EigenNet: Towards Fast and Structural Learning of Deep Neural Networks

Ping Luo
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong
pluo@ie.cuhk.edu.hk

Abstract

Deep Neural Network (DNN) is difficult to train and easy to overfit in training. We address these two issues by introducing EigenNet, an architecture that not only accelerates training but also adjusts number of hidden neurons to reduce over-fitting. They are achieved by whitening the information flows of DNNs and removing those eigenvectors that may capture noises. The former improves conditioning of the Fisher information matrix, whilst the latter increases generalization capability. These appealing properties of EigenNet can benefit many recent DNN structures, such as network in network and inception, by wrapping their hidden layers into the layers of EigenNet. The modeling capacities of the original networks are preserved. Both the training wall-clock time and number of updates are reduced by using EigenNet, compared to stochastic gradient descent on various datasets, including MNIST, CIFAR-10, and CIFAR-100.

1 Introduction

Deep neural networks (DNNs) have improved performances of many computer vision tasks, as the non-linearity of DNNs provides expressive learning power, enabling them to learn complicated relationships between images and labels. However, these complicated relationships may have two issues. First, they typically trigger a highly inhomogeneous curvature matrix [LeCun *et al.*, 1991], which impedes the efficiency of stochastic gradient descent (SGD). The learning rate of each hidden layer needs careful tuning, as different layers usually require different learning rates. Second, in large and deep network, some of the learned relationships are noises, which leads to over-fitting.

To reduce over-fitting, recent DNN architectures typically incorporated dropout [Srivastava *et al.*, 2014], which randomly zeroes activations of the hidden neurons to reduce correlation between each pair of the neurons, preventing their learned features from co-adaptation. To accelerate training, previous methods exploited second-order information such as adaptive learning rate [LeCun *et al.*, 1993], centering gradients [Schraudolph, 2002a], and curvature matrix-vector products [Schraudolph, 2002b]. These approaches approximated the curvature matrixes either by their diagonals or by their products with the gradients, to avoid expensive computation of the matrix inverse.

Except for directly computing curvature matrixes, linear multilayer perceptron [Raiko *et al.*, 2012] and projected natural gradient descent (PRONG) [Desjardins *et al.*, 2015] implicitly whitened the input features of each hidden layer to improve conditioning of the Fisher information matrix (FIM). For example, in each update, PRONG projected network parameters into the eigenspace of the input features to mimic the whitening transformation. This scheme performed well in supervised learning such as image recognition, because the inhomogeneity of the FIM is mainly triggered by the input images rather than the supervised labels.

We present EigenNet, which conditions FIM by explicitly whitening both the input features and the gradients, rather than only the input features. It has two appealing properties. First, EigenNet not only accelerates supervised learning, but also improves generative unsupervised learning, where inhomogeneity of FIM is produced by both input and target images. Second, it improves generalization by reducing over-fitting. This is achieved by removing eigenvectors that has small eigenvalues, which may capture noises.

In this paper, Sec.2 introduces the architecture and properties of EigenNet. This is the main contribution. Sec.3 presents a carefully devised training algorithm, which is another important component of EigenNet. Sec.3.1 connects EigenNet with previous works. Experiments in Sec.4 on MNIST, CIFAR-10, and CIFAR-100 datasets demonstrated that both training wall-clock time and number of updates can be reduced by using EigenNet, mimicking the fast convergence of natural gradient descent.

2 EigenNet

Network Overview. This work takes multilayer perceptron (MLP) as an example. The following descriptions can be also adapted to the other networks such as convolutional network and auto-encoder. Fig.1 (a) shows the forward (FP) and backward passes (BP) of a fully-connected (FC) layer in a MLP. FP transforms an input vector, $x \in \mathbb{R}^{m\times 1}$, to an output vector, $a \in \mathbb{R}^{n\times 1}$, by applying a parameter matrix W and a bias vector b. We have $a = Wx + b$. BP propagates the error vector from the upper layer, δa, to the previous layer by $\delta x = W^{\mathsf{T}}\delta a$. Fig.1 (b) shows that each FC layer is wrapped

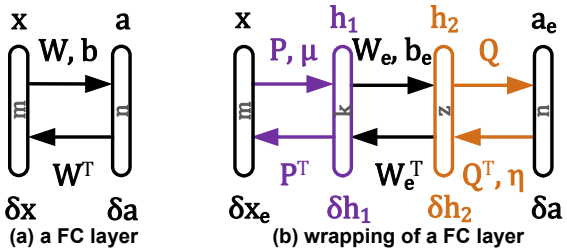

Figure 1: The architecture of EigenNet.

into an EigenNet by appending two intermediate layers, h_1 and h_2. Its information flows are defined as

$$h_1 = P(x-\mu), \quad h_2 = W_e h_1 + b_e, \quad a_e = Q h_2, \quad (1)$$

$$\delta h_2 = Q^\mathsf{T}(\delta a - \eta), \quad \delta h_1 = W_e^\mathsf{T} \delta h_2, \quad \delta x_e = P^\mathsf{T} \delta h_1. \quad (2)$$

Eqn.(1) transforms x to a_e by successively multiplying x with three matrixes, P, W_e, and Q. Eqn.(2) propagates the error signal δa to δx_e by using the transpose of these matrixes. μ and η represent the estimated sample means of x and δa. In particular, P and Q are two projection matrixes that whiten the centered representations of x and δa. They are determined by eigen-decomposition. W_e and b_e denote a parameter matrix and a bias vector.

