Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors

Christos Louizos   C.LOUIZOS@UVA.NL

AMLAB, Informatics Institute, University of Amsterdam

Max Welling   M.WELLING@UVA.NL

AMLAB, Informatics Institute, University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)

Abstract

We introduce a variational Bayesian neural network where the parameters are governed via a probability distribution on random matrices. Specifically, we employ a matrix variate Gaussian (Gupta & Nagar, 1999) parameter posterior distribution where we explicitly model the covariance among the input and output dimensions of each layer. Furthermore, with approximate covariance matrices we can achieve a more efficient way to represent those correlations that is also cheaper than fully factorized parameter posteriors. We further show that with the "local reprarametrization trick" (Kingma et al., 2015) on this posterior distribution we arrive at a Gaussian Process (Rasmussen, 2006) interpretation of the hidden units in each layer and we, similarly with (Gal & Ghahramani, 2015), provide connections with deep Gaussian processes. We continue in taking advantage of this duality and incorporate "pseudo-data" (Snelson & Ghahramani, 2005) in our model, which in turn allows for more efficient posterior sampling while maintaining the properties of the original model. The validity of the proposed approach is verified through extensive experiments.

1. Introduction

While deep learning methods are beating every record in terms of predictive accuracy, they do not yet provide the user with reliable confidence intervals. Yet, for most applications where *decisions* are made based on these predictions, confidence intervals are key. Take the example of an autonomous driving vehicle that enters a new unknown traffic situation: recognizing that predictions become unreliable and handing the steering wheel back to the driver is essential. Similarly, when a physician diagnoses a patient with some ailment and prescribes a drug with potentially severe side effects, it is essential that she/he knows when predictions are unreliable and additional investigation is necessary. These considerations have motivated us to develop a fully Bayesian deep learning framework that is accurate, efficient and delivers reliable confidence intervals.

Furthermore, by being Bayesian we can also harvest another property as a byproduct; natural protection against *overfitting*. Instead of making point estimates for the parameters of the network, which can overfit and provide erroneously certain predictions, we estimate a full posterior distribution over these parameters. Armed with these posterior distributions we can now perform predictions using the posterior predictive distribution, i.e. we can now marginalize over the network parameters and make predictions on the basis of the datapoints alone. As a result we can both obtain the aforementioned confidence intervals and better regularize our networks, which is very important in problems where we do not have enough data relative to the amount of features.

Obtaining the parameter posterior distributions for large neural networks is however intractable. To this end, many methods for approximate posterior inference have been devised. Markov Chain Monte Carlo (MCMC) methods are one class of methods that have been explored in this context via Hamiltonian Monte Carlo (Neal, 2012) and stochastic gradient methods (Welling & Teh, 2011; Ahn et al., 2012).

Another family of methods that provide deterministic approximations to the posterior are based on variational inference. These cast inference as an optimization problem and minimize the KL-divergence between the approximate and true posterior. There have been many recent attempts

that have adopted this paradigm (Graves, 2011; Hernández-Lobato & Adams, 2015; Blundell et al., 2015; Kingma et al., 2015). However, most of these approaches assume a fully factorized posterior distribution over the neural network weights. We conjecture that this assumption is very restricting as the "true" posterior distribution does have some correlations among the network weights. Therefore by using a fully factorized posterior distribution the learning task becomes "harder" as there is not enough information sharing among the weights.

We therefore introduce a variational Bayesian neural network that instead of treating each element of the weight matrix independently, it treats the weight matrix *as a whole* via a matrix variate Gaussian distribution (Gupta & Nagar, 1999), i.e. a distribution over random matrices. This parametrization will significantly reduce the amount of variance-related parameters that we have to estimate: instead of estimating a separate variance for each weight we can now estimate separate variances for each row and column of the weight matrix, i.e input and output feature specific variances. This will immediately introduce correlations, and consequently information sharing, among the weights. As a result, it will allow for an easier estimation of the weight posterior uncertainty.

In addition, we will also provide a distinct relation between our model and deep (multi-output) Gaussian Processes (Damianou & Lawrence, 2013); this relation arises through the application of the "local reparametrization trick" (Kingma et al., 2015) on the matrix variate Gaussian distribution. This fact reveals an interesting property for this Bayesian neural network: we can now sample more efficiently while maintaining the properties of the original model through the introduction of pseudo-data (Snelson & Ghahramani, 2005).

2. Beyond fully factorized parameter posteriors

2.1. Matrix variate Gaussian distribution

The matrix variate Gaussian (Gupta & Nagar, 1999) is a three parameter distribution that governs a random matrix, e.g. $\mathbf{W}$:

$$p(\mathbf{W}) = \mathcal{MN}(\mathbf{M}, \mathbf{U}, \mathbf{V})$$
$$= \frac{\exp\left(-\frac{1}{2}\operatorname{tr}\left[\mathbf{V}^{-1}(\mathbf{W}-\mathbf{M})^T\mathbf{U}^{-1}(\mathbf{W}-\mathbf{M})\right]\right)}{(2\pi)^{np/2}|\mathbf{V}|^{n/2}|\mathbf{U}|^{n/2}}$$
$$(1)$$

where $\mathbf{M}$ is a $r \times c$ matrix that is the mean of the distribution, $\mathbf{U}$ is a $r \times r$ matrix that provides the covariance of the rows and $\mathbf{V}$ is a $c \times c$ matrix that governs the covariance of the columns of the matrix. According to (Gupta & Nagar, 1999) this distribution is essentially a multivariate

Gaussian distribution where:

$$p(\operatorname{vec}(\mathbf{W})) = \mathcal{N}(\operatorname{vec}(\mathbf{M}), \mathbf{V} \otimes \mathbf{U})$$

where $\operatorname{vec}(\cdot)$ is the vectorization operator (i.e. stacking the columns into a single vector) and $\otimes$ is the Kronecker product. Despite the fact that the matrix variate Gaussian is a simple generalization of the multivariate case it provides us a straightforward way to separate the correlations among the rows and columns of the matrix, which implicitly affects the correlations among the input and output hidden units.

