DEEP BAYESIAN NEURAL NETS AS DEEP MATRIX GAUSSIAN PROCESSES

Christos Louizos
AMLAB, Informatics Institute, University of Amsterdam
c.louizos@uva.nl

Max Welling
AMLAB, Informatics Institute, University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
m.welling@uva.nl

ABSTRACT

We show that by employing a distribution over random matrices, the matrix variate Gaussian (Gupta & Nagar, 1999), for the neural network parameters we can obtain a Gaussian Process (Rasmussen, 2006) interpretation for each neural network layer after the application of the "local reprarametrization trick" (Kingma et al., 2015). This provides a nice duality between deep Bayesian neural networks and deep Gaussian Processes (Damianou & Lawrence, 2013), a property that was also shown by Gal & Ghahramani (2015). We show that we can borrow ideas from the Gaussian Process literature so as to better exploit the properties of such a model. We empirically verified this model on a regression task.

1 MATRIX-VARIATE GAUSSIAN

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

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

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 over the "flattened" matrix $\mathbf{W}$: $p(\operatorname{vec}(\mathbf{W})) = \mathcal{N}(\operatorname{vec}(\mathbf{M}), \mathbf{V} \otimes \mathbf{U})$, where $\operatorname{vec}(\cdot)$ is the vectorization operator (i.e. stacking the columns into a single vector) and $\otimes$ is the Kronecker product.

2 BAYESIAN NEURAL NETS WITH MATRIX-VARIATE GAUSSIANS

For the following derivation 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 will perform variational inference and 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:

$$\mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log p(\mathbf{Y}|\mathbf{X})] \leq \mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}\left[\mathbb{E}_{q_\phi(\mathbf{W})}\left[\log p(\mathbf{Y}|\mathbf{X},\mathbf{W})\right] - KL(q_\phi(\mathbf{W})||p_\theta(\mathbf{W}))\right] \tag{2}$$

Furthermore, by approximating the covariance matrices for each matrix variate Gaussian we can achieve a more efficient layer parametrization; for a rank-1 approximation with diagonal corrections (Rezende et al., 2014) we have a total of $n_{in} \times n_{out} + 2(n_{in} + n_{out})$ parameters whereas with a fully factorized Gaussian posterior we have $2(n_{in} \times n_{out})$. In addition, due to the relation with the multivariate normal, the KL-divergence can still be calculated efficiently in closed form.

3 DEEP MATRIX VARIATE GAUSSIAN PROCESSES

Directly using the expected log-likelihood estimator of eq. 2 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 $M \leq r$ 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:

$$p(\mathbf{B}|\mathbf{A}) = \mathcal{MN}(\mathbf{AM}, \mathbf{AUA}^T, \mathbf{V}) \tag{3}$$

As we 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, i.e. marginalizing out a row from $\mathbf{B}$ corresponds to simply removing that particular datapoint from the minibatch. This fact exposes a Gaussian Process (Rasmussen, 2006) (GP) interpretation for the output $\mathbf{B}$.

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:

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

where $\sigma(\cdot)$ is an elementwise 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 $\mathbf{F}$. Now if we make use of the matrix variate Gaussian property 3 we have that the generative model becomes:

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

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 $\hat{\mathbf{K}}_\theta(\mathbf{z}_1, \mathbf{z}_2) = \mathbf{K}_{out} \otimes \mathbf{K}_{in}(\mathbf{z}_1, \mathbf{z}_2; \mathbf{U}) = \mathbf{V} \otimes \left(\sigma(\mathbf{z}_1)\mathbf{U}\sigma(\mathbf{z}_2)^T\right)$[1]. In other words, we have a composition 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). Now 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:

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

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

[1] $\sigma(\cdot)$ is the identity function for the input layer.

