality (using Wendel's version of the upper bound):

$$\mathsf{E}[jj\ jj] = \not{p}_{\overline{2}}\frac{((d+1)=2)}{(d=2)} \qquad \not{p}_{\overline{2}}\ \frac{d}{2}^{\ 1=2} = \not{p}_{\overline{d}}$$

Combining the above and using Lemma 5, we have

$$jjg^2jj_{\ 1} \qquad 6L^2 + \not{p}\frac{L}{\log 2}(\not{p}_{\overline{d}} + jjg^{-1}(0)jj)^{\ 2}$$

Setting C to be the RHS and applying Theorem 7 yield the desired result.

E Experimental Details

For the predictive tasks (Section 5.1), we use a cosine annealing schedule for the learning rate, scaling down to 0.01 of the initial learning rate, and pretrain a deterministic network for 10 epochs using the Adam optimizer with a learning rate of 0.001, to initialize the mean of the Gaussian q_0, and train q for 200 epochs.

LeNet-5 MNIST. We use a linear annealing schedule of the coefficient (from 0 back to 1) for 50,000 iterations. We use the Adam optimizer with a learning rate of 0.0005. The result we get for K-Linear uses polyak averaging with exponential decay coefficient 0.995. We use the volume preserving version of RealNVP for the K-Nonlinear. We use the standard Gaussian prior for p.

LeNet-5 CIFAR-5 We use the same architecture as Louizos and Welling (2017) (192 convolutional kernels and 1,000 hidden units for the fully connected layers). We use a linear annealing schedule of the coefficient (from 0 back to 1) for 20,000 iterations for Diag, and no annealing for K-Linear and K-Nonlinear. We use the Adam optimizer with a learning rate of 0.0003, 0.0003, 0.0005 for Diag, K-Linear and K-Nonlinear, respectively. We use the volume preserving version of RealNVP for K-Nonlinear. We use the standard Gaussian prior for p.

VGG-16 CIFAR-10 We use the modified version of VGG-16 proposed by Zhang et al. (2017). We use a learning rate of 0.0005 for all experiments but K-Nonlinear in the regular setup (where we use 0.001). We use the isotropic Gaussian prior with variance being 0.1, and set to be [0.5, 0.1, 0.1, 0.5] in the regular setup and [0.5, 0.1, 0.1, 0.1] in the data augmented setup for Diag, K-Diag, K-Linear, and K-Nonlinear, respectively. We use the volume preserving version of RealNVP for the K-Nonlinear.

PAC-Bayes MLP We follow the same steps as Dziugaite and Roy (2017a), except we did not discretize the prior variance after tuning. In practice this does not affect the bound much. We also did not initialize the mean of q_0 in our setup using SGD for our experiments. We train the stochastic network for 300 epochs, with a learning rate of 0.002. The bound holds with probability at least 0.965 over the choice of prior and the training set. The b and c coefficients in Dziugaite and Roy (2017a) are set as 100 and 0.1. We use the volume preserving version of IAF for the K-Nonlinear.

PAC-Bayes LeNet-5 The same setup as PAC-Bayes MLP, except with polyak averaging with coefficient 0.995. We use the volume preserving version of IAF for the K-Nonlinear.

Bandit Benchmark All the models share the same architechture: one hidden layer with 50 units. We use the volume preserving version of RealNVP for K-NonLinear. We train models every 50 time steps for 200 training iterations using a batch-size of 512.

Table 5: Description of bandit problem: number of actions and number of contexts used for experiments. Comparing to Riquelme et al. (2018) benchmark, we restrict ourself to 50000 contexts for Covertype instead of 150000 contexts.

Bandit problem	number of actions	number of contexts
Mushroom	2	50000
Statlog	7	43500
Covertype	7	50000
Financial	8	3713
Jester	8	19181
Adult	14	45222

Table 6: Additional results: Cumulative regret incurred by different algorithms on the bandit benchmarks described in Riquelme et al. (2018). Values reported are the mean over 5 independent trials with standard error of the mean, normalized with respect to the performance of the uniform policy. We use the same hyperparameters for different algorithms without any finetuning: learning rate = 0.0001 and 100 training epochs.

Bandit	SGD	Diag	K-Diag	K-Linear	K-Nonlinear
Mushroom	**1.82 ± 0.53**	2.12 ± 0.13	2.12 ± 0.47	2.20 ± 0.18	3.55 ± 0.74
Statlog	3.31 ± 1.27	4.12 ± 0.16	**1.23 ± 0.06**	3.49 ± 0.16	1.33 ± 0.02
Covertype	31.70 ± 0.17	34.64 ± 0.16	30.82 ± 0.21	32.84 ± 0.16	**29.24 ± 0.10**
Financial	20.31 ± 2.24	27.60 ± 1.37	**11.83 ± 0.67**	25.10 ± 1.10	13.10 ± 0.36
Jester	56.90 ± 1.20	59.26 ± 1.38	57.22 ± 1.35	58.16 ± 1.20	**56.88 ± 1.91**
Adult	78.70 ± 0.46	79.16 ± 0.19	77.30 ± 0.18	**76.98 ± 0.04**	77.83 ± 0.19