2.2. Variational inference with matrix variate Gaussian posteriors

For the following we will assume that each input to a layer is augmented with an extra dimension containing $\mathbf{1}$'s so as to account for the biases and thus we are only dealing with weights $\mathbf{W}$ on this expanded input. In order to obtain a matrix variate Gaussian posterior distribution for these weights we can work in a pretty straightforward way: the derivation is similar to (Graves, 2011; Kingma & Welling, 2014; Blundell et al., 2015; Kingma et al., 2015). Let $p_\theta(\mathbf{W}), q_\phi(\mathbf{W})$ be a matrix variate Gaussian prior and posterior distribution with parameters θ, ϕ respectively and $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$ be the training data sampled from the empirical distribution $\tilde{p}(\mathbf{x}, \mathbf{y})$. Then the following lower bound on the marginal log-likelihood can be derived:

$$\mathcal{L}(\phi; \theta) = \mathbb{E}_{\tilde{p}(\mathbf{x}, \mathbf{y})}[\log p(\mathbf{Y}|\mathbf{X})] \leq$$
$$\mathbb{E}_{\tilde{p}(\mathbf{x}, \mathbf{y})}\left[\int q_\phi(\mathbf{W}) \log \frac{p_\theta(\mathbf{W})p(\mathbf{Y}|\mathbf{X}, \mathbf{W})}{q_\phi(\mathbf{W})} d\mathbf{W}\right]$$
$$= \mathbb{E}_{\tilde{p}(\mathbf{x}, \mathbf{y})}\left[\mathbb{E}_{q_\phi(\mathbf{W})}\left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W})\right]-\right.$$
$$\left. - KL(q_\phi(\mathbf{W})||p_\theta(\mathbf{W}))\right] \qquad (2)$$

Following (Graves, 2011; Blundell et al., 2015; Kingma et al., 2015) we will refer to $L_{(\mathbf{X},\mathbf{Y})} = \mathbb{E}_{q_\phi(\mathbf{W})}\left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W})\right]$ as the *expected log-likelihood* and to $L_c = -KL(q_\phi(\mathbf{W})||p_\theta(\mathbf{W}))$ as the *complexity loss*. To estimate $L_{(\mathbf{X},\mathbf{Y})}$ we will use simple Monte Carlo integration along with the "reparametrization trick" (Kingma & Welling, 2014; Rezende et al., 2014):

$$\mathbb{E}_{q_\phi(\mathbf{W})}\left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W})\right] = \frac{1}{L}\sum_{i=1}^{L}\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}^{(l)})$$
$$(3)$$

$$\mathbf{W}^{(l)} = \mathbf{M} + \mathbf{U}^{\frac{1}{2}}\mathbf{E}^{(l)}\mathbf{V}^{\frac{1}{2}}$$
$$\mathbf{E}^{(l)} \sim \mathcal{MN}(\mathbf{0}, \mathbf{I}, \mathbf{I}) \quad (\text{i.e. } E_{ij} \sim \mathcal{N}(0, 1))$$

As for the complexity loss L_c; due to the relation with the multivariate Gaussian we can still calculate the KL-

divergence between the matrix variate Gaussian prior and posterior efficiently in closed form.

However, maintaining a full covariance over the rows and columns of the weight matrix is both memory and computationally intensive. In order to still have a tractable model we approximate each of the covariances with a diagonal matrix (i.e. independent rows and columns) for simplicity[1]. This approximation provides a per-layer parametrization that requires significantly less parameters than a simple fully factorized Gaussian posterior: we have a total of $(n_{in} \times n_{out}) + n_{in} + n_{out}$ parameters, whereas a fully factorized Gaussian posterior has $2(n_{in} \times n_{out})$ per layer. This in turn makes the posterior uncertainty estimation easier as there are both fewer parameters to learn and also "information sharing" among the weights due to the induced correlations.

With this diagonal approximation to the covariance matrices the KL-divergence between the matrix variate Gaussian posterior $q(\mathbf{W}|\mathbf{M}, \boldsymbol{\sigma}_r^2\mathbf{I}, \boldsymbol{\sigma}_c^2\mathbf{I})$ and a standard isotropic matrix variate Gaussian prior $p(\mathbf{W}|\mathbf{0}, \mathbf{I}, \mathbf{I})$ for a matrix of size $r \times c$ corresponds to the following simple expression:

$$KL(q(\mathbf{W}|\mathbf{M}, \boldsymbol{\sigma}_r^2\mathbf{I}, \boldsymbol{\sigma}_c^2\mathbf{I})||p(\mathbf{W}|\mathbf{0}, \mathbf{I}, \mathbf{I})) =$$
$$\frac{1}{2}\left(\left(\sum_{i=1}^{r}\sigma_{r_i}^2\right)\left(\sum_{j=1}^{c}\sigma_{c_j}^2\right) + \|\mathbf{M}\|_F^2 - rc \right.$$
$$\left. - c\left(\sum_{i=1}^{r}\log\sigma_{r_i}^2\right) - r\left(\sum_{j=1}^{c}\log\sigma_{c_j}^2\right)\right) \qquad (4)$$

The derivation for arbitrary covariance matrices is given in the appendix.

2.3. Deep matrix variate Bayesian nets as deep multi-output Gaussian Processes

Directly using the expected log-likelihood estimator 3 yields increased variance and higher memory requirements, as it was pointed in (Kingma et al., 2015). Fortunately, similarly to a standard multivariate Gaussian, the inner product between a matrix and a matrix variate Gaussian is again a matrix variate Gaussian (Gupta & Nagar, 1999) and as a result we can use the "local reparametrization trick" (Kingma et al., 2015). Let $\mathbf{A}_{M \times r}$, with $M \le r$, be a minibatch of M inputs with dimension r that is the input to a network layer; the inner product $\mathbf{B}_{M \times c} = \mathbf{AW}$, where $\mathbf{W}$ is a matrix variate variable with size $r \times c$, has the following

distribution:

$$p(\mathbf{B}|\mathbf{A}) = \mathcal{MN}(\mathbf{AM}, \mathbf{AUA}^T, \mathbf{V}) \qquad (5)$$

As we can see, after the inner product the inputs $\mathbf{A}$ become *dependent* due to the non-diagonal row covariance $\mathbf{AUA}^T$. Furthermore, the resulting matrix variate Gaussian maintains the same marginalization properties as a multivariate Gaussian. More specifically, if we marginalize out a row from the $\mathbf{B}$ matrix, then the resulting distribution depends only on the remaining inputs, i.e. it corresponds to simply removing that particular input from the minibatch. This fact exposes a Gaussian Process (Rasmussen, 2006) nature for the output $\mathbf{B}$ of each layer.

