Learning Convolutional Neural Networks using Hybrid Orthogonal Projection and Estimation

Hengyue Pan PANHY@CSE.YORKU.CA
Hui Jiang HJ@CSE.YORKU.CA
iFLYTEK Laboratory for Neural Computing and Machine Learning (iNCML), Department of Electrical Engineering and Computer Science, York University, Toronto, CANADA

Editors: Yung-Kyun Noh and Min-Ling Zhang

Abstract

Convolutional neural networks (CNNs) have yielded the excellent performance in a variety of computer vision tasks, where CNNs typically adopt a similar structure consisting of convolution layers, pooling layers and fully connected layers. In this paper, we propose to apply a novel method, namely Hybrid Orthogonal Projection and Estimation (HOPE), to CNNs in order to introduce orthogonality into the CNN structure. The HOPE model can be viewed as a hybrid model to combine feature extraction using orthogonal linear projection with mixture models. It is an effective model to extract useful information from the original high-dimension feature vectors and meanwhile filter out irrelevant noises. In this work, we present three different ways to apply the HOPE models to CNNs, i.e., *HOPE-Input*, *single-HOPE-Block* and *multi-HOPE-Blocks*. For *HOPE-Input* CNNs, a HOPE layer is directly used right after the input to de-correlate high-dimension input feature vectors. Alternatively, in *single-HOPE-Block* and *multi-HOPE-Blocks* CNNs, we consider to use HOPE layers to replace one or more blocks in the CNNs, where one block may include several convolutional layers and one pooling layer. The experimental results on CIFAR-10, CIFAR-100 and ImageNet databases have shown that the orthogonal constraints imposed by the HOPE layers can significantly improve the performance of CNNs in these image classification tasks (we have achieved one of the best performance when image augmentation has not been applied, and top 5 performance with image augmentation).

Keywords: Deep Learning, Neural Networks, HOPE

1. Introduction

Convolutional neural networks (CNNs) (LeCun et al., 1990) currently play an important role in the deep learning and computer vision fields. In the past several years, researchers have revealed that CNNs can give the state-of-the-art performance in many computer vision tasks, especially for image classification and object detection tasks (Krizhevsky et al., 2012a; Szegedy et al., 2014; Simonyan and Zisserman, 2015; Pan and Jiang, 2017). Comparing with the fully connected deep neural networks (DNNs), CNNs are superior in exploiting spatial constraints and in turn extracting better local features from input images using the convolution layers and weight sharing, and furthermore may provide better invariance through the pooling mechanism. All of these make CNNs very suitable for image-related tasks (LeCun and Bengio, 1995). Moreover, large-scale deep CNNs can be effectively learned end-to-end in a supervised way from a large amount of labelled images.

In the past several years, a tremendous amount of research efforts have been devoted to further improve the performance of deep CNNs. In (Hinton et al., 2012; Srivastava et al., 2014), the dropout method has been proposed to prevent CNNs from overfitting by randomly dropping a small portion of hidden nodes in the network during the training procedure. Many experiments have confirmed that the dropout technique can significantly improve the network performance, especially when only a small training set is available. Besides, a similar idea, called dropconnect (Wan et al., 2013), has been proposed to drop connections between layers instead of hidden nodes during the training stage. Another interesting research field is to design good nonlinear activation functions for neural networks beyond the popular rectified linear function (ReLU), such as maxout (Goodfellow et al., 2013) and PReLU (He et al., 2015), which are also demonstrated to yield improvement in terms of classification performance. On the other hand, another important path to improve model performance is to search for some new CNN structures. For example, in (Lin et al., 2013), Network in Network (NIN) has been proposed, in which one micro neural network is used to replace the regular linear convolutional filter. Recurrent Convolutional Neural Network (RCNN) (Liang and Hu, 2015) is another new CNN structure, which introduces recurrent connections into the convolution layers.

