Eigenvalue Corrected Noisy Natural Gradient

Juhan Bae[1,2], **Guodong Zhang**[1,2], **Roger Grosse**[1,2]
University of Toronto[1] Vector Institute[2]
juhan.bae@mail.utoronto.ca
{gdzhang, rgrosse}@cs.toronto.edu

Abstract

Variational Bayesian neural networks combine the flexibility of deep learning with Bayesian uncertainty estimation. However, inference procedures for flexible variational posteriors are computationally expensive. A recently proposed method, noisy natural gradient, is a surprisingly simple method to fit expressive posteriors by adding weight noise to regular natural gradient updates. Noisy K-FAC is an instance of noisy natural gradient that fits a matrix-variate Gaussian posterior with minor changes to ordinary K-FAC. Nevertheless, a matrix-variate Gaussian posterior does not capture an accurate diagonal variance. In this work, we extend on noisy K-FAC to obtain a more flexible posterior distribution called eigenvalue corrected matrix-variate Gaussian. The proposed method computes the full diagonal re-scaling factor in Kronecker-factored eigenbasis. Empirically, our approach consistently outperforms existing algorithms (e.g., noisy K-FAC) on regression and classification tasks.

1 Introduction

Building flexible and scalable uncertainty models [MacKay, 1992, Neal, 2012, Hinton and Van Camp, 1993] has long been a goal in Bayesian deep learning. Variational Bayesian neural networks [Graves, 2011, Blundell et al., 2015] are especially appealing because they combine the flexibility of deep learning with Bayesian uncertainty estimation. However, such models tend to impose overly restricted assumptions (e.g., fully-factorized) in approximating posterior distributions. There have been attempts to fit more expressive distributions [Louizos and Welling, 2016, Sun et al., 2017], but they are difficult to train due to strong and complicated posterior dependencies.

Noisy natural gradient is a simple and efficient method to fit multivariate Gaussian posteriors [Zhang et al., 2017]. It adds adaptive weight noise to regular natural gradient updates. Noisy K-FAC is a practical algorithm in the family of noisy natural gradient [Zhang et al., 2017], which fits a matrix-variate Gaussian posterior (flexible posterior) with only minimal changes to ordinary K-FAC update [Martens and Grosse, 2015] (cheap inference). The update for noisy K-FAC closely resembles standard K-FAC update with correlated weight noise.

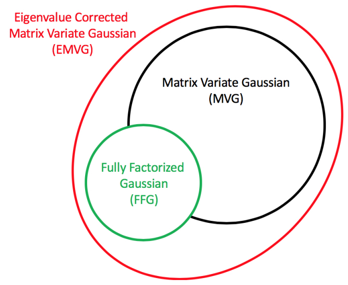

Figure 1: A cartoon illustration to describe the relationships of FFG, MVG, and EMVG.

Nevertheless, we note that a matrix-variate Gaussian cannot capture an accurate diagonal variance. In this work, we build upon the large body of noisy K-FAC and Eigenvalue corrected Kronecker-factored Approximate Curvature (EK-FAC) [George et al., 2018] to improve the flexibility of the posterior distribution. We compute the diagonal variance, not in parameter coordinates, but in K-FAC eigenbasis. This leads to a more expressive posterior distribution. The relationship is described in Figure 1. Using this insight, we introduce a modified training method for variational Bayesian neural networks called *noisy EK-FAC*.

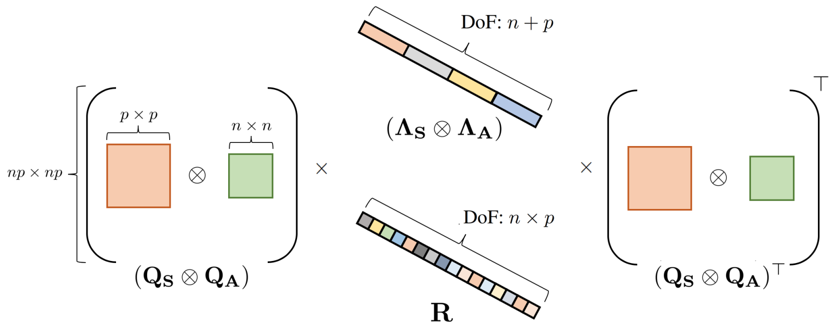

Figure 2: The diagonal re-scaling factor in K-FAC has Kronecker structure with $n+p$ degrees of freedom. The diagonal re-scaling matrix in EK-FAC is the second moment of the gradient vector with $n \times p$ degrees of freedom.

2 Natural Gradient

Natural gradient descent is a second-order optimization technique first proposed by Amari [1997]. It is classically motivated as a way of implementing steepest descent in the space of distributions instead of the space of parameters. The distance function for distribution space is the KL divergence on the model's predictive distribution: $\mathrm{D}_{\mathrm{KL}}(p_{\boldsymbol{\theta}} \,\|\, p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}) \approx \frac{1}{2}\boldsymbol{\theta}^\top \mathbf{F}\boldsymbol{\theta}$, where $\mathbf{F}$ is the Fisher matrix.

$$\mathbf{F} = \mathbb{E}\left[\nabla_{\mathbf{w}} \log p(y|\mathbf{x}, \mathbf{w}) \nabla_{\mathbf{w}} \log p(y|\mathbf{x}, \mathbf{w})^\top\right] \tag{1}$$

This results in the preconditioned gradients $\tilde{\nabla}_{\mathbf{w}} h = \mathbf{F}^{-1}\nabla_{\mathbf{w}} h$. Natural gradient descent is invariant to smooth and invertible reparameterizations of the model [Martens, 2014].

2.1 Kronecker-Factored Approximate Curvature

Modern neural networks contain millions of parameters which makes storing and computing the inverse of the Fisher matrix impractical. Kronecker-Factored Approximate Curvature (K-FAC) [Martens and Grosse, 2015] uses Kronecker products to efficiently approximate the inverse Fisher matrix[1].