To make the connection even clearer we can consider an example similar to the one presented in (Gal & Ghahramani, 2015). Let's assume that we have a neural network with one hidden layer and one output layer. Furthermore, let $\mathbf{X}$, with dimensions $N \times D_x$, be the input to the network and $\mathbf{Y}$, with dimensions $N \times D_y$, be the target variable. Finally, let's also assume that for the first weight matrix $p_{\theta_1}(\mathbf{W}_1) = \mathcal{MN}(\mathbf{0}, \mathbf{U}_1^0, \mathbf{V}_1^0)$ and that for the second weight matrix $p_{\theta_2}(\mathbf{W}_2) = \mathcal{MN}(\mathbf{0}, \mathbf{U}_2^0, \mathbf{V}_2^0)$. Now we can define the following generative model:

$$\mathbf{W}_1 \sim \mathcal{MN}(\mathbf{0}, \mathbf{U}_1^0, \mathbf{V}_1^0); \quad \mathbf{W}_2 \sim \mathcal{MN}(\mathbf{0}, \mathbf{U}_2^0, \mathbf{V}_2^0)$$
$$\mathbf{B} = \mathbf{XW}_1; \quad \mathbf{F} = \psi(\mathbf{B})\mathbf{W}_2$$
$$\mathbf{Y} \sim \mathcal{MN}(\mathbf{F}, \tau^{-1}\mathbf{I}_N, \mathbf{I}_{D_y})$$

where $\psi(\cdot)$ is a nonlinearity and $\mathcal{MN}(\mathbf{F}, \tau^{-1}\mathbf{I}_N, \mathbf{I}_{D_y})$ corresponds to an independent multivariate Gaussian over each column of $\mathbf{Y}$, i.e. $p(\mathbf{Y}|\mathbf{X}) = \prod_{i=1}^{D_y}\mathcal{N}(\mathbf{y}_i|\mathbf{f}_i, \tau^{-1}\mathbf{I}_N)$, where $\mathbf{f}_i$ is a column of $\mathbf{F}$[2]. Now if we make use of the matrix variate Gaussian property 5 we have that the generative model becomes:

$$\mathbf{B}|\mathbf{X} \sim \mathcal{MN}(\mathbf{0}, \mathbf{XU}_1^0\mathbf{X}^T, \mathbf{V}_1^0)$$
$$\mathbf{F}|\mathbf{B} \sim \mathcal{MN}(\mathbf{0}, \psi(\mathbf{B})\mathbf{U}_2^0\psi(\mathbf{B})^T, \mathbf{V}_2^0)$$
$$\mathbf{Y}|\mathbf{F} \sim \mathcal{MN}(\mathbf{F}, \tau^{-1}\mathbf{I}_N, \mathbf{I}_{D_y})$$

or else equivalently:

$$\text{vec}(\mathbf{B})|\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \hat{\mathbf{K}}_{\theta_1}(\mathbf{X}, \mathbf{X}))$$
$$\text{vec}(\mathbf{F})|\mathbf{B} \sim \mathcal{N}(\mathbf{0}, \hat{\mathbf{K}}_{\theta_2}(\mathbf{B}, \mathbf{B}))$$
$$\text{vec}(\mathbf{Y})|\mathbf{F} \sim \mathcal{N}(\text{vec}(\mathbf{F}), \tau^{-1}(\mathbf{I}_N \otimes \mathbf{I}_{D_y}))$$

where $\hat{\mathbf{K}}_\theta(\mathbf{z}_1, \mathbf{z}_2) = \mathbf{K}_{out} \otimes \mathbf{K}_{in}(\mathbf{z}_1, \mathbf{z}_2; \mathbf{U}) = \mathbf{V} \otimes \left(\psi(\mathbf{z}_1)\mathbf{U}\psi(\mathbf{z}_2)^T\right)$[3]. In other words, we have a composition

---

[1]Note that we could also easily use rank-1 matrices with diagonal corrections (Rezende et al., 2014) and increase the flexibility of our posterior. For example we could apply the rank-1 approximation to the square root of the covariance matrix (as we directly use it for sampling), i.e. $\mathbf{C}^{\frac{1}{2}} = \mathbf{D}_c + \mathbf{u}\mathbf{u}^T$ where $\mathbf{D}_c$ is a diagonal matrix with positive elements.

[2]Note that this is just a simplifying assumption and not a limitation for our method. We could instead also model the correlations among the output variables $\mathbf{Y}$ if we used a full covariance $\mathbf{C}_{D_y}$ instead of $\mathbf{I}_{D_y}$.

[3]$\psi(\cdot)$ is the identity function for the input layer.

of GPs where the covariance of each GP is governed by a kernel function of a specific form; it is the kroneker product of a global output and an input dependent kernel function, where the latter is composed of fixed dimension nonlinear basis functions (the inputs to each layer) weighted by their covariance. Essentially this kernel provides a distribution for each layer that is similar to a (correlated) multi-output GP, which was previously explored in the context of shallow GPs (Yu et al., 2006; Bonilla et al., 2007a;b). Therefore, in order to obtain the marginal likelihood of the targets $\mathbf{Y}$ we have to marginalize over the function values $\mathbf{B}$ and $\mathbf{F}$, which results into a deep GP (Damianou & Lawrence, 2013) with the aforementioned kernel function for each GP:

$$\log p(\mathbf{Y}|\mathbf{X}) =$$
$$\log \mathbb{E}_{p_{\theta_1}(\mathbf{B}|\mathbf{X})p_{\theta_2}(\mathbf{F}|\mathbf{B})} \left[\mathcal{N}(\mathrm{vec}(\mathbf{F}), \tau^{-1}(\mathbf{I}_N \otimes \mathbf{I}_{D_y})) \right]$$