Construct P and Q. We first construct P and Q in Eqn.(1) and (2). To improve training efficiency, we encourage each dimension of the hidden features h_1 and the errors δh_2 to be independent with each other. Therefore, we have $\mathbb{E}[h_1 h_1^\mathsf{T}] = \mathbb{E}[P(x-\mu)(x-\mu)^\mathsf{T} P^\mathsf{T}] = I$ as well as $\mathbb{E}[\delta h_2 \delta h_2^\mathsf{T}] = \mathbb{E}[Q^\mathsf{T}(\delta a - \eta)(\delta a - \eta)^\mathsf{T} Q] = I$, where $\mathbb{E}[\cdot]$ and I denote expectation and an identity matrix. These equalities are achieved by eigen-decomposition of the correlation matrixes of x and δa. As a result, P and Q are defined as

$$P = S^{-\frac{1}{2}} U^\mathsf{T} \mathrm{diag}(\Sigma)^{-\frac{1}{2}}, \text{ and } Q^\mathsf{T} = E^{-\frac{1}{2}} V^\mathsf{T} \mathrm{diag}(\Upsilon)^{-\frac{1}{2}}, \quad (3)$$

where $\mathrm{diag}(\cdot)$ indicates a diagonal matrix, while $\Sigma \in \mathbb{R}^{m \times m}$ and $\Upsilon \in \mathbb{R}^{n \times n}$ are the covariance matrixes of x and δa, respectively. Moreover, S and E are two diagonal square matrixes whose diagonal elements are the eigenvalues, whilst U and V are two orthogonal matrixes of eigenvectors. For instance, we have $\Sigma = \mathbb{E}[(x-\mu)(x-\mu)^\mathsf{T}]$ and its corresponding correlation matrix is $\mathrm{diag}(\Sigma)^{-\frac{1}{2}} \Sigma \mathrm{diag}(\Sigma)^{-\frac{1}{2}}$. Then, S and U in Eqn.(3) are obtained by eigen-decomposition of the correlation matrix. We have $\mathbb{E}[h_1 h_1^\mathsf{T}] = P\mathbb{E}[(x-\mu)(x-\mu)^\mathsf{T}]P^\mathsf{T} = S^{-\frac{1}{2}} U^\mathsf{T} \mathrm{diag}(\Sigma)^{-\frac{1}{2}} \mathbb{E}[(x-\mu)(x-\mu)^\mathsf{T}] \mathrm{diag}(\Sigma)^{-\frac{1}{2}} U S^{-\frac{1}{2}}$, which becomes an identity matrix I, because $\mathrm{diag}(\Sigma)^{-\frac{1}{2}} \mathbb{E}[(x-\mu)(x-\mu)^\mathsf{T}] \mathrm{diag}(\Sigma)^{-\frac{1}{2}} = \mathrm{diag}(\Sigma)^{-\frac{1}{2}} \Sigma \mathrm{diag}(\Sigma)^{-\frac{1}{2}} = USU^\mathsf{T}$ by construction and $U^\mathsf{T} U = I$.

Pruning Neurons. The number of hidden neurons can be pruned in EigenNet to reduce over-fitting. This is accomplished by pruning feature dimensions that have eigenvalues smaller than a threshold ρ. These dimensions typically capture noises. Removing them reduces over-fitting. To this end, S and E are truncated by removing rows and columns associated with the smallest eigenvalues, whilst the corresponding eigenvectors in U and V are also removed respectively. As a result, we have $\overline{P} = \overline{S}^{-\frac{1}{2}} \overline{U}^\mathsf{T} \mathrm{diag}(\Sigma)^{-\frac{1}{2}} \in \mathbb{R}^{k \times m}$ as well as $\overline{Q}^\mathsf{T} = \overline{E}^{-\frac{1}{2}} \overline{V}^\mathsf{T} \mathrm{diag}(\Upsilon)^{-\frac{1}{2}} \in \mathbb{R}^{z \times n}$, where the overline of a matrix indicates its truncation and we have $k < m$ and $z < n$. In this case, the layer's structure is determined in training by projecting x and δa into lower dimensional spaces.

2.1 Properties of EigenNet

EigenNet has several appealing properties. First, it resembles and generalizes batch normalization (BN) [Ioffe and Szegedy, 2015] to accelerate training. This property can be understood by substituting Eqn.(3) into Eqn.(1) and (2). We have $h_1 = S^{-\frac{1}{2}} U^\mathsf{T} \mathrm{diag}(\Sigma)^{-\frac{1}{2}} (x-\mu)$ and $\delta h_2 = E^{-\frac{1}{2}} V^\mathsf{T} \mathrm{diag}(\Upsilon)^{-\frac{1}{2}} (\delta a - \eta)$, which imply that each dimension of input and error is standardized independently before multiplying by the whitening matrixes, making it have zero mean and unit variance.