More recently, a novel model, called Hybrid Orthogonal Projection and Estimation (HOPE) (Zhang et al., 2016), has been proposed to learn fully-connected deep neural networks in either supervised or unsupervised ways. This model introduces a linear orthogonal projection to reduce the dimensionality of the raw high-dimension data and then uses a finite mixture distribution to model the extracted features. By splitting the feature extraction and data modeling into two separate stages, it may derive a good feature extraction model that can generate better low-dimension features for the further learning process. More importantly, based on the analysis in (Zhang et al., 2016), the HOPE model has a tight relationship with neural networks since each hidden layer of DNNs can also be viewed as a HOPE model being composed of the feature extraction layer and data modeling layer. Therefore, the maximum likelihood based unsupervised learning as well as the minimum cross-entropy error based supervised learning algorithms can be used to learn neural networks under the HOPE framework for deep learning. In this case, the standard back-propagation method may be used to optimize the objective function to learn the models except that the orthogonal constraints are imposed for all projection layers during the training procedure.

However, (Zhang et al., 2016) has not taken CNNs into account but merely investigated the HOPE models for the fully connected neural networks and demonstrated good performance in the small MNIST data set. To make the HOPE model work for more image-related tasks, we need to consider how to combine the basic idea of the HOPE model with non-fully connected CNNs. In this paper, we extend the HOPE model to the popular CNNs by considering the special model structures of both convolution and pooling layers, and further consider how to introduce the orthogonal constraints into the CNN model structure and learn CNNs under the HOPE framework. The main contribution of this paper is to propose a suitable method to split one convolution layer into one HOPE projection layer and one HOPE model layer. The projection layer introduces orthogonal constraints into the convolution filters and removes the correlations from the feature maps of convolution layers, and the model layer can then model the projected vectors. Moreover, the proposed HOPE CNNs can be learned end-to-end via a modified error back-propagation algorithm. Specifically, we force the convolution filter in the projection layer becomes orthogonal during the weight updating process, then with the moving of the orthogonal convolution filter, most noise signals of each local part in the input feature maps can be removed. The proposed HOPE CNN framework is technically novel and significantly differs from previous HOPE DNN models, because the projection layers of HOPE CNNs work on the convolution filters and the model layers not only consider single projected vector, but also its neighbors. The most straightforward idea is to use a HOPE layer as the first hidden layer in CNNs to de-correlate the high-dimension input CNN features and remove the irrelevant noises, which is called a HOPE-Input layer. This idea is similar as the original formulation in (Zhang et al., 2016) except the HOPE model is applied to each convolutional filter. Moreover, we may introduce even more HOPE layers into the CNNs for better performance. Generally speaking, we can split one CNN into several building blocks, and each block may include several convolutional layers and end with one pooling layer. In practice, we can either replace one block (single-HOPE-Block CNNs) or multiple blocks (multi-HOPE-Blocks CNNs) by using HOPE layers for better performance.

Our experimental results on CIFAR-10, CIFAR-100 and ImageNet databases have shown that the application of HOPE layers results in significant performance improvement over the regular CNN baseline models.

2. Hybrid Orthogonal Projection and Estimation (HOPE) Framework

In the original Hybrid Orthogonal Projection and Estimation (HOPE) formulation (Zhang et al., 2016), it is assumed that any high-dimension feature vector can be modelling by a hybrid model consisting of feature extraction using a linear orthogonal projection and statistic modeling using a finite mixture model. Assuming that each high-dimension feature vector $\mathbf{x}$ is of dimension D, then the linear orthogonal projection maps $\mathbf{x}$ to an M-dimension feature space ($M < D$), and the projected vector may retain the most useful information of $\mathbf{x}$. Specifically, we define a $D \times D$ orthogonal matrix $[\mathbf{U}; \quad V]$ which satisfies:

$$[\mathbf{z}; \quad \mathbf{n}] = \begin{bmatrix} \mathbf{U} \\ V \end{bmatrix} \mathbf{x} \tag{1}$$

where $\mathbf{z}$ is an M-dimension vector, called the signal component, and $\mathbf{n}$ is the residual noise vector with the dimensionality of $D - M$.