A similar scenario was also considered theoretically in (Duvenaud et al., 2014). Now in order to obtain the posterior distribution of the parameters $\mathbf{W}$ we will perform variational inference. We place a matrix variate Gaussian posterior distribution over the weights of the neural network, i.e. $q_{\phi_1}(\mathbf{W}_1)q_{\phi_2}(\mathbf{W}_2) = \mathcal{MN}(\mathbf{M}_1, \mathbf{U}_1, \mathbf{V}_1)\mathcal{MN}(\mathbf{M}_2, \mathbf{U}_2, \mathbf{V}_2)$, and the marginal likelihood lower bound in eq. 2 becomes:

$$\mathcal{L}(\phi_{1,2}, \theta_{1,2}) \leq$$
$$\mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})} \left[\mathbb{E}_{q_{\phi_1}(\mathbf{W}_1)q_{\phi_2}(\mathbf{W}_2)} \left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2) \right] - \right.$$
$$\left. - \sum_{i=1}^{2} KL(q_{\phi_i}(\mathbf{W}_i)||p_{\theta_i}(\mathbf{W}_i)) \right]$$
$$= \mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})} \left[L_{(\mathbf{X},\mathbf{Y})}(\phi_1, \phi_2) + \sum_{i=1}^{2} L_c(\phi_i, \theta_i) \right] \quad (6)$$

Noting that $\mathbf{Y}$ only depends on $\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2$ through $\mathbf{F} = \psi(\mathbf{B})\mathbf{W}_2 = \psi(\mathbf{X}\mathbf{W}_1)\mathbf{W}_2$, and applying the reparametrization trick, i.e.

$$\int q_{\phi_1}(\mathbf{W}_1)q_{\phi_2}(\mathbf{W}_2) \log p(\mathbf{Y}|\mathbf{F}(\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2))d\mathbf{W}_{1,2} =$$
$$\int \tilde{q}_{\phi_1}(\mathbf{B}|\mathbf{X})\tilde{q}_{\phi_2}(\mathbf{F}|\mathbf{B}) \log p(\mathbf{Y}|\mathbf{F})d\mathbf{B}d\mathbf{F}$$

where (using 5),

$$\tilde{q}_{\phi_1}(\mathbf{B}|\mathbf{X}) = \mathcal{N}(\mathrm{vec}(\boldsymbol{\mu}_{\phi_1}(\mathbf{X})), \hat{\mathbf{K}}_{\phi_1}(\mathbf{X}, \mathbf{X}))$$
$$\tilde{q}_{\phi_2}(\mathbf{F}|\mathbf{B}) = \mathcal{N}(\mathrm{vec}(\boldsymbol{\mu}_{\phi_2}(\mathbf{B})), \hat{\mathbf{K}}_{\phi_2}(\mathbf{B}, \mathbf{B}))$$

where $\phi_1 = (\mathbf{M}_1, \mathbf{U}_1, \mathbf{V}_1)$, $\phi_2 = (\mathbf{M}_2, \mathbf{U}_2, \mathbf{V}_2)$ are the variational parameters and $\boldsymbol{\mu}_\phi(\mathbf{z}) = \psi(\mathbf{z})\mathbf{M}$ is the mean function. As we can see, $\tilde{q}_{\phi_1}(\mathbf{B}|\mathbf{X}), \tilde{q}_{\phi_2}(\mathbf{F}|\mathbf{B})$ can be considered as approximate posterior GP functions while the

local reparametrization trick provides the connection between the primal and dual GP view of the model. The variational objective thus becomes:

$$\mathcal{L}(\phi_{1,2}, \theta_{1,2}) \leq \mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})} \left[\mathbb{E}_{\tilde{q}_{\phi_1}(\mathbf{B}|\mathbf{X})\tilde{q}_{\phi_2}(\mathbf{F}|\mathbf{B})} \left[\log p(\mathbf{Y}|\mathbf{F}) \right] + \right.$$
$$\left. + \sum_{i=1}^{2} L_c(\phi_i, \theta_i) \right] \quad (7)$$

2.4. Efficient sampling and pseudo-data

Sampling distribution 5 for every layer is however computationally intensive as we have to calculate the square root of the row covariance $\mathbf{K}_{in}(\mathbf{A}, \mathbf{A}; \mathbf{U}) = \mathbf{A}\mathbf{U}\mathbf{A}^T$ (which has a cubic cost w.r.t. the amount of datapoints in $\mathbf{A}$) every time. A simple solution is to only use its diagonal for sampling. This corresponds to samples from the marginal distribution of each pre-activation latent variable $\mathbf{b}_i$ in the minibatch $\mathbf{A}$. More specifically, we have that $\mathbf{b}_i$ follows a multivariate Gaussian distribution where the covariance is controlled by two sources: the local scalar row variance (i.e. per datapoint feature correlations) and the global column, i.e. pre-activation latent variable (or target variable in the case of the output layer), covariance: $p(\mathbf{b}_i|\mathbf{a}_i) = \mathcal{N}(\mathbf{a}_i\mathbf{M}, (\mathbf{a}_i\mathbf{U}\mathbf{a}_i^T) \odot \mathbf{V})$.