For a layer of a neural network whose input activations are $\mathbf{a} \in \mathbb{R}^n$, weight matrix $\mathbf{W} \in \mathbb{R}^{n \times p}$, and pre-activation output $\mathbf{s} \in \mathbb{R}^p$, we can write $\mathbf{s} = \mathbf{W}^\top \mathbf{a}$. The gradient with respect to the weight matrix is $\nabla_{\mathbf{W}} h = \mathbf{a}(\nabla_{\mathbf{s}} h)^\top$. Assuming $\mathbf{a}$ and $\mathbf{s}$ are independent under the model's predictive distribution, K-FAC decouples the Fisher matrix $\mathbf{F}$:

$$\begin{aligned}\mathbf{F} &= \mathbb{E}\left[\mathrm{vec}\{\nabla_{\mathbf{W}} h\}\mathrm{vec}\{\nabla_{\mathbf{W}} h\}^\top\right] = \mathbb{E}\left[\{\nabla_{\mathbf{s}} h\}\{\nabla_{\mathbf{s}} h\}^\top \otimes \mathbf{a}\mathbf{a}^\top\right] \\ &\approx \mathbb{E}\left[\{\nabla_{\mathbf{s}} h\}\{\nabla_{\mathbf{s}} h\}^\top\right] \otimes \mathbb{E}\left[\mathbf{a}\mathbf{a}^\top\right] = \mathbf{S} \otimes \mathbf{A}\end{aligned} \tag{2}$$

where $\mathbf{A} = \mathbb{E}\left[\mathbf{a}\mathbf{a}^\top\right]$ and $\mathbf{S} = \mathbb{E}\left[\{\nabla_{\mathbf{s}} h\}\{\nabla_{\mathbf{s}} h\}^\top\right]$. Bernacchia et al. [2018] showed that the Kronecker approximation is exact for deep linear networks, justifying the validity of the above assumption. Further assuming the between-layer independence, the Fisher matrix is approximated as block diagonal consisting of layer-wise Fisher matrices. Decoupling $\mathbf{F}$ into $\mathbf{A}$ and $\mathbf{S}$ avoids the memory issue of storing the full matrix $\mathbf{F}$ while also having the ability to perform efficient inverse Fisher vector products:

$$\mathbf{F}^{-1}\mathrm{vec}\{\nabla_{\mathbf{W}} h\} = \mathbf{S}^{-1} \otimes \mathbf{A}^{-1}\mathrm{vec}\{\nabla_{\mathbf{W}} h\} = \mathrm{vec}[\mathbf{A}^{-1}\nabla_{\mathbf{W}} h \mathbf{S}^{-1}] \tag{3}$$

As shown in equation (3), natural gradient descent with K-FAC only consists of a series of matrix multiplications comparable to the size of $\mathbf{W}$. This enables an efficient computation of a natural gradient descent.

[1]Extending on this work, K-FAC was shown to be amenable to distributed computation [Ba et al., 2016] and could generalize as well as SGD [Zhang et al., 2018].

2.2 An Alternative Interpretation of Natural Gradient

George et al. [2018] suggest an alternative way of interpreting the natural gradient update. It can be broken down into three stages:

$$\mathbf{F}^{-1}\mathrm{vec}\{\nabla_{\mathbf{W}} h\} = {\color{orange}\underline{\mathbf{Q}\,{\color{cyan}\underline{\color{black}\mathbf{R}^{-1}\,{\color{red}\underline{\color{black}\mathbf{Q}^{\top}\mathrm{vec}\{\nabla_{\mathbf{W}} h\}}}}}}} \tag{4}$$

The first stage (${\color{red}-}$) projects the gradient vector to the full Fisher eigenbasis $\mathbf{Q}$. The next step (${\color{cyan}-}$) re-scales the coordinates in full Fisher eigenbasis $\mathbf{Q}$ with the diagonal re-scaling factor $\mathbf{R}$. The last stage (${\color{orange}-}$) projects back to the parameter coordinates.

For a diagonal approximation of the Fisher matrix, the basis is chosen to be identity matrix $\mathbf{I}$ and the re-scaling factor $\mathbf{R}_{ii} = \mathbb{E}[(\nabla_{\mathbf{w}})_i^2]$ is the second moment of the gradient vector. While estimating the diagonal factor is simple and efficient, obtaining an accurate eigenbasis is difficult. The crude basis in the diagonal Fisher introduces a significant approximation error.

K-FAC decouples the Fisher matrix into $\mathbf{S}$ and $\mathbf{A}$. Since $\mathbf{S}$ and $\mathbf{A}$ are symmetric positive semi-definite matrices, by eigendecomposition, they can be represented as $\mathbf{S} = \mathbf{Q_S}\mathbf{\Lambda_S}\mathbf{Q_S}^{\top}$ and $\mathbf{A} = \mathbf{Q_A}\mathbf{\Lambda_A}\mathbf{Q_A}^{\top}$, where $\mathbf{Q}$ is an orthogonal matrix whose columns are eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues. We use properties of the Kronecker product to further decompose the factorization:

$$\mathbf{S}\otimes\mathbf{A} = \mathbf{Q_S}\mathbf{\Lambda_S}\mathbf{Q_S}^{\top}\otimes\mathbf{Q_A}\mathbf{\Lambda_A}\mathbf{Q_A}^{\top} = (\mathbf{Q_S}\otimes\mathbf{Q_A})(\mathbf{\Lambda_S}\otimes\mathbf{\Lambda_A})(\mathbf{Q_S}\otimes\mathbf{Q_A})^{\top} \tag{5}$$

Based on the new interpretation, we have K-FAC eigenbasis $\mathbf{Q_S}\otimes\mathbf{Q_A}$ and diagonal re-scaling factor $\mathbf{\Lambda_S}\otimes\mathbf{\Lambda_A}$. K-FAC eigenbasis is provably a more accurate approximation of the full Fisher eigenbasis. However, it does not use the estimated variance along the basis. The re-scaling factor in K-FAC is constrained to the Kronecker structure.