Second, EigenNet prevents over-fitting. A modern component to reduce over-fitting is dropout, which gates hidden neurons with a Bernoulli random variable r, so that $\mathbb{E}[(r \circ x)(r \circ x)^\mathsf{T}] = p(1-p)\mathbb{E}[xx^\mathsf{T}]$, where $\circ$ and p denote the element-wise product and dropout ratio. For instance, when $p = 0.5$, covariance after dropout becomes $\frac{1}{4}\mathbb{E}[xx^\mathsf{T}]$. EigenNet factorizes the correlation matrix of hidden neurons, and reduces over-fitting by characterizing only the most significant variations using the eigenvectors that have the largest eigenvalues.

Third, EigenNet improves training efficiency by conditioning the Fisher information matrix (FIM) [Amari and Nagaoka, 2000]. FIM consists of $L \times L$ blocks. Here, L is the number of hidden layers. The (i,j)-th block of a FIM is represented by $F_{ij} = \mathbb{E}[\mathrm{vec}(\Delta W_e^i)\mathrm{vec}(\Delta W_e^j)^\mathsf{T}]$, where ΔW_e^i indicates the gradient of the i-th hidden layer and $\mathrm{vec}(\cdot)$ stacks the columns of a matrix into a single column vector. For example, as illustrated in Fig.1 (b), we have $\mathrm{vec}(\Delta W_e^i) = \mathrm{vec}(h_1^i \delta h_2^{i\mathsf{T}}) = \delta h_2^i \otimes h_1^i$, where $\otimes$ denotes the Kronecker product. In this case, F_{ij} can be rewritten as $\mathbb{E}[(\delta h_2^i \otimes h_1^i)(\delta h_2^j \otimes h_1^j)^\mathsf{T}] = \mathbb{E}[\delta h_2^i \delta h_2^{j\mathsf{T}} \otimes h_1^i h_1^{j\mathsf{T}}]$, which can be approximated by $\mathbb{E}[\delta h_2^i \delta h_2^{j\mathsf{T}}] \otimes \mathbb{E}[h_1^i h_1^{j\mathsf{T}}]$, by assuming δh_2 and h_1 are independent as demonstrated in [Desjardins *et al.*, 2015; Martens and Grosse, 2015; Raiko *et al.*, 2012]. As a result when $i = j$, each diagonal block F_{ii} becomes an identity matrix because we have $\mathbb{E}[h_1^i h_1^{i\mathsf{T}}] = \mathbb{E}[\delta h_2^i \delta h_2^{i\mathsf{T}}] = I$ as discussed in Sec.2, which improves conditioning of FIM of each hidden layer and thus speeds up training.

3 Training Algorithm

Algorithm 1 summarizes the training procedure. The loss function of an EigenNet can be defined as $\mathcal{L}\big(\{W_e^\ell, b_e^\ell\}_{\ell=1}^L;\ \text{s.t. } \{a_e^\ell = a^\ell, \delta x_e^\ell = \delta x^\ell\}_{\ell=1}^L\big)$, which is minimized by optimizing the network parameters as well as satisfying two constraints for each layer ℓ. These constraints maintain the outputs and back-propagated errors of each layer being identical before and after each time of

Algorithm 1 Training EigenNet

Init: Let ℓ and t indicate the ℓ-th layer and t-th update, $t = 1...T$ and $\ell = 1...L$. Let $P^\ell = Q^\ell = I$, and $\mu^\ell = \eta^\ell = 0$, where I and 0 indicate an identity matrix and a vector of zeros, for all ℓ.
for $t = 1 : T$ **and** $\ell = 1...L$
1 perform forward and backward passes by Eqn.(1) and (2).
2 update network parameters W_{et}^ℓ, b_{et}^ℓ by SGD as Eqn.(5).
3 **if** $\mathrm{mod}(t, \gamma) == 0$ ▷*eigen-construction*
4 store old variables obtained at the last eigen-construction, $\mu_o^\ell = \mu^\ell, \eta_o^\ell = \eta^\ell, P_o^\ell = P^\ell$, and $Q_o^\ell = Q^\ell$.
5 estimate new variables $\mu^\ell, \eta^\ell, \Sigma^\ell, \Upsilon^\ell$ by τ mini-batches of samples.
6 construct and truncate new P^ℓ and Q^ℓ by Eqn.(3).
7 transform network parameters W_{et}^ℓ, b_{et}^ℓ by Eqn.(4).
8 **end**
end

eigen-construction (3rd line), enabling a smooth and stable optimization process.

The network parameters, W_{et}^ℓ and b_{et}^ℓ, are updated in every iteration t by using SGD as shown in the 2nd line. The 3rd line indicates that the projection matrixes, P^ℓ and Q^ℓ, and the sample means, μ^ℓ and η^ℓ, can be updated at an interval of γ and kept unchanged in the consecutive γ-1 iterations. They are estimated using τ mini-batches (5th line).