Despite its simplicity however this approach does not use the Gaussian Process nature of our model. In order to fully utilize this property we adopt an idea from the GP literature: the concept of pseudo-data (Snelson & Ghahramani, 2005). More specifically, we introduce pseudo inputs $\tilde{\mathbf{A}}$ and pseudo outputs $\tilde{\mathbf{B}}$ for each layer in the network and sample the distribution of each pre-activation latent variable $\mathbf{b}_i$ conditioned on the pseudo-data:

$$p(\mathbf{b}_i|\mathbf{a}_i, \tilde{\mathbf{A}}, \tilde{\mathbf{B}}) = \mathcal{N}\left(\mathbf{a}_i\mathbf{M} + \boldsymbol{\sigma}_{12}^T\boldsymbol{\Sigma}_{11}^{-1}(\tilde{\mathbf{B}} - \tilde{\mathbf{A}}\mathbf{M}), \right.$$
$$\left. (\sigma_{22} - \boldsymbol{\sigma}_{12}^T\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\sigma}_{12}) \odot \mathbf{V} \right) \quad (8)$$

where each of the covariance terms can be estimated as:

$$\boldsymbol{\Sigma}_{11} = \tilde{\mathbf{A}}\mathbf{U}\tilde{\mathbf{A}}^T; \boldsymbol{\sigma}_{12} = \tilde{\mathbf{A}}\mathbf{U}\mathbf{a}_i^T; \sigma_{22} = \mathbf{a}_i\mathbf{U}\mathbf{a}_i^T \quad (9)$$

As can be seen, the pseudo-data directly affect the distribution of each pre-activation latent variable: if the inputs are similar to the pseudo-inputs then the variance of the latent variable $\mathbf{b}_i$ decreases and the mean is shifted towards the pseudo-data. This allows each layer in the network to be more certain in particular regions of the input space. However, if the inputs are not similar to the pseudo-inputs then the distribution of $\mathbf{b}_i$ depends mostly on the parameters of the underlying matrix variate Gaussian posterior.

It should be noted that the amount of pseudo-data for each layer N_p should be $N_p < D$, where D is the dimensionality of the input, as we are using a linear kernel for the row

covariance (that becomes non-linear via the neural network nonlinearities) that has finite rank D. This enforces that the pseudo-data combined with a real input $\mathbf{a}_i$ provide a positive definite kernel $\hat{\mathbf{K}}$ for the joint Gaussian output distribution $p(\tilde{\mathbf{B}}, \mathbf{b}_i | \hat{\mathbf{A}}, \mathbf{a}_i)$. Furthermore, we also "dampen" $\boldsymbol{\Sigma}_{11}$ by adding to it a small diagonal matrix $\sigma^2 \mathbf{I}$ where $\sigma^2 = 1e^{-8}$. This corresponds to assuming "noisy" pseudo-observations $\tilde{\mathbf{B}}$ (Rasmussen, 2006) (where the noise is i.i.d. from $\mathcal{N}(0, \sigma^2)$) which helps avoiding numerical instabilities during optimization (this is particularly helpful with limited precision floating-point).

At first glance it might seem that we now overparametrize each neural network layer, however in practice this does not seem to be the case. From our experience relatively few pseudo-data per layer (compared to the input dimensionality) are necessary for increased performance. This still yields less parameters than fully factorized Gaussian posteriors. In addition, note that with the pseudo data formulation we could also assume that the weight posterior has zero mean $\mathbf{M} = \mathbf{0}$ (in GP parlance this corresponds to removing the mean function); this would reduce the number of parameters even further and still provide a useful model. This assumption leads to sampling the following distribution:

$$p(\mathbf{b}_i | \mathbf{a}_i, \tilde{\mathbf{A}}, \tilde{\mathbf{B}}) =$$
$$\mathcal{N}\left(\boldsymbol{\sigma}_{12}^T \boldsymbol{\Sigma}_{11}^{-1} \tilde{\mathbf{B}}, \left(\sigma_{22} - \boldsymbol{\sigma}_{12}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\sigma}_{12}\right) \odot \mathbf{V}\right) \quad (10)$$

Finally, since we want a fully Bayesian model, we also place fully factorized multiplicative Gaussian posteriors on both $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ along with log-uniform priors, as it was described in (Kingma et al., 2015). The final form of the bound 7 with the inclusion of the pseudo-data is:

$$\mathcal{L}(\phi_{1,2}, \theta_{1,2}) \leq$$
$$\mathbb{E}_{\tilde{p}(\mathbf{x}, \mathbf{y})}\left[\mathbb{E}_{q_{\phi_1}(\mathbf{B}, \tilde{\mathbf{A}}_1, \tilde{\mathbf{B}}_1 | \mathbf{X}) q_{\phi_2}(\mathbf{F}, \tilde{\mathbf{A}}_2, \tilde{\mathbf{B}}_2 | \mathbf{B})}\left[\log p(\mathbf{Y} | \mathbf{F})\right] +$$
$$+ \sum_{i=1}^{2} L_c(\phi_i, \theta_i)\right] \quad (11)$$

where:

$$q_{\phi_i}(\mathbf{B}, \tilde{\mathbf{A}}, \tilde{\mathbf{B}} | \mathbf{X}) = \tilde{q}_{\phi_i}(\mathbf{B} | \mathbf{X}, \tilde{\mathbf{A}}, \tilde{\mathbf{B}}) q_{\phi_i}(\tilde{\mathbf{A}}) q_{\phi_i}(\tilde{\mathbf{B}})$$
$$L_c(\phi_i, \theta_i) = -KL(q(\mathbf{W}_i) || p(\mathbf{W}_i)) -$$
$$- KL(q(\tilde{\mathbf{A}}_i) || p(\tilde{\mathbf{A}}_i)) - KL(q(\tilde{\mathbf{B}}_i) || p(\tilde{\mathbf{B}}_i))$$

where now ϕ_i, θ_i also include the parameters of the distributions of the pseudo-data. The KL-divergence for these can be found at (Kingma et al., 2015). We can thus readily optimize the marginal likelihood lower bound of eq. 11 w.r.t. the parameters of the posterior and the pseudo data with stochastic gradient ascent.

2.5. Computational complexity

A typical variational Bayesian neural network with a fully factorized Gaussian posterior sampled "locally" (Kingma et al., 2015) has asymptotic per-datapoint time complexity $\mathcal{O}(D^2)$ for the mean and variance in each layer, where D is the input/output dimensionality. Our model adds the extra cost of inverting $\boldsymbol{\Sigma}_{11}^{-1}$, that has cubic complexity with respect to the amount of pseudo-data M for each layer. Therefore the asymptotic time complexity is $\mathcal{O}(D^2 + M^3)$ and since usually $M << D$, this does not incur a significantly extra computational cost.