2.3 Eigenvalue Corrected Kronecker-Factored Approximate Curvature

Eigenvalue corrected K-FAC (EK-FAC) [George et al., 2018] extends on K-FAC to compute a more accurate diagonal re-scaling factor in K-FAC eigenbasis. The re-scaling factor for K-FAC is expressed in $n+p$ degrees of freedom, where n and p are input and output size of a layer. K-FAC factorization in equation (5) does not capture an accurate diagonal re-scaling factor in K-FAC eigenbasis because of the Kronecker structure. Instead, EK-FAC computes the second moment of the gradient vector in K-FAC eigenbasis. We define the re-scaling matrix $\mathbf{R}\in\mathbb{R}^{np\times np}$ as follows:

$$\mathbf{R}_{ii} = \mathbb{E}[((\mathbf{Q_S}\otimes\mathbf{Q_A})^{\top}\nabla_{\mathbf{w}})_i^2] \tag{6}$$

$\mathbf{R}$ is a diagonal matrix whose entries are the second moment. The Fisher matrix can be approximated with K-FAC eigenbasis and the re-scaling matrix:

$$\mathbf{F}\approx(\mathbf{Q_S}\otimes\mathbf{Q_A})\mathbf{R}(\mathbf{Q_S}\otimes\mathbf{Q_A})^{\top} \tag{7}$$

EK-FAC re-scaling matrix minimizes the approximation error of the above equation in Frobenius norm. In comparison to K-FAC approximation, Eigenvalue corrected K-FAC (EK-FAC) approximation is more flexible in representing the diagonal re-scaling factor with $n\times p$ degrees of freedom. Figure 2 illustrates the difference between K-FAC and EK-FAC.

3 Variational Bayesian Neural Networks

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^{n}\}$, a Bayesian Neural Network (BNN) is composed of a log-likelihood $p(\mathcal{D}|\mathbf{w})$ and a prior $p(\mathbf{w})$ on the weights. Performing inference on BNN requires integrating over the intractable posterior distribution $p(\mathbf{w}\,|\,\mathcal{D})$. Variational Bayesian methods [Hinton and Van Camp, 1993, Graves, 2011, Blundell et al., 2015] attempt to fit an approximate posterior $q(\mathbf{w})$ to maximize the evidence lower bound (ELBO):

$$\mathcal{L}[q] = \mathbb{E}_q[\log p(\mathcal{D}\,|\,\mathbf{w})] - \lambda \mathrm{D}_{\mathrm{KL}}(q(\mathbf{w})\,\|\,p(\mathbf{w})) \tag{8}$$

where λ is a regularization parameter and ϕ are parameters of the variational posterior. The exact Bayesian inference uses $\lambda = 1$, but it can be tuned in practical settings.

Bayes By Backprop (BBB) [Blundell et al., 2015] is the most common variational BNN training method. It uses a fully-factorized Gaussian approximation to the posterior i.e. $q(\mathbf{w}) =$

$\mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$. The variational parameters $\boldsymbol{\phi} = (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ are updated according to stochastic gradients of $\mathcal{L}$ obtained by the reparameterization trick [Kingma and Welling, 2013].

There has been attempts to fit a matrix-variate Gaussian posterior for BNNs [Louizos and Welling, 2016, Sun et al., 2017]. Compared to overly restricted variational families, a matrix-variate Gaussian effectively captures correlations between weights. However, computing the gradients and enforcing the positive semi-definite constraint for $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ make the inference challenging. Existing methods typically impose additional structures such as diagonal covariance [Louizos and Welling, 2016] or products of Householder transformation [Sun et al., 2017] to ensure efficient updates.

3.1 Noisy Natural Gradient

Noisy natural gradient (NNG) is an efficient method to fit multivariate Gaussian posteriors [Zhang et al., 2017] by adding adaptive weight noise to ordinary natural gradient updates[2]. Assuming $q(\mathbf{w})$ is a multivariate Gaussian posterior parameterized by $\boldsymbol{\phi} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $p(\mathbf{w})$ is a spherical Gaussian, the update rules are:

$$\begin{aligned}
\mathbf{F} &\leftarrow \left(1 - \tilde{\beta}\right) \mathbf{F} + \tilde{\beta} \left(\nabla_{\mathbf{w}} \log p(y|\mathbf{x}, \mathbf{w}) \nabla_{\mathbf{w}} \log p(y|\mathbf{x}, \mathbf{w})^\top\right) \\
\boldsymbol{\mu} &\leftarrow \boldsymbol{\mu} + \tilde{\alpha} \left(\mathbf{F} + \frac{\lambda}{N\eta}\mathbf{I}\right)^{-1} \left(\nabla_{\mathbf{w}} \log p(y|\mathbf{x}, \mathbf{w}) - \frac{\lambda}{N\eta}\mathbf{w}\right)
\end{aligned} \tag{9}$$

where λ is the KL weight and η is the prior variance. In each iteration, NNG samples weights from the variational posterior $q_\phi(\mathbf{w})$, which is a multivariate Gaussian with the covariance matrix:

$$\boldsymbol{\Sigma} = \left(\frac{N}{\lambda}\mathbf{F} + \eta^{-1}\mathbf{I}\right)^{-1} \tag{10}$$

However, due to computational intractability, it is necessary to impose a structured restriction to the covariance matrix. This is equivalent to imposing the same structure to the Fisher matrix.