Transform $\mathbf{W_t}$ and $\mathbf{b_t}$. Here, we simplify the notations by discarding ℓ and e. As outlined in Algorithm 1, when $t < \gamma$ (the iterations before the first time of eigen-construction), the information are propagated similarly as in a conventional MLP, because the variables P, Q, μ, and η are initialized as identity matrixes and vectors of zeros, respectively. Eigen-construction is performed when the modulus $\mathrm{mod}(t, \gamma)$ equals zero.

At the first time of eigen-construction when $t = \gamma$, P and Q are constructed and truncated following Eqn.(3). The inverse of them are employed to transform the parameter matrix and the bias vector, so as to satisfy the above two constraints. For instance, we have $W_{\gamma+1} = Q^{-1}W_\gamma P^{-1}$ when $t = \gamma + 1$. In other words, at the beginning of the (γ+1)-th iteration, the transformed parameter matrix is updated at the next consecutive γ-1 iterations with respect to P and Q. More precisely, at the beginning of each eigen-construction, the loss function has been minimized for γ iterations, regarding P_o and Q_o, which have been attained in the preceding eigen-construction. Therefore, to ensure the learned models are identical before and after eigen-construction, we transform the network parameters by

$$W_{t+1} = Q^{-1}Q_o W_t P_o P^{-1} \text{ and} \qquad (4)$$
$$b_{t+1} = Q^{-1}Q_o b_t + Q^{-1}Q_o W_t P_o(\mu - \mu_o).$$

When $t = \gamma$, Eqn.(4) turns into $W_{t+1} = Q^{-1}W_t P^{-1}$ and $b_{t+1} = Q^{-1}b_t + Q^{-1}W_t\mu$ because $P_o = Q_o = I$ and $\mu_o = 0$.

The above two constraints can be achieved by Eqn.(4). Assume that t represents the iteration right before the stage of eigen-construction. By using Eqn.(1), we have $a_t = Q_o\big(W_t P_o(x - \mu_o) + b_t\big)$. At the ($t$+1)-th iteration after eigen-construction, given the same input x, $a_{t+1} = Q\big(W_{t+1}P(x - \mu) + b_{t+1}\big)$. By substituting Eqn.(4) into a_{t+1}, we have $a_{t+1} = a_t$, because $a_{t+1} = Q_o W_t P_o(x - \mu) + Q_o b_t + Q_o W_t P_o(\mu - \mu_o) = Q_o W_t P_o(x - \mu_o) + Q_o b_t = a_t$. The second constraint can be satisfied similarly as above.

Eqn.(4) is able to preserve norm of the parameter matrix. We have $\|W_{t+1}\|_2 \approx \|W_t\|_2$, implying that $Q^{-1}Q_o \approx P_o P^{-1} \approx I$, as the distribution of a learned representation typically varies smoothly within a short period. $Q^{-1}Q_o$ and $P_o P^{-1}$ rarely scale up the parameters. In contrast, eigen-construction may scale up the gradient.

Update $\mathbf{W_t}$ by Shrinking Gradient. As shown in Fig.1 (b), gradient ΔW_t is attained by $\Delta W_t = h_1 \delta h_2{}^\mathsf{T} = P(x - \mu)(\delta a - \eta)^\mathsf{T} Q$, which develops into $S^{-\frac{1}{2}}U^\mathsf{T}\mathrm{diag}(\Sigma)^{-\frac{1}{2}}(x - \mu)(\delta a - \eta)^\mathsf{T}\mathrm{diag}(\Upsilon)^{-\frac{1}{2}}V^\mathsf{T}E^{-\frac{1}{2}}$ by utilizing Eqn.(3). In this case, $\|\Delta W_t\|_2$ is upper-bounded by the sub-multiplications of $\|S^{-\frac{1}{2}}\|_2 \cdot \|U^\mathsf{T}\|_2 \cdot \|\mathrm{diag}(\Sigma)^{-\frac{1}{2}}(x-\mu)(\delta a-\eta)^\mathsf{T}\mathrm{diag}(\Upsilon)^{-\frac{1}{2}}\|_2 \cdot \|V^\mathsf{T}\|_2 \cdot \|E^{-\frac{1}{2}}\|_2$. Specifically, all the three terms in the middle equal one, because of eigen-decomposition and standardization of FP and BP. Therefore, we have $\|\Delta W_t\|_2 \leq \|S^{-\frac{1}{2}}\|_2 \cdot \|E^{-\frac{1}{2}}\|_2 = 1/\sqrt{s_{\min} \cdot e_{\min}}$, where $s_{\min}$ and $e_{\min}$ indicate the smallest eigenvalues in the diagonal of S and E respectively. This implies that eigen-construction may scale up gradient by a significantly large factor, because $s_{\min}$ and $e_{\min}$ are typically small (*e.g.* smaller than 10^{-6} in experiments). As a result, conventional SGD leads to exploding of parameters after a few updates. To preserve gradient norm and stabilize training, we update W_t^ℓ for each layer ℓ by shrinking gradient

$$W_{t+1}^\ell = W_t^\ell - \big(\lambda\sqrt{(s_{\min}^\ell + \epsilon)(e_{\min}^\ell + \epsilon)}\big)\Delta W_t^\ell, \qquad (5)$$

where λ denotes a global learning rate and ϵ is a smoothing constant that prevents the gradient from over-penalizing. The shrinking term in Eqn.(5) is calculated at every γ iterations.