3. Related work

(Graves, 2011) firstly introduced a practical way of variational inference for neural networks. Despite the fact that the proposed (biased) estimator had good performance on a recurrent neural network task, it was not as effective on the regression task of (Hernández-Lobato & Adams, 2015). (Blundell et al., 2015) proposed to use an alternative unbiased estimator that samples on the relatively high variance weight space but nonetheless provided good performance on a reinforcement learning task. (Kingma et al., 2015) subsequently presented the "local reparametrization trick", which makes use of Gaussian properties so as to sample in the function space, i.e. the hidden units. This provides both reduced memory requirements as well as reduced variance for the expected log-likelihood estimator. However, for their model they still use a fully factorized posterior distribution that doubles the amount of parameters in each layer and does not allow the incorporation of pseudo-data.

(Gal & Ghahramani, 2015) also provides connections between Bayesian neural networks and deep Gaussian processes, but they only consider independent Gaussians for each column of the weight matrix (which in our case correspond to $p(\mathbf{W}) = \mathcal{MN}(\mathbf{M}, \sigma^2 \mathbf{I}, \mathbf{I})$) and do not model the variances of the hidden units. Furthermore the approximating variational distribution is quite limited as it corresponds to simple Bernoulli noise and delta approximating distributions for the weight matrix: it is a mixture of two delta functions for each column of the weight matrix, one at zero and the other at the mean of the Gaussian. This is in contrast to our model where we can explicitly learn the (possibly non-diagonal) covariance for both the input and output dimensions of each layer through the matrix variate Gaussian posterior. In addition, sampling is done in the weight space and not the function space as in our model, thus preventing the use of pseudo-data.

Finally, (Hernández-Lobato & Adams, 2015) also assume fully factorized posterior distributions and uses Expectation Propagation (Minka, 2001) instead of variational in-

ference. Closed form approximations bypass the need for sampling in the model, which in turn makes it easier to converge. However their derivation is limited to rectified linear nonlinearities and regression problems, thus limiting the applicability of their model. Furthermore, since each datapoint is treated as new during the update of the parameters, special care has to be given so as to not perform a lot of passes through the dataset since this will in general shrink the variances of the weights of the network.

4. Experiments

All of the models were coded in Theano (Bergstra et al., 2010) and optimization was done with Adam (Kingma & Ba, 2015), using the default hyper-parameters and temporal averaging. We parametrized the prior for each weight matrix as $p(\mathbf{W}) = \mathcal{MN}(\mathbf{0}, \mathbf{I}, \mathbf{I})$ unless stated otherwise. Following (Hernández-Lobato & Adams, 2015) we also divide the input to each layer (both real and pseudo) by the square root of its dimensionality so as to keep the scale of the output (before the nonlinearity) independent of the incoming connections. We used rectified linear units (Nair & Hinton, 2010) (ReLU) and we initialized the mean of each matrix variate Gaussian via the scheme proposed in (He et al., 2015). For the initialization of the pseudo-data we sampled the entries of $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$ from $\mathcal{U}[-0.01, 0.01]$. We used one posterior sample to estimate the expected log-likelihood before we update the parameters.

We test under two different scenarios: regression and classification. For the regression task we experimented with the UCI (Asuncion & Newman, 2007) datasets that were used in "Probabilistic Backpropagation" (PBP) (Hernández-Lobato & Adams, 2015) and in "Dropout as a Bayesian Approximation" (Gal & Ghahramani, 2015). For the classification task we evaluated our model on the permutation invariant MNIST benchmark dataset, so as to compare against other popular neural network models.

Finally we also performed a toy regression experiment on the same artificially generated data as (Hernández-Lobato & Adams, 2015), so that we can similarly visualize the predictive distribution that our model provides.

4.1. Regression experiments

For the regression experiments we followed a similar experimental protocol with (Hernández-Lobato & Adams, 2015): we randomly keep 90% of the dataset for training and use the remaining to test the performance. This process is repeated 20 times (except from the "Protein" dataset where it is performed 5 times and the "Year" dataset where it is performed once) and the average values along with their standard errors are reported at Table 1. Following (Hernández-Lobato & Adams, 2015) we also introduce

a Gamma prior, $p(\tau) = Gam(a_0 = 6, b_0 = 6)$ and posterior $q(\tau) = Gam(a_1, b_1)$ for the precision of the Gaussian likelihood and we parametrized the matrix variate Gaussian prior for each layer as $p(\mathbf{W}) = \mathcal{MN}(\mathbf{0}, \tau_r^{-1}\mathbf{I}, \tau_c^{-1}\mathbf{I})$, where $p(\tau_r), p(\tau_c) = Gam(a_0 = 1, b_0 = 0.5)$ and $q(\tau_r)q(\tau_c) = Gam(a_r, b_r)Gam(a_c, b_c)$[4]. We optimized $a_1, b_1, a_r, b_r, a_c, b_c$ along with the remaining variational parameters. We do not use a validation set and instead train the networks up until convergence in the training set. We use one hidden layer of 50 units for all of the datasets, except for the larger "Protein" and "Year" datasets where we use 100 units. We normalized the inputs $\mathbf{x}$ of the network to zero mean and unit variance but we did not normalize the targets $\mathbf{y}$. Instead we parametrized the network output as $\mathbf{y} = f(\mathbf{x}) \odot \boldsymbol{\sigma}_y + \boldsymbol{\mu}_y$ where $f(\cdot)$ represents the neural network and $\boldsymbol{\mu}_y, \boldsymbol{\sigma}_y$ are the, per-dimension, mean and standard deviation of the target variable, estimated from the training set. Similarly to (Gal & Ghahramani, 2015) we set the upper bound of the variational dropout rate to 0.005, 0.05 and we used 10 pseudo-data pairs for each layer for all of the datasets, except for the smaller "Yacht" dataset where we used 5 and the bigger "Protein" and "Year" where we used 20.

As we can see from the results at Table 1 our model provides significantly lower root mean square errors, compared to VI (Graves, 2011), PBP (Hernández-Lobato & Adams, 2015) and Dropout (Gal & Ghahramani, 2015), on all of the datasets. In addition, we also observe better performance according to the predictive log-likelihoods; our model outperforms VI and PBP on all of the datasets and is better than Dropout on 8 out of 10. These results empirically verify the effectiveness of our mdoel: with the matrix variate Gaussian posteriors along with the Gaussian Process interpretation we have a model that is very flexible and consequently can both better fit the data, and, in the case of the predictive log-likelihoods, make an accurate estimation of the predictive uncertainty.