3.2 Fitting Matrix-Variate Gaussian Posteriors with Noisy K-FAC

Noisy K-FAC is a tractable instance of NNG with Kronecker-factored approximation to the Fisher. Because imposing a structured approximation to the covariance is equivalent to imposing the same structure to the Fisher matrix, noisy K-FAC enforces a Kronecker product structure to the covariance matrix. It efficiently fits the matrix-variate Gaussian posterior. The posterior covariance is given by

$$\begin{aligned}
\boldsymbol{\Sigma} &= (\mathbf{Q}_\mathbf{S} \otimes \mathbf{Q}_\mathbf{A}) \left(\frac{N}{\lambda}(\boldsymbol{\Lambda}_\mathbf{S} \otimes \boldsymbol{\Lambda}_\mathbf{A}) + \eta^{-1}\mathbf{I}\right)^{-1} (\mathbf{Q}_\mathbf{S} \otimes \mathbf{Q}_\mathbf{A})^\top \\
&\approx \frac{\lambda}{N} \left(\mathbf{S} + \frac{1}{\pi}\sqrt{\frac{\lambda}{N\eta}}\mathbf{I}\right)^{-1} \otimes \left(\mathbf{A} + \pi\sqrt{\frac{\lambda}{N\eta}}\mathbf{I}\right)^{-1}
\end{aligned} \tag{11}$$

where π is a scalar constant introduced by Martens and Grosse [2015] in the context of damping to keep a compact representation of the Kronecker product. The pseudo-code for noisy K-FAC is given in Appendix D. In comparison to existing methods that fit MVG posteriors [Sun et al., 2017, Louizos and Welling, 2016], noisy K-FAC does not assume additional approximations.

4 Methods

While matrix-variate Gaussian posterior efficiently captures correlations between different weights, the diagonal variance in K-FAC eigenbasis is not optimal. K-FAC diagonal re-scaling factor $\boldsymbol{\Lambda}_\mathbf{S} \otimes \boldsymbol{\Lambda}_\mathbf{A}$ does not match the second moment along the associated eigenvector $\mathbb{E}[((\mathbf{Q}_\mathbf{S} \otimes \mathbf{Q}_\mathbf{A})^\top \nabla_\mathbf{w})_i^2]$.

We develop a new tractable instance of noisy natural gradient. It keeps track of the diagonal variance in K-FAC eigenbasis, resulting in a more flexible posterior distribution. In the context of NNG, imposing a structural restriction to the Fisher matrix $\mathbf{F}$ is equivalent to imposing the same restriction

[2] Khan et al. [2018] also found the relationship between natural gradient and variational inference and derived VAdam by adding weight noise to Adam, which is similar to noisy Adam in Zhang et al. [2017].

Algorithm 1 Noisy EK-FAC. A subscript l denotes the index of a layer, $\mathbf{w}_l = \text{vec}(\mathbf{W}_l)$, and $\boldsymbol{\mu}_l = \text{vec}(\mathbf{M}_l)$. We assume zero momentum for simplicity. $\oslash$ denotes element-wise division and $\mathbf{r}_l^{\gamma} = \text{diag}(\mathbf{R}_l^{\gamma})$. unvec$(\cdot)$ is an inverse of vec$(\cdot)$ operation i.e. $\text{vec}(\mathbf{A}) = \mathbf{a}$ and $\text{unvec}(\mathbf{a}) = \mathbf{A}$. Differences from standard EK-FAC are shown in blue.

```
Require: $\alpha$: stepsize
Require: $\beta$: exponential moving average parameter for covariance factors
Require: $\omega$: exponential moving average parameter for re-scaling factor
Require: $\lambda, \eta, \gamma_{\text{ex}}$ : KL weighting, prior variance, extrinsic damping term
Require: $T_{\text{stats}}, T_{\text{eig}}, T_{\text{scale}}$: stats, eigendecomposition, and re-scaling update intervals.
  $k \leftarrow 0$ and initialize $\{\boldsymbol{\mu}_l\}_{l=1}^{L}, \{\mathbf{S}_l\}_{l=1}^{L}, \{\mathbf{A}_l\}_{l=0}^{L-1}, \{\mathbf{R}_l\}_{l=1}^{L}$
  Calculate the intrinsic damping term $\gamma_{\text{in}} = \frac{\lambda}{N\eta}$, total damping term $\gamma = \gamma_{\text{in}} + \gamma_{\text{ex}}$
  while stopping criterion not met do
    $k \leftarrow k + 1$
    $\mathbf{W}_l \sim \mathcal{EMN}(\mathbf{M}_l,\ \mathbf{A}_l,\ \mathbf{S}_l,\ \frac{\lambda}{N}[\mathbf{R}_l^{\gamma_{\text{in}}}]^{-1})$
    if $k \equiv 0 \pmod{T_{\text{stats}}}$ then
      Update the covariance factors $\{\mathbf{S}_l\}_{l=1}^{L}, \{\mathbf{A}_l\}_{l=0}^{L-1}$ using eq. (12)
    end if
    if $k \equiv 0 \pmod{T_{\text{scale}}}$ then
      Update the re-scaling factor $\{\mathbf{R}_l\}_{l=1}^{L}$ using eq. (12)
    end if
    if $k \equiv 0 \pmod{T_{\text{eig}}}$ then
      Compute the eigenbasis $\mathbf{Q}_{\mathbf{S}_l}$ and $\mathbf{Q}_{\mathbf{A}_l}$ using eq. (5).
    end if
    $\mathbf{V}_l = \nabla_{\mathbf{W}_l} \log p(y|\mathbf{x}, \mathbf{w}) - \gamma_{\text{in}} \cdot \mathbf{W}_l$
    $\mathbf{M}_l \leftarrow \mathbf{M}_l + \alpha \mathbf{Q}_{\mathbf{A}_l}[(\mathbf{Q}_{\mathbf{A}_l}^{\top} \mathbf{V}_l \mathbf{Q}_{\mathbf{S}_l}) \oslash \text{unvec}(\mathbf{r}_l^{\gamma})]\mathbf{Q}_{\mathbf{S}_l}^{\top}$ {Derivation is shown in Appendix C}
  end while
```

to the variational posterior. For example, noisy K-FAC imposes a Kronecker product structure to the covariance matrix as shown in equation (11).