3.1 Connections with Previous Models

The relations between EigenNet, BN, and dropout are disclosed in Sec.2.1. It also closely connects to the natural gradient descent (NGD) [Amari and Nagaoka, 2000], which updates network parameters by $\Theta_{t+1} = \Theta_t - \lambda F^{-1}\Delta\Theta_t$, where $\Theta = \{W^\ell\}_{\ell=1}^L$ is a set of parameters of the entire network and F is the FIM, whose matrix inverse is usually computational impractical, especially for networks with large amount of parameters. Eqn.(5) addresses this issue. To understand, we rewrite Eqn.(5) as $W_{t+1}^\ell = W_t^\ell - \big(\lambda\sqrt{(s_{\min}^\ell + \epsilon)(e_{\min}^\ell + \epsilon)}\big)F_{\ell\ell}^{-1}\Delta W_t^\ell$, implying that FIM is approximated as L non-zero diagonal blocks, $\{F_{\ell\ell}\}_{\ell=1}^L \neq 0$. The entries of the non-diagonal blocks are zeros, $\{F_{\ell\ell'}\}_{\ell\neq\ell'} = 0$. As discussed in Sec.2.1, EigenNet naturally regularizes $F_{\ell\ell} = I$, mimicking fast convergence of NGD. The K-FAC [Martens and Grosse, 2015] method also employed the diagonal-block approximation of FIM, but it explicitly inverses each diagonal block matrix in gradient descent, rather than wrapping network architecture to reduce computations as EigenNet does.

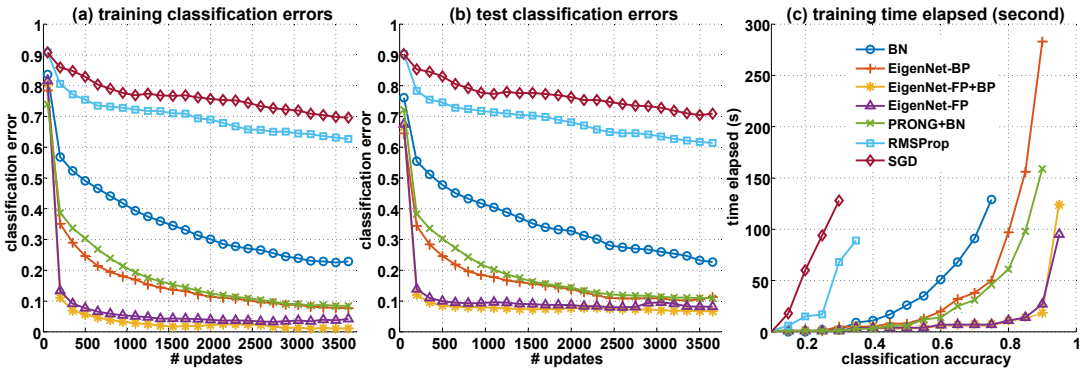

Figure 2: Comparisons on MNIST. (a) and (b) show the classification error vs number of updates in training and test respectively. (c) compares the runtime of training vs classification accuracy.

4 Experiments

EigenNet is extensively evaluated on three datasets, including an ablation study on MNIST [Larochelle *et al.*, 2007], a compatibility study with the recent advanced deep architectures, NIN [Lin *et al.*, 2014] and Inception [Ioffe and Szegedy, 2015], on the CIFAR-10 and CIFAR-100 datasets [Krizhevsky, 2009].

4.1 Ablation Study

Supervised Learning. We evaluate three variants of EigenNet, denoted as EigenNet-FP, EigenNet-BP, and EigenNet-FP+BP, which whiten the information flows of forward, backward, and both of them. They are compared with existing methods including SGD, RMSprop [Tieleman and Hinton, 2012], BN, and PRONG [Desjardins *et al.*, 2015] on MNIST with rotations. It contains 12,000 randomly rotated digit images for training and 50,000 images for test. All approaches are applied on the same convolutional neural network, containing four convolutional layers as shown in the first row of Table 1, where (x, y) denote number and size of the filter respectively. For example, 'conv0' learns 8 filters of size 5×5 to extract low-level features, while 'conv3' learns 10 filters to predict ten digit labels[1].

Other than the 'conv' layers, different approaches involve different additional layers as indicated in the second row of Table 1, including the layers whitening hidden features of FP and BP, denoted as 'fp' and 'bp'. Batch normalization layer is denoted as 'bn', where the subscript of 'bn_0' indicates that 'bn' is stacked after 'conv0'. For example, BN standardizes hidden features with three 'bn' layers. For PRONG, EigenNet-FP, and EigenNet-FP+BP, 'fp' layer is stacked behind two convolutional layers in the middle, *i.e.* 'conv1' and 'conv2', but not behind 'conv0' and 'conv3', because they already have compact feature channels. Similarly, 'bp' layers of EigenNet-BP and EigenNet-FP+BP are allocated behind 'conv1' and 'conv2'. Furthermore, PRONG is explicitly combined with BN to reduce instability during training, denoted as PRONG+BN.