4.2. Classification experiments

For the classification experiments we trained networks with a varying number of layers and hidden units per layer. We used the last 10000 samples of the training set as a validation set for model selection, minibatches of 100 datapoints and set the upper bound for the variational dropout rate to 0.25. We used the same amount of pseudo-data pairs for each layer, but tuned those according to the validation set performance (we set an upper bound of 150 pseudo-data pairs per layer). We did not use any kind of data augmentation or preprocessing. The results can be seen at Table 2.

---

[4]For this choice of distribution both $\mathbb{E}_{q(\tau_r)q(\tau_c)}[KL(q(\mathbf{W}|\mathbf{M}, \mathbf{U}, \mathbf{V})||p(\mathbf{W}|\mathbf{0}, \tau_r^{-1}\mathbf{I}, \tau_c^{-1}\mathbf{I}))]$ and the KL-divergence between $q(\tau_r)q(\tau_c)$ and $p(\tau_r)p(\tau_c)$ can be computed in closed form.

Dataset	Avg. Test RMSE and Std. Errors				Avg. Test LL and Std. Errors			
	VI	PBP	Dropout	VMG	VI	PBP	Dropout	VMG
Boston	4.32±0.29	3.01±0.18	2.97±0.85	**2.81±0.11**	-2.90±0.07	-2.57± 0.09	**-2.46±0.25**	-2.54±0.08
Concrete	7.19±0.12	5.67±0.09	5.23± 0.53	**4.70±0.14**	-3.39±0.02	-3.16±0.02	-3.04±0.09	**-2.98±0.03**
Energy	2.65±0.08	1.80±0.05	1.66±0.19	**1.16±0.03**	-2.39±0.03	-2.04±0.02	-1.99±0.09	**-1.45±0.03**
Kin8nm	0.10±0.00	0.10±0.00	0.10±0.00	**0.08±0.00**	0.90±0.01	0.90±0.01	0.95±0.03	**1.14±0.01**
Naval	0.01±0.00	0.01±0.00	0.01±0.00	**0.00±0.00**	3.73±0.12	3.73±0.01	3.80±0.05	**5.84±0.00**
Pow. Plant	4.33±0.04	4.12±0.03	4.02±0.18	**3.88±0.03**	-2.89±0.01	-2.84±0.01	-2.80±0.05	**-2.78±0.01**
Protein	4.84±0.03	4.73±0.01	4.36±0.04	**4.14±0.01**	-2.99±0.01	-2.97±0.00	-2.89±0.01	**-2.84±0.00**
Wine	0.65±0.01	0.64±0.01	0.62±0.04	**0.61±0.01**	-0.98±0.01	-0.97±0.01	-0.93±0.06	**-0.93±0.02**
Yacht	6.89±0.67	1.02±0.05	1.11±0.38	**0.77±0.06**	-3.43±0.16	-1.63±0.02	-1.55±0.12	**-1.29±0.02**
Year	9.034±NA	8.879±NA	8.849±NA	**8.780±NA**	-3.622±NA	-3.603±NA	**-3.588±NA**	-3.589±NA

Table 1. Average test set RMSE, predictive log-likelihood and standard errors for the regression datasets. VI, PBP and Dropout correspond to the variational inference method of (Graves, 2011), probabilistic backpropagation (Hernández-Lobato & Adams, 2015) and dropout uncertainty (Gal & Ghahramani, 2015). VMG (Variational Matrix Gaussian) corresponds to the proposed model.

Method	# layers	Test err.
Max. Likel. (Simard et al., 2003)	2×800	1.60
Dropout (Srivastava, 2013)	-	1.25
DropConnect (Wan et al., 2013)	2×800	1.20
Bayes B. SM (Blundell et al., 2015)	2×400	1.36
	2×800	1.34
	2×1200	1.32
Var. Dropout (Kingma et al., 2015)	3×150	≈ 1.42
	3×250	≈ 1.28
	3×500	≈ 1.18
	3×750	≈ 1.09
VMG	2×400	**1.15**
	3×150	1.18
	3×250	1.11
	3×500	1.08
	3×750	**1.05**

Table 2. Test errors for the permutation invariant MNIST dataset. Bayes B. SM correspond to Bayes by Backprop with the scale mixture prior and the variational dropout results are from the Variational (A) model that doesn't downscale the KL-divergence (so as to keep the comparison fair).

As we can observe our Bayesian neural network performs better than other popular neural networks models for small network sizes. For example, with only three hidden layers of 150 units it achieves 1.18% test error on MNIST, a result that is better than maximum likelihood (Simard et al., 2003), Dropout (Srivastava, 2013), DropConnect (Wan et al., 2013) and Bayes by Backprop (Blundell et al., 2015), where all of the aforementioned methods have significantly bigger architectures than our model. Furthermore, it is also significantly better than a neural network of the same size trained with variational dropout (Kingma et al., 2015). We can probably attribute this effect to the Gaussian Process property; for regular neural networks a small network size means that there are not enough parameters to learn an ef-

fective classifier. This is in contrast to our model where through the learned pseudo-data we can maintain this property and consequently increase the flexibility of the model and compensate for the lack of network size.

4.3. Toy experiment

In order to visually access the quality of the uncertainty that our model provides, we also performed an experiment on the simple toy dataset that was used in (Hernández-Lobato & Adams, 2015). We sampled 20 inputs x from $\mathcal{U}[-4, 4]$ and parametrized the target variable as $y_n = x_n^3 + \epsilon_n$ where $\epsilon_n \sim N(0, 9)$. We then fitted a neural network with matrix Gaussian posteriors (with diagonal covariance matrices), a neural network that had a fully factorized Gaussian distribution for the weights and a dropout network. All of the networks had a single hidden layer of 100 units. For our model we used two pseudo-data pairs for the input layer, four for the output layer and set the upper bound of the variational dropout rate to 0.2. The dropout rate for the dropout network was zero for the input layer and 0.2 for the hidden layer. The resulting predictive distributions (after 200 samples) can be seen in figure 1 (with three standard deviations around the mean).