Given these insights, building a flexible variational posterior boils down to finding an improved approximation of the Fisher matrix. We adopt EK-FAC method, which is provably a better approximation of the Fisher matrix than K-FAC. We term the new BNN training method *noisy EK-FAC*.

EK-FAC uses eigenvalue corrected Kronecker-factored approximation to the Fisher matrix as described in equation (7). For each layer, it estimates $\mathbf{A}$, $\mathbf{S}$, and $\mathbf{R}$ online using exponential moving averages. Conveniently, this resembles the exponential moving average updates for the noisy natural gradient in equation (9).

$$\begin{aligned}
\mathbf{A} &\leftarrow (1-\tilde{\beta})\mathbf{A} + \tilde{\beta}\mathbf{a}\mathbf{a}^{\top} \\
\mathbf{S} &\leftarrow (1-\tilde{\beta})\mathbf{S} + \tilde{\beta}\nabla_{\mathbf{s}} \log p(y|\mathbf{x},\mathbf{w}) \nabla_{\mathbf{s}} \log p(y|\mathbf{x},\mathbf{w})^{\top} \\
\mathbf{R}_{ii} &\leftarrow (1-\tilde{\omega})\mathbf{R}_{ii} + \tilde{\omega}\left[(\mathbf{Q}_{\mathbf{S}} \otimes \mathbf{Q}_{\mathbf{A}})^{\top} \nabla_{\mathbf{w}} \log p(y|\mathbf{x},\mathbf{w})\right]_i^2
\end{aligned} \tag{12}$$

where $\tilde{\beta}$ is the learning rate for Kronecker factors and $\tilde{\omega}$ is the learning rate for the diagonal re-scaling factor.

We introduce an eigenvalue corrected matrix-variate Gaussian (EMVG) posterior shown in Figure 1. An EMVG is a generalization of a multivariate Gaussian distribution with the following form:

$$\mathcal{EMN}(\mathbf{W}; \mathbf{M}, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \mathbf{R}) = \mathcal{N}(\text{vec}(\mathbf{W}); \text{vec}(\mathbf{M}), (\mathbf{Q}_{\boldsymbol{\Sigma}_1} \otimes \mathbf{Q}_{\boldsymbol{\Sigma}_2})\mathbf{R}(\mathbf{Q}_{\boldsymbol{\Sigma}_1} \otimes \mathbf{Q}_{\boldsymbol{\Sigma}_2})^{\top}) \tag{13}$$

An EMVG posterior is potentially powerful because it not only compactly represents covariances between weights but also computes a full diagonal variance in K-FAC eigenbasis. Applying EK-FAC approximation into equation (10) yields an EMVG posterior. Therefore, we factorize the covariance matrix in the same sense EK-FAC approximates the Fisher matrix:

$$\begin{aligned}
\boldsymbol{\Sigma} &= \frac{\lambda}{N}(\mathbf{Q}_{\mathbf{S}} \otimes \mathbf{Q}_{\mathbf{A}})(\mathbf{R}^{\gamma})^{-1}(\mathbf{Q}_{\mathbf{S}} \otimes \mathbf{Q}_{\mathbf{A}})^{\top} \\
&= \frac{\lambda}{N}(\mathbf{Q}_{\mathbf{S}} \otimes \mathbf{Q}_{\mathbf{A}})\left(\mathbf{R} + \frac{\lambda}{N\eta}\mathbf{I}\right)^{-1}(\mathbf{Q}_{\mathbf{S}} \otimes \mathbf{Q}_{\mathbf{A}})^{\top}
\end{aligned} \tag{14}$$

where γ is an intrinsic damping term. Since the damping $\frac{\lambda}{N\eta}\mathbf{I}$ does not affect K-FAC eigenbasis, we explicitly represent the damping term in the re-scaling matrix. In practice, it may be advantageous to add extrinsic damping to the re-scaling matrix for the stable training process.

The only difference from standard EK-FAC is that the weights are sampled from the variational posterior q. We can interpret noisy EK-FAC in the sense that $\boldsymbol{\mu}$ is a point estimate of the weights and $\boldsymbol{\Sigma}$ is the covariance of correlated Gaussian noise for each training examples. The full algorithm is described in alg. 1.

The inference is efficient because the covariance matrix is factorized with three small matrices $\mathbf{A}, \mathbf{S}$, and $\mathbf{R}$. We can use the following identity to compute Kronecker products efficiently: $(\mathbf{A} \otimes \mathbf{B})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{B}\mathbf{X}\mathbf{A}^\top)$.

5 Related Work

Variational inference was first applied to neural networks by Peterson [1987] and Hinton and Van Camp [1993]. Then, Graves [2011] proposed a practical method for variational inference with fully factorized Gaussian posteriors which uses a simple (but biased) gradient estimator. Improving on this work, Blundell et al. [2015] proposed an unbiased gradient estimator using the reparameterization trick of Kingma and Welling [2013]. Several non-Gaussian variational posteriors have also been proposed such as Multiplicative Normalizing Flows [Louizos and Welling, 2017] and implicit distributions [Shi et al., 2018]. Neural networks with dropout were also interpreted as BNNs [Gal and Ghahramani, 2016, Gal et al., 2017].

A few recent works explored structured covariance approximations by exploiting the relationship between natural gradient and variational inference. Both Zhang et al. [2017] and Khan et al. [2018] used a diagonal Fisher approximation in natural gradient VI, obtaining a fully factorized Gaussian posterior. Zhang et al. [2017] also proposed an interesting extension by using K-FAC, which leads to a matrix-variate Gaussian posterior. Concurrently, Mishkin et al. [2018] adopted a "diagonal plus low-rank" approximation. This method shares the same spirit as this work. However, their low rank approximation is computationally expensive and thus only applied to two-layer (shallow) neural networks.

6 Experiments

In order to empirically evaluate the proposed method, we test under two scenarios, regression and classification, to investigate the following questions. (1) Does noisy EK-FAC have improved prediction performance compared to existing methods? (2) Is it able to scale to large dataset and modern convolution neural network architecture?