Table 1: Comparisons of network specifications.

conv0 (8,5)	conv1 (128,1)			conv2 (128,1)			conv3 (10,1)		
	bn_0	fp_0	bp_1	bn_1	fp_1	bp_2	bn_2	fp_2	bp_3
BN	✓			✓			✓		
PRONG+BN	✓			✓	✓		✓	✓	
EigenNet-FP	✓				✓			✓	
EigenNet-BP	✓		✓			✓			
EigenNet-FP+BP	✓		✓		✓	✓		✓	

Results are given in Fig.2. For all algorithms, we initialize network parameters by sampling from a standard normal distribution and employ a batch of 64 samples for each update, where the parameters are updated with a momentum value of 0.9. In particular, for EigenNet, the whitening interval γ, the number of mini-batches τ, and the pruning threshold ρ are selected from {100,200,500,1000}, {10,20,30}, and {0.01,0.001}, respectively. For PRONG+BN, we consider $\gamma \in \{10, 10^2, 10^3, 10^4\}$ and a constant factor in {1,0.1,0.01,0.001}, which is added to all the eigenvalues. This factor penalizes small eigenvalues, preventing them from scaling up the gradients.

In Fig.2 (a) and (b), differences of the convergence rates among different methods are consistent in both training and test. In general, variants of EigenNet and PRONG+BN significantly outperform the other competitors by a large margin. This confirms the superiority of whitening transformation in training. In (c), EigenNet significantly outperforms the other methods in terms of runtime taken to reach a certain classification accuracy. For example, when training converged, EigenNet-FP reduces runtime compared to PRONG+BN and BN by more than 5 and 11 times respectively. This shows that although conditioning FIM in EigenNet and PRONG+BN needs more computations than conventional SGD, for instance, runtime of each feed-forward iteration of EigenNet-FP, PRONG+BN, and BN are 2×, 1.8×, and 1.3× that of SGD, updates produced by them make much more progress optimizing the objective, resulting in algorithms that can be much faster.

Analysis. To better understand the above results, the

[1]More specific, each 'conv' layer is followed by an activation function, $f(x) = \max(0, x)$. Two max-pooling layers are stacked after 'conv1' and 'conv2' respectively, reducing the spatial size of the channel by half. The output of 'conv3' is then pooled to 10×1×1 to predict 10 digit labels.

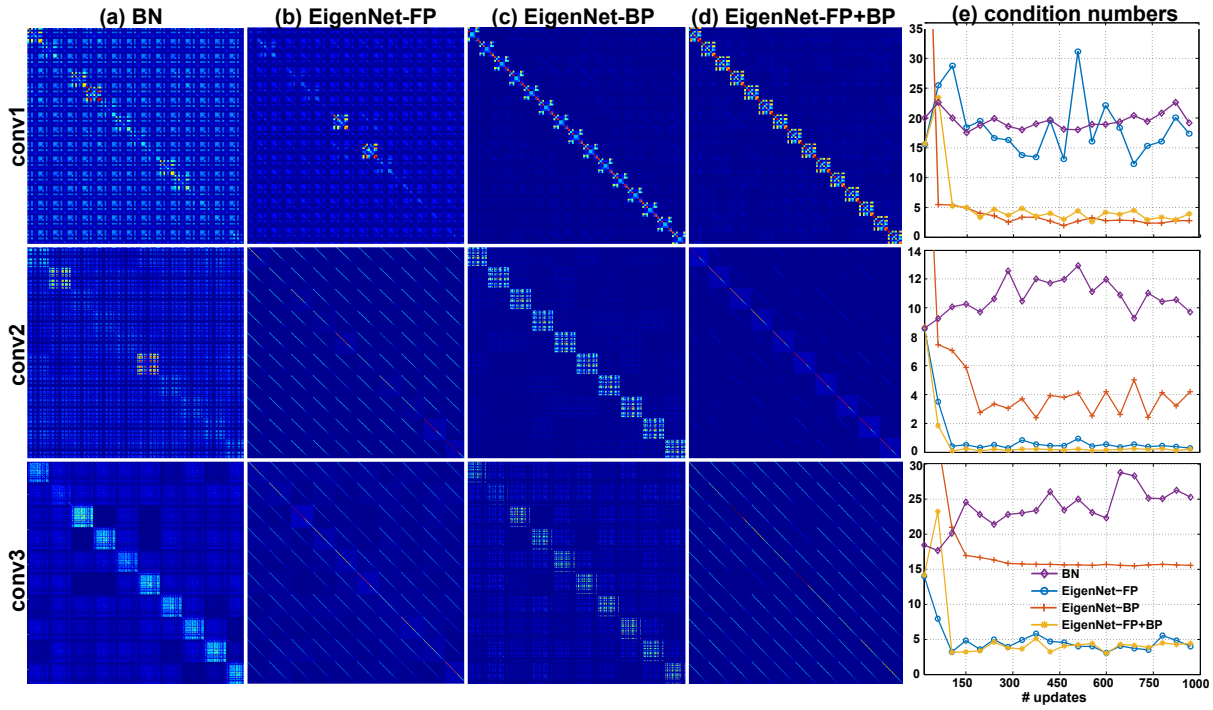

Figure 3: Comparisons of FIM blocks of different 'conv' layers and their condition numbers.