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:

$$\mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log p(\mathbf{Y}|\mathbf{X})] \leq \mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}\left[\mathbb{E}_{q_{\phi_1}(\mathbf{W}_1)q_{\phi_2}(\mathbf{W}_2)}\left[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2)\right] - \sum_{i=1}^{2} KL(q_{\phi_i}(\mathbf{W}_i)||p_{\theta_i}(\mathbf{W}_i))\right] \tag{4}$$

Noting that $\mathbf{Y}$ only depends on $\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2$ through $\mathbf{F} = \sigma(\mathbf{B})\mathbf{W}_2 = \sigma(\mathbf{X}\mathbf{W}_1)\mathbf{W}_2$, 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 3),

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

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, $\boldsymbol{\mu}_\phi(\mathbf{z}) = \sigma(\mathbf{z})\mathbf{M}$ is the mean function and $\sigma(\cdot)$ is the identity for the input layer. 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:

$$\mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log p(\mathbf{Y}|\mathbf{X})] \leq \mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}\left[\mathbb{E}_{\tilde{q}_{\phi_1}(\mathbf{B}|\mathbf{X})\tilde{q}_{\phi_2}(\mathbf{F}|\mathbf{B})}\left[\log p(\mathbf{Y}|\mathbf{F})\right] - \sum_{i=1}^{2} KL(q_{\phi_i}(\mathbf{W}_i)||p_{\theta_i}(\mathbf{W}_i))\right] \tag{5}$$

However, sampling distribution 3 for every layer is computationally intensive, as we have to calculate the square root of the row covariance $\mathbf{K}(\mathbf{A}, \mathbf{A}) = \mathbf{A}\mathbf{U}\mathbf{A}^T$ 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: $p(\mathbf{b}_i|\mathbf{a}_i) = \mathcal{N}(\mathbf{a}_i\mathbf{M}, \left(\mathbf{a}_i\mathbf{U}\mathbf{a}_i^T\right)$. Despite its simplicity however this approach does not use the Gaussian Process nature of this 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 can introduce pseudo inputs $\tilde{\mathbf{A}}$ and pseudo outputs $\tilde{\mathbf{B}}$ for each layer in the network and sample the marginal distribution of each pre-activation latent variable $\mathbf{b}_i$ conditioned on the pseudo-data:

$$p(\mathbf{b}_i|\mathbf{a}_i, \tilde{\mathbf{A}}, \tilde{\mathbf{B}}) = \mathcal{N}\Big(\mathbf{a}_i\mathbf{M} + \boldsymbol{\sigma}_{12}^T\boldsymbol{\Sigma}_{11}^{-1}\big(\tilde{\mathbf{B}} - \tilde{\mathbf{A}}\mathbf{M}\big), \big(\sigma_{22} - \boldsymbol{\sigma}_{12}^T\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\sigma}_{12}\big) \odot \mathbf{V}\Big) \tag{6}$$

$$\boldsymbol{\Sigma}_{11} = \tilde{\mathbf{A}}\mathbf{U}\tilde{\mathbf{A}}^T; \quad \boldsymbol{\sigma}_{12} = \tilde{\mathbf{A}}\mathbf{U}\mathbf{a}_i^T; \quad \sigma_{22} = \mathbf{a}_i\mathbf{U}\mathbf{a}_i^T \tag{7}$$

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 nonlinear 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 $p(\tilde{\mathbf{B}}, \mathbf{b}_i|\tilde{\mathbf{A}}, \mathbf{a}_i)$ of each layer.

Finally, since we want a fully Bayesian model, we also place fully factorized "dropout 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 5 with the inclusion of the pseudo-data is:

$$\mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log p(\mathbf{Y}|\mathbf{X})] \leq \mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}\left[\mathbb{E}_{q_{\phi_1}(\mathbf{B},\tilde{\mathbf{A}}_1,\tilde{\mathbf{B}}_1|\mathbf{X})q_{\phi_2}(\mathbf{F},\tilde{\mathbf{A}}_2,\tilde{\mathbf{B}}_2|\mathbf{B})}\left[\log p(\mathbf{Y}|\mathbf{F})\right] + \sum_{i=1}^{2} L_c(\phi_i, \theta_i)\right] \tag{8}$$

$$q_{\phi_i}(\mathbf{B}, \tilde{\mathbf{A}}, \tilde{\mathbf{B}}|\mathbf{X}) = \tilde{q}_{\phi_i}(\mathbf{B}|\mathbf{X}, \tilde{\mathbf{A}}, \tilde{\mathbf{B}})q_{\phi_i}(\tilde{\mathbf{A}})q_{\phi_i}(\tilde{\mathbf{B}}) \tag{9}$$

$$L_c(\phi_i, \theta_i) = -KL(q(\mathbf{W}_i)||p(\mathbf{W}_i)) - KL(q(\tilde{\mathbf{A}}_i)||p(\tilde{\mathbf{A}}_i)) - KL(q(\tilde{\mathbf{B}}_i)||p(\tilde{\mathbf{B}}_i)) \tag{10}$$

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).