6.1 Regression

We evaluate our method on standard BNN benchmarks from UCI collections [Dheeru and Karra Taniskidou, 2017], adopting evaluation protocols from Hernández-Lobato and Adams [2015]. In particular, we introduce a Gamma prior $p(\gamma) = \text{Gam}(a_0 = 6, b_0 = 6)$ for the precision of Gaussian likelihood and include a Gamma posterior $p(\gamma) = \text{Gam}(\alpha^\gamma, \beta^\gamma)$ into the variational objective.

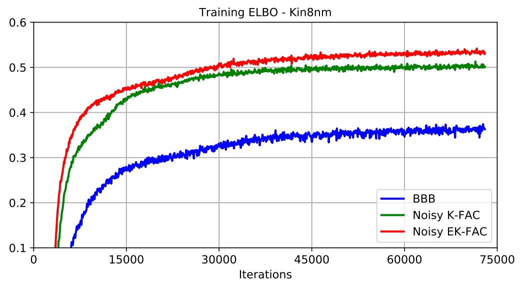

Figure 3: Training curve for all three methods. Note that BBB, noisy K-FAC, and noisy EK-FAC use FFG, MVG, EMVG accordingly. EMVG has the most flexible variational posterior distribution.

We randomly split training (90%) and test (10%) data. To reduce the randomness, we repeat the splitting 10 times, except for two largest datasets. "Year" and "Protein" are repeated 1 and 5 times. During training, we normalize input features and training targets to zero mean and unit variance. We do not adopt this normalization at test time. All experiments train a network with a single hidden layer with 50 units except for "Protein" and "Year" datasets, which have 100 hidden units. We use batch size of 10 for smaller datasets, 100 for larger datasets, and 500 for "Year" dataset. To stabilize

Table 1: Average RMSE and log-likelihood in test data for UCI regression benchmarks.

	Test RMSE				Test log-likelihood			
Dataset	BBB	Noisy Adam	Noisy K-FAC	Noisy EK-FAC	BBB	Noisy Adam	Noisy K-FAC	Noisy EK-FAC
Boston	3.171±0.149	3.031±0.155	2.742±0.015	**2.527±0.158**	-2.602±0.031	-2.558±0.032	-2.409±0.047	**-2.378±0.044**
Concrete	5.678±0.087	5.613±0.113	5.019±0.127	**4.880±0.120**	-3.149±0.018	-3.145±0.023	-3.039±0.025	**-3.002±0.025**
Energy	0.565±0.018	0.839±0.046	**0.485±0.019**	0.497±0.023	-1.500±0.006	-1.629±0.020	**-1.421±0.004**	-1.448±0.004
Kin8nm	0.080±0.001	0.079±0.001	**0.076±0.001**	**0.076±0.000**	1.111±0.007	1.112±0.008	1.148±0.007	**1.149±0.012**
Naval	**0.000±0.000**	0.001±0.000	**0.000±0.000**	**0.000±0.000**	6.143±0.032	6.231±0.041	7.079±0.034	**7.287±0.002**
Pow. Plant	4.023±0.036	4.002±0.039	**3.886±0.041**	3.895±0.053	-2.807±0.010	-2.803±0.010	-2.776±0.012	**-2.774±0.012**
Protein	4.321±0.017	4.380±0.016	4.097±0.009	**4.042±0.027**	-2.882±0.004	-2.896±0.004	-2.836±0.002	**-2.819±0.007**
Wine	0.643±0.012	0.644±0.011	0.637±0.011	**0.635±0.013**	-0.977±0.017	-0.976±0.016	-0.969±0.014	**-0.964±0.002**
Yacht	1.174±0.086	1.289±0.069	0.979±0.077	**0.974±0.116**	-2.408±0.007	-2.412±0.006	-2.316±0.006	**-2.224±0.007**
Year	9.076±NA	9.071±NA	8.885±NA	**8.642±NA**	-3.614±NA	-3.620±NA	-3.595±NA	**-3.573±NA**

the training, we re-initialize the re-scaling matrix every 50 iterations with K-FAC eigenvalues. This is equivalent to executing a K-FAC iteration. We found that the second moment of the gradient vector is unstable for a small batch. We amortize the basis update to ensure re-scaling matrix matches the eigenbasis, setting $T_{\text{eig}} = 5$. For learning rates, we use $\alpha = 0.01$, $\beta = 0.001$, and $\omega = 0.01$ for all datasets. They are decayed by 0.1 for the second half of the training.

We compare the results with Bayes by Backprop [Blundell et al., 2015], noisy Adam, and noisy K-FAC [Zhang et al., 2017]. We report root mean square error (RMSE) and log-likelihood on the test dataset. The results are summarized in Table 1. The evaluation result shows that noisy EK-FAC yields a higher test log-likelihood compared to other methods. The training plot is shown in Figure 3. Noisy EK-FAC also achieves a higher ELBO compared to noisy K-FAC.

6.2 Classification

To evaluate the scalability of the proposed method, we train a modified VGG16 [Simonyan and Zisserman, 2014] and test on CIFAR10 benchmarks [Krizhevsky, 2009]. The modified VGG16 has a half reduced number of hidden units in all layers. Similar to applying K-FAC on convolutional layers with Kronecker factors [Grosse and Martens, 2016], EK-FAC can be extended to convolutional layers. We compare the results with SGD (with momentum), K-FAC, and noisy K-FAC.

Table 2: Classification accuracy on CIFAR10 with modified VGG16. **[D]** denotes data augmentation and **[B]** denotes Batch Normalization.