diagonal blocks of FIM corresponding to the 'conv' layers of three variants of EigenNet and BN are visualized in Fig.3 (a)-(d), where the FIM block of each layer is down-sampled to enhance visibility. Their condition numbers are compared in (e), measured by the percentage with respect to the value before the first time of eigen-construction. As shown in (a), FIM of BN has dense diagonal blocks and large condition numbers in training, indicating that standardization is not able to improve conditioning. In (b), FIM blocks of the last two 'conv' layers are constrained to be diagonal-stripped, because their input features are whitened, leading to small condition numbers. When backward errors are whitened, FIM blocks become block-diagonal as illustrated in the first two 'conv' layers of (c). In (d), three FIM blocks have good conditions, where the block of 'conv2' approximates an identity matrix because both the forward and backward flows are whitened.

Generative Unsupervised Learning. We evaluate EigenNet on reconstructing images of MNIST using auto-encoder (AE), which contains an encoder to transform an input image into a compact feature vector and a decoder to reconstruct the input image. In this study, the encoder has 500-200-30 hidden neurons. EigenNet is compared to SGD, BN, K-FAC, and PRONG+BN. For all methods, network parameters are initialized by sampling from a standard normal distribution and each weight vector is normalized to have unit l^2-norm. Learning rate and batch size are selected from $\{0.01, 0.001\}$ and $\{64, 128, 256\}$.

Reconstruction error/accuracy with respect to number of updates and wall-clock time are shown in Fig.4 (a,b). In general, the methods based on natural gradient, such as EigenNet, PRONG+BN, and K-FAC, achieve better performance.

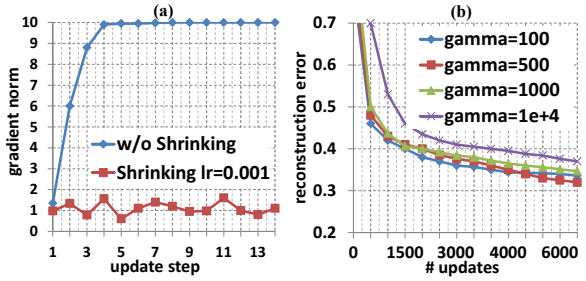

Figure 5: (a) l^2-norm of gradient vector trained with and without shrinking. (b) Comparisons between different values of gamma.

These approaches regularize either forward or backward pass to accelerate training. Specifically, EigenNet-FP+BP achieves the best performance, demonstrating large potential of whitening both forward and backward propagations in unsupervised learning, since the irregular of curvature is induced by both input and reconstructed images. We also observe that EigenNet-FP+BP usually attains the lowest reconstruction error. This confirms that better conditioning FIM produces better shaping the objective's landscape and more effective to minimize reconstruction error.

Fig.5 (a) visualizes l^2-norm of gradient vector with respect to step of update, when EigenNet is trained with/without shrinking gradient. We see that when the initial learning rate is 0.001, gradient norm is fluctuated slightly around one with shrinking, but explodes (clipped at 10) in a few steps without it. We observe that when parameters are also initialized to have unit norms, their norms can be naturally

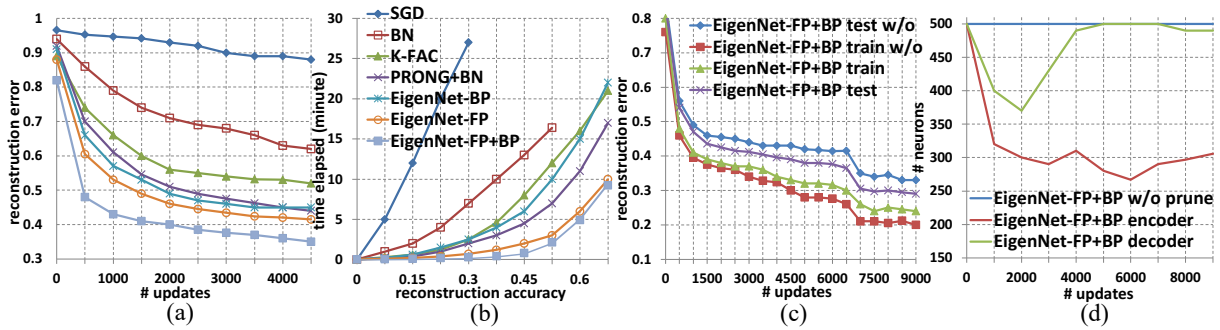

Figure 4: Reconstruction error with respect to number of updates and wall-clock time are presented in (a,b). EigenNet is compared to SGD, BN, K-FAC, and PRONG+BN. (c,d) reports the effectiveness of noise reduction by pruning hidden neurons.