As we can see the network with matrix Gaussian posteriors provides a realistic predictive distribution that seems slightly better compared to the one obtained from PBP (Hernández-Lobato & Adams, 2015). Interestingly, the simple fully factorized Gaussian (sampled with the "local reparametrization trick") neural network failed to obtain a good fit for the data as it was severely underfitting due to the limited amount of datapoints. This resulted into a very uncertain and noisy predictive distribution that vaguely captured the mean function. This effect is not observed with our model; with the Gaussian Process property we effectively increase the flexibility of our model thus allowing the weight posterior to be closer to the prior with-
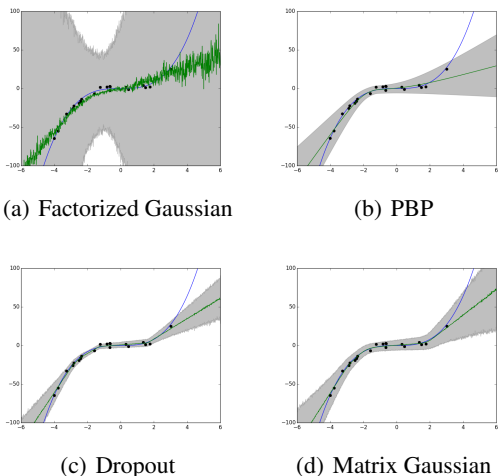
(a) Factorized Gaussian   (b) PBP

(c) Dropout   (d) Matrix Gaussian

Figure 1. Predictive distributions for the toy dataset. Grey areas correspond to ± 3 standard deviations around the mean function.

out severe loss in performance. Furthermore, we only have a handful of variance parameters to learn, which consequently provides easier and more robust posterior uncertainty estimation. Finally, it seems that the dropout network provides a predictive distribution that is slightly "overfitted" as the confidence intervals do not diverge as heavily in areas where there are no data.

5. Conclusions

We introduce a scalable variational Bayesian neural network where the parameters are governed by a probability distribution over random matrices: the matrix variate Gaussian. By utilizing properties of this distribution we can see that our model can be considered as a composition of Gaussian Processes with nonlinear kernels of a specific form. This kernel is formed from the kroneker product of two separate kernels; a global output kernel and an input specific kernel, where the latter is composed from fixed dimension nonlinear basis functions (the inputs to each layer) weighted by their covariance. We continue in exploiting this duality and introduce pseudo input-output pairs for each layer in the network, which in turn better maintain the Gaussian Process properties of our model thus increasing the flexibility of the posterior distribution.

We tested our model in two scenarios: the same regression task as PBP (Hernández-Lobato & Adams, 2015) and Dropout uncertainty (Gal & Ghahramani, 2015) and the benchmark permutation invariant MNIST classification task. For the regression task we found that our model overall achieves better RMSE and predictive log-likelihoods than VI (Graves, 2011), PBP and Dropout uncertainty. For the classification task we found that our model provides

better errors than state of the art methods for small architectures. This demonstrates the effectiveness of the Gaussian Process property; with the pseudo-data we increase the flexibility of our model thus countering the fact that we have a limited capacity neural network.

Finally, we also empirically verified the quality of the predictive distribution that our model provides on the same toy experiment as PBP (Hernández-Lobato & Adams, 2015).

References

Ahn, Sungjin, Balan, Anoop Korattikara, and Welling, Max. Bayesian posterior sampling via stochastic gradient fisher scoring. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

Asuncion, Arthur and Newman, David. Uci machine learning repository, 2007.

Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, pp. 3. Austin, TX, 2010.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015.

Bonilla, Edwin V, Agakov, Felix V, and Williams, Christopher. Kernel multi-task learning using task-specific features. In *International Conference on Artificial Intelligence and Statistics*, pp. 43–50, 2007a.

Bonilla, Edwin V, Chai, Kian M, and Williams, Christopher. Multi-task gaussian process prediction. In *Advances in neural information processing systems*, pp. 153–160, 2007b.

Damianou, Andreas C. and Lawrence, Neil D. Deep gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, pp. 207–215, 2013.

Duvenaud, David K., Rippel, Oren, Adams, Ryan P., and Ghahramani, Zoubin. Avoiding pathologies in very deep networks. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, pp. 202–210, 2014.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2015.

Graves, Alex. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

Gupta, Arjun K and Nagar, Daya K. *Matrix variate distributions*, volume 104. CRC Press, 1999.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. 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.

Hernández-Lobato, José Miguel and Adams, Ryan. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1861–1869, 2015.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR), San Diego*, 2015.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014.

Kingma, Diederik P, Salimans, Tim, and Welling, Max. Variational dropout and the local reparametrization trick. *Advances in Neural Information Processing Systems*, 2015.

Minka, Thomas P. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pp. 362–369. Morgan Kaufmann Publishers Inc., 2001.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

Neal, Radford M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Osband, Ian, Blundell, Charles, Pritzel, Alexander, and Van Roy, Benjamin. Deep exploration via bootstrapped dqn. *arXiv preprint arXiv:1602.04621*, 2016.

Rasmussen, Carl Edward. Gaussian processes for machine learning. 2006.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1278–1286, 2014.

Simard, Patrice Y, Steinkraus, Dave, and Platt, John C. Best practices for convolutional neural networks applied to visual document analysis. pp. 958. IEEE, 2003.

Snelson, Edward and Ghahramani, Zoubin. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pp. 1257–1264, 2005.

Srivastava, Nitish. Improving neural networks with dropout, 2013.

Wan, Li, Zeiler, Matthew, Zhang, Sixin, Cun, Yann L, and Fergus, Rob. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1058–1066, 2013.

Welling, Max and Teh, Yee W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.

Yu, Kai, Chu, Wei, Yu, Shipeng, Tresp, Volker, and Xu, Zhao. Stochastic relational models for discriminative link prediction. In *Advances in neural information processing systems*, pp. 1553–1560, 2006.