Method	Test Accuracy			
		D	B	D + B
SGD	81.79	88.35	85.75	91.39
KFAC	82.39	88.89	86.86	92.13
Noisy-KFAC	85.52	89.35	88.22	92.01
Noisy-EKFAC	**87.07**	**89.86**	**88.45**	**92.22**

We use batch size of 128 for all experiments. To reduce the computational overhead, we amortize covariance, inverse, and re-scaling updates. Specifically, we set $T_{\text{stats}} = 10$, $T_{\text{scale}} = 10$, and $T_{\text{eig}} = 200$. We noticed that the amortization does not significantly impact per-iteration optimization performance. β and ω are set to 0.01 for both noisy K-FAC and noisy EK-FAC. We adopt batch normalization [Ioffe and Szegedy, 2015] and data augmentation. We tune regularization parameter λ and prior variance η. With data augmentation, we use a smaller regularization parameter. η is set to 0.1 without batch normalization and 1.0 with batch normalization.

The results are summarized in table 2. Noisy EK-FAC achieves the highest test accuracy in all settings without introducing computational overhead. Without extra regularization tricks, noisy EK-FAC has 1.55% improvement compared to noisy K-FAC.

7 Conclusion

In this paper, we introduced a modified training method for variational Bayesian neural networks. An eigenvalue corrected matrix-variate Gaussian extends on a matrix-variate Gaussian to represent the posterior distribution with more flexibility. It not only efficiently captures correlations between weights but also computes an accurate diagonal variance under K-FAC eigenbasis. For both regression and classification evaluations, noisy EK-FAC achieves higher ELBO and test accuracy, demonstrating its effectiveness.

References

Shun-ichi Amari. Neural learning in structured parameter spaces-natural riemannian gradient. In *Advances in neural information processing systems*, pages 127–133, 1997.

Jimmy Ba, Roger Grosse, and James Martens. Distributed second-order optimization using kronecker-factored approximations. 2016.

Alberto Bernacchia, Máté Lengyel, and Guillaume Hennequin. Exact natural gradient in deep linear networks and application to the nonlinear case. 2018.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems*, pages 3581–3590, 2017.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker-factored eigenbasis. *arXiv preprint arXiv:1806.03884*, 2018.

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582, 2016.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM, 1993.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. *arXiv preprint arXiv:1806.04854*, 2018.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

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

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational Bayesian neural networks. In *International Conference on Machine Learning*, pages 2218–2227, 2017.

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

James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark Schmidt, and Mohammad Emtiyaz Khan. Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems*, pages 6246–6256, 2018.

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

Carsten Peterson. A mean field theory learning algorithm for neural networks. *Complex systems*, 1: 995–1019, 1987.

Jiaxin Shi, Shengyang Sun, and Jun Zhu. Kernel implicit variational inference. In *International Conference on Learning Representations*, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning structured weight uncertainty in bayesian neural networks. In *Artificial Intelligence and Statistics*, pages 1283–1292, 2017.

Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390*, 2017.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*, 2018.

A Matrix-Variate Gaussian

A matrix-variate Gaussian distribution models a multivariate Gaussian distribution for a matrix $\mathbf{W} \in \mathbb{R}^{n\times p}$.

$$p(\mathbf{W}|\mathbf{M},\mathbf{U},\mathbf{V}) = \frac{\exp(-\frac{1}{2}\mathrm{tr}[\mathbf{V}^{-1}(\mathbf{W}-\mathbf{M})^\top\mathbf{U}^{-1}(\mathbf{W}-\mathbf{M})])}{(2\pi)^{np/2}|\mathbf{V}|^{n/2}|\mathbf{U}|^{p/2}} \tag{15}$$

where $\mathbf{M} \in \mathbb{R}^{n\times p}$ is the mean, $\mathbf{U} \in \mathbb{R}^{n\times n}$ is the covariance among rows and $\mathbf{V} \in \mathbb{R}^{p\times p}$ is the covariance among columns. Since $\mathbf{U}$ and $\mathbf{V}$ are covariance matrices, they are positive definite. Vectorization of $\mathbf{W}$ forms a multivariate Gaussian distribution whose covariance matrix $\mathbf{\Sigma}$ is a Kronecker product of $\mathbf{V}$ and $\mathbf{U}$.

$$\mathrm{vec}(\mathbf{W}) \sim \mathcal{N}(\mathrm{vec}(\mathbf{M}), \mathbf{V}\otimes\mathbf{U}) \tag{16}$$

B Eigenvalue Corrected Matrix-Variate Gaussian

An eigenvalue corrected matrix-variate Gaussian is an extension of a matrix-variate Gaussian to consider the full diagonal variance in Kronecker-factored eigenbasis.

$$p(\mathbf{W}|\mathbf{M},\mathbf{U},\mathbf{V},\mathbf{R}) = \frac{\exp(-\frac{1}{2}\mathbf{P}^\top\mathbf{R}^{-1}\mathbf{P})}{(2\pi)^{np/2}|\mathbf{R}|^{np/2}} \text{ and } \mathbf{P} = \mathrm{vec}(\mathbf{Q}_\mathbf{U}^\top(\mathbf{W}-\mathbf{M})\mathbf{Q}_\mathbf{V}) \tag{17}$$

$\mathbf{R} \in \mathbb{R}^{np\times np}$ is the re-scaling matrix. Because covariance matrices are positive definite, diagonal entries in $\mathbf{R}$ are all positive. Similar to a matrix-variate Gaussian distribution, vectorization of $\mathbf{W}$ generalizes a multivariate distribution whose covariance matrix has a Kronecker structure.

$$\mathrm{vec}(\mathbf{W}) \sim \mathcal{N}(\mathrm{vec}(\mathbf{M}), (\mathbf{Q}_\mathbf{V}\otimes\mathbf{Q}_\mathbf{U})\,\mathbf{R}\,(\mathbf{Q}_\mathbf{V}\otimes\mathbf{Q}_\mathbf{U})^\top) \tag{18}$$

Sampling from an eigenvalue corrected matrix-variate distribution is also a special case of sampling from a multivariate Gaussian distribution. Let $\mathbf{X} \in \mathbb{R}^{n\times p}$ be a matrix with independent samples for a standard multivariate Gaussian.

$$\mathbf{X} \sim \mathcal{N}(\mathbf{0},\mathbf{I}) \tag{19}$$

Then let

$$\mathbf{Y} = \mathbf{M} + \mathbf{Q}_\mathbf{U}[\mathbf{X}\odot\mathrm{unvec}(\sqrt{\mathbf{r}})]\mathbf{Q}_\mathbf{V}^\top \tag{20}$$

where $\sqrt{\mathbf{r}} = \sqrt{\mathrm{diag}(\mathbf{R})}$, $\odot$ is an element-wise multiplication, and $\mathrm{unvec}(\cdot)$ is an inverse of $\mathrm{vec}(\cdot)$ operation.