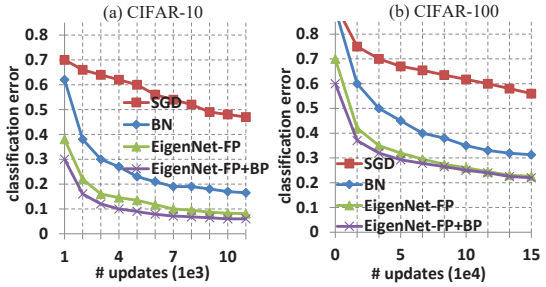

Figure 6: Comparisons of EigenNet and BN on (a) CIFAR-10 and (b) CIFAR-100.

preserved during training with a proper initial learning rate. In this case, parameters can be optimized with respect to a "unit sphere", yielding significant speedups often faster than conventional SGD. Fig.5 (b) evaluates reconstruction error with respect to the number of updates, using different values of gamma ('γ'). Performance of using large gamma such as 10^3 is comparable to those of using smaller ones. Performance can degrade due to ill-condition when gamma increases.

Pruning Neurons. We further study the effectiveness of noise reduction as shown in Fig.4 (c), where compares performances of EigenNet trained with/without noise reduction. We see that EigenNet with pruning substantially reduces over-fitting compared to those without it. Fig.4 (d) plots the numbers of neurons of two layers in encoder and decoder respectively. These layers are initialized with 500 neurons. With pruning, numbers of neurons vary differently during training. For instance, number of neurons of the encoder layer is removed by 40% to reduce noise. That of the decoder layer is decreased first as the encoder and increased afterwards. It automatically adjusts amount of hidden features to capture discriminative patterns instead of noises.

CIFAR. The compatibility of EigenNet with two popular network structures, NIN and Inception, are evaluated on CIFAR-10 and -100 respectively. We compare EigenNet-FP and EigenNet-FP+BP with SGD and BN. All models are trained by splitting a batch of 256 samples on 8 GPUs. The setting of EigenNet is similar to Sec.4.1. Fig.6 (a,b) plot classification error with respect to number of updates. EigenNet-FP and -FP+BP yield consistent improvements over SGD and BN on both datasets. For wall-clock time, for example, EigenNet-FP+BP spends 55% and 70% runtime of BN respectively when training converged.

5 Conclusions

This work presented EigenNet, a novel theoretically-justified network architecture that accelerates training and reduces over-fitting. It whitens both forward and backward flows of a neural network and stabilizes training with a carefully devised gradient descent update scheme. More importantly, it is able to remove noise in training and automatically adapt its network structure to the cleaned representation, which has not been explored in previous works. The effectiveness of EigenNet and its compatibility with existing deep architectures have been demonstrated in the experiments. The other future works include applying EigenNet on generative model such as generative adversarial nets and LSTMs, and regularizing the non-diagonal blocks of FIM in addition to the diagonal blocks, reducing co-adaptation across different hidden layers.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China (61503366, 61472410, U1613211), the National Key Research and Development Program of China (No.2016YFC1400700), the External Cooperation Program of BIC, Chinese Academy of Sciences (No.172644KYSB20160033), and the Science and Technology Planning Project of Guangdong Province (2015B010129013, 2014B050505017).

References

[Amari and Nagaoka, 2000] Shun-ichi Amari and Hiroshi Nagaoka. Methods of information geometry. In *Tanslations of Mathematical Monographs*, 2000.

[Desjardins *et al.*, 2015] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, and Koray Kavukcuoglu. Natural neural networks. In *NIPS*, 2015.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network

training by reducing internal covariate shift. In *ICML*, 2015.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. In *Technical Report*, 2009.

[Larochelle *et al.*, 2007] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, 2007.

[LeCun *et al.*, 1991] Yann LeCun, Ido Kanter, and Sara A. Sona. Second order properties of error surfaces: Learning time and generalization. In *NIPS*, 1991.

[LeCun *et al.*, 1993] Yann LeCun, Patrice Y. Simard, and Barak Pearlmutter. Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In *NIPS*, 1993.

[Lin *et al.*, 2014] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.

[Martens and Grosse, 2015] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *ICML*, 2015.

[Raiko *et al.*, 2012] Tapani Raiko, Harri Valpola, and Yann LeCun. Deep learning made easier by linear transformations in perceptrons. In *AISTATS*, 2012.

[Schraudolph, 2002a] Nicol N. Schraudolph. Centering neural network gradient factors. In *Neural Networks: Tricks of the Trade*, 2002.

[Schraudolph, 2002b] Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. In *Neural Computation*, 2002.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, 2014.

[Tieleman and Hinton, 2012] Tijmen Tieleman and Geoffrey Hinton. Rmsprop: Divide the gradient by a running average of its recent magnitude. In *Neural Networks for Machine Learning (Lecture 6.5)*, 2012.