4 REGRESSION EXPERIMENT

We tested the aforementioned model with a rank-1 approximation for the square root of the covariance matrices of the matrix variate Gaussian posteriors on the regression task used in Hernández-Lobato & Adams (2015) and Gal & Ghahramani (2015). Similarly to Hernández-Lobato & Adams (2015) we also introduced Gamma priors and posteriors for the precision of the Gaussian likelihood and the precision of the row and column for the matrix-variate prior. We used 5 pseudo input-output pairs for each layer for the small 'Yacht Hydrodynamics' dataset, 20 for the larger 'Protein Structure' and 'Year Prediction MSD' and 10 for the rest. The results can be seen at Tables 1 and 2 where VI, PBP, Dropout and DMGP correspond to the variational inference method of Graves (2011), probabilistic backpropagation (Hernández-Lobato & Adams, 2015), dropout uncertainty (Gal & Ghahramani, 2015) and the model considered here.

Table 1: Average test set RMSE and standard errors for the regression datasets.

Dataset	**VI**	**PBP**	**Dropout**	**DMGP**
Boston Housing	4.32±0.29	3.01±0.18	2.97±0.85	**2.81±0.13**
Concrete Compression Strength	7.19±0.12	5.67±0.09	5.23± 0.53	**4.64±0.14**
Energy Efficiency	2.65±0.08	1.80±0.05	1.66±0.19	**1.11±0.03**
Kin8nm	0.10±0.00	0.10±0.00	0.10±0.00	**0.08±0.00**
Naval Propulsion	0.01±0.00	0.01±0.00	0.01±0.00	**0.00±0.00**
Combined Cycle Power Plant	4.33±0.04	4.12±0.03	4.02±0.18	**3.85±0.03**
Protein Structure	4.84±0.03	4.73±0.01	4.36±0.04	**4.11±0.01**
Wine Quality Red	0.65±0.01	0.64±0.01	0.62±0.04	**0.61±0.01**
Yacht Hydrodynamics	6.89±0.67	1.02±0.05	1.11±0.38	**0.78±0.06**
Year Prediction MSD	9.034±NA	8.879±NA	8.849±NA	**8.790±NA**

Table 2: Average predictive log-likelihood and standard errors for the regression datasets.

Dataset	**VI**	**PBP**	**Dropout**	**DMGP**
Boston Housing	-2.90±0.07	-2.57± 0.09	**-2.46±0.25**	-2.58±0.08
Concrete Compression Strength	-3.39±0.02	-3.16±0.02	-3.04±0.09	**-2.97±0.04**
Energy Efficiency	-2.39±0.03	-2.04±0.02	-1.99±0.09	**-1.34±0.02**
Kin8nm	0.90±0.01	0.90±0.01	0.95±0.03	**1.15±0.01**
Naval Propulsion	3.73±0.12	3.73±0.01	3.80±0.05	**5.85±0.00**
Combined Cycle Power Plant	-2.89±0.01	-2.84±0.01	-2.80±0.05	**-2.77±0.01**
Protein Structure	-2.99±0.01	-2.97±0.00	-2.89±0.01	**-2.83±0.00**
Wine Quality Red	-0.98±0.01	-0.97±0.01	-0.93±0.06	**-0.93±0.02**
Yacht Hydrodynamics	-3.43±0.16	-1.63±0.02	-1.55±0.12	**-1.15±0.04**
Year Prediction MSD	-3.622±NA	-3.603±NA	**-3.588±NA**	-3.591±NA

REFERENCES

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

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

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

Andreas C. Damianou and Neil D. Lawrence. 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.

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

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

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

José Miguel Hernández-Lobato and Ryan Adams. 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.

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

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

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

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 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.

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

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