C Derivation of EK-FAC Update

Let $\mathbf{V} \in \mathbb{R}^{n\times p}$ be the weight gradient, $\mathbf{A} \in \mathbb{R}^{n\times n}$ the covariance matrix of input activations, and $\mathbf{S} \in \mathbb{R}^{p\times p}$ the covariance matrix of output pre-activations. $\mathbf{R} \in \mathbb{R}^{np\times np}$ is the diagonal re-scaling matrix in K-FAC eigenbasis: $\mathbf{Q}_\mathbf{S}\otimes\mathbf{Q}_\mathbf{A}$. The following is the derivation of EK-FAC update shown in alg. 1.

$$\begin{aligned}(\mathbf{Q}_\mathbf{S}\otimes\mathbf{Q}_\mathbf{A})\,\mathbf{R}^{-1}\,(\mathbf{Q}_\mathbf{S}\otimes\mathbf{Q}_\mathbf{A})^\top\,\mathrm{vec}(\mathbf{V}) &= (\mathbf{Q}_\mathbf{S}\otimes\mathbf{Q}_\mathbf{A})\,\mathbf{R}^{-1}\left(\mathbf{Q}_\mathbf{S}^\top\otimes\mathbf{Q}_\mathbf{A}^\top\right)\mathrm{vec}(\mathbf{V})\\ &= (\mathbf{Q}_\mathbf{S}\otimes\mathbf{Q}_\mathbf{A})\,\mathbf{R}^{-1}\mathrm{vec}\left(\mathbf{Q}_\mathbf{A}^\top\mathbf{V}\mathbf{Q}_\mathbf{S}\right)\\ &= (\mathbf{Q}_\mathbf{S}\otimes\mathbf{Q}_\mathbf{A})\,\mathrm{vec}((\mathbf{Q}_{\mathbf{A}_l}^\top\mathbf{V}_l\mathbf{Q}_{\mathbf{S}_l})\oslash\mathrm{unvec}(\mathbf{r}_l))\\ &= \mathrm{vec}(\mathbf{Q}_\mathbf{A}[(\mathbf{Q}_\mathbf{A}^\top\mathbf{V}\mathbf{Q}_\mathbf{S})\oslash\mathrm{unvec}(\mathbf{r})]\mathbf{Q}_\mathbf{S}^\top)\end{aligned} \tag{21}$$

where $\mathbf{r} = \mathrm{diag}(\mathbf{R})$, $\oslash$ is an element-wise division, and $\mathrm{unvec}(\cdot)$ is an inverse of $\mathrm{vec}(\cdot)$ operation.

D Pseudo-Code for Noisy K-FAC

Algorithm 2 Noisy K-FAC. A subscript l denotes the index of a layer, $\mathbf{w}_l = \mathrm{vec}(\mathbf{W}_l)$, and $\boldsymbol{\mu}_l = \mathrm{vec}(\mathbf{M}_l)$. We assume zero momentum for simplicity. Differences from standard K-FAC are shown in blue.

```
Require: $\alpha$: stepsize
Require: $\beta$: exponential moving average parameter for covariance factors
Require: $\lambda, \eta, \gamma_{\text{ex}}$ : KL weighting, prior variance, extrinsic damping term
Require: $T_{\text{stats}}, T_{\text{inv}}$: stats and inverse update intervals.
  $k \leftarrow 0$ and initialize $\{\boldsymbol{\mu}_l\}_{l=1}^{L}, \{\mathbf{S}_l\}_{l=1}^{L}, \{\mathbf{A}_l\}_{l=0}^{L-1}, \{\mathbf{R}_l\}_{l=1}^{L}$
  Calculate the intrinsic damping term $\gamma_{\text{in}} = \frac{\lambda}{N\eta}$, total damping term $\gamma = \gamma_{\text{in}} + \gamma_{\text{ex}}$
  while stopping criterion not met do
    $k \leftarrow k + 1$
    $\mathbf{W}_l \sim \mathcal{MN}(\mathbf{M}_l, \ \frac{\lambda}{N}[\mathbf{A}_l^{\gamma_{\text{in}}}]^{-1}, \ [\mathbf{S}_l^{\gamma_{\text{in}}}]^{-1})$
    if $k \equiv 0 \pmod{T_{\text{stats}}}$ then
      Update the covariance factors $\{\mathbf{S}_l\}_{l=1}^{L}, \{\mathbf{A}_l\}_{l=0}^{L-1}$ using eq. (12)
    end if
    if $k \equiv 0 \pmod{T_{\text{inv}}}$ then
      Compute the inverses $\{[\mathbf{S}_l^{\gamma}]^{-1}\}_{l=1}^{L}, \{[\mathbf{A}_l^{\gamma}]^{-1}\}_{l=0}^{L-1}$ using eq. (11).
    end if
    $\mathbf{V}_l = \nabla_{\mathbf{W}_l} \log p(y|\mathbf{x}, \mathbf{w}) - \gamma_{\text{in}} \cdot \mathbf{W}_l$
    $\mathbf{M}_l \leftarrow \mathbf{M}_l + \alpha[\mathbf{A}_l^{\gamma}]^{-1}\mathbf{V}_l[\mathbf{S}_l^{\gamma}]^{-1}$
  end while
```