In practice, $\mathbf{z}$ is heavily de-correlated but it may still locate in a rather high dimension feature space. In the HOPE formulation, it is proposed to model $\mathbf{z}$ with a finite mixture model:

$$p(\mathbf{z}) = \sum_{k=1}^{K} \pi_k \cdot f_k(\mathbf{z}|\theta_k) \tag{2}$$

where K is the number of mixture components, π_k is the mixture weight of the kth component ($\sum_{k=1}^{K} \pi_k = 1$), $f_k()$ denotes a selected distribution from the exponential family, and θ_k denotes all model parameters of $f_k()$. As discussed in (Zhang et al., 2016), if the von Mises-Fisher (vMF) distribution is chosen for $f_k()$, the resultant HOPE model is equivalent in mathematical formulation to a hidden layer in neural networks using the popular rectified linear units (ReLU).

The HOPE model combines a linear orthogonal projection and a finite mixture model under a unified generative modeling framework. It can be learned unsupervisingly based on maximum likelihood estimation from unlabelled data as well as discriminatively from labelled data. In (Zhang et al., 2016), the HOPE model has been applied to the fully connected DNNs, and the models can be learned in either supervised or unsupervised ways. For one hidden layer with input vector $\mathbf{x}$ ($\mathbf{x} \in R^D$) and output vector $\mathbf{y}$ ($\mathbf{y} \in R^G$), it is first split into two layers: i) The first layer is a linear orthogonal projection layer, which is used to project $\mathbf{x}$ to a feature vector $\mathbf{z}$ ($\mathbf{z} \in R^M, M < D$) and remove the noise signals by using an orthogonal projection matrix $\mathbf{U}$:

$$\mathbf{z} = \mathbf{U}\mathbf{x}. \tag{3}$$

ii) The second layer is a non-linear model layer, which convert $\mathbf{z}$ to the output vector $\mathbf{y}$ following the selected model $f_k()$ and a nonlinear log-likelihood pruning operation (in supervised learning the model can be learned automatically from the training dataset). An example of a HOPE layer in DNNs is shown in Figure 1.

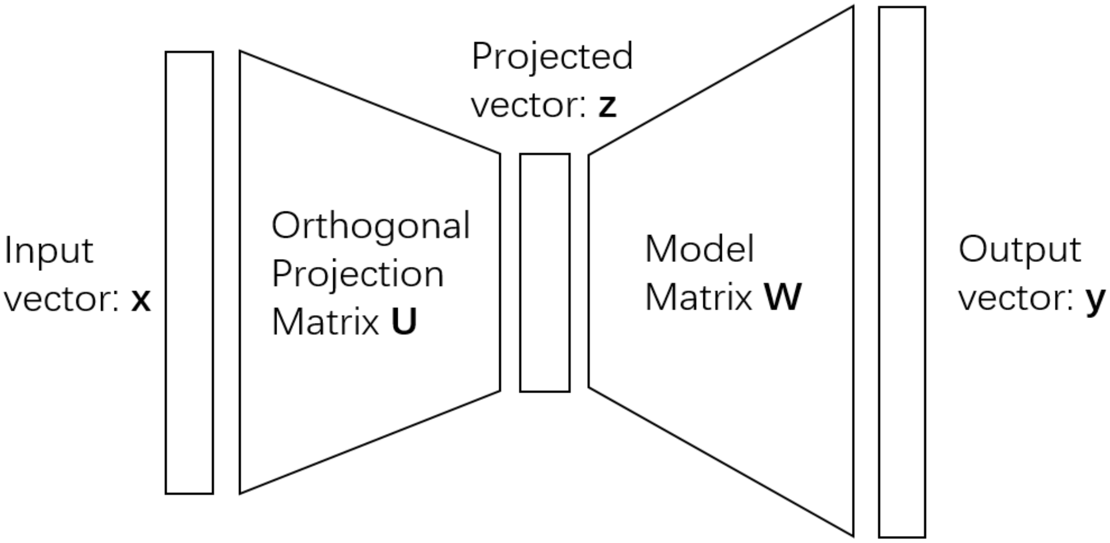

Figure 1: The HOPE model is viewed as a hidden layer in DNNs.

As in (Zhang et al., 2016), all HOPE model parameters, including the projection matrix U and the model matrix W, can be learned, using the error back-propagation algorithm with stochastic gradient descent, to optimize an objective function subject to an orthogonal constraint, $\mathbf{U}\mathbf{U}^T = \mathbf{I}$, for each projection layer. As in (Zhang et al., 2016), for computational simplicity, the constraint is cast as the following penalty term to gradually de-correlate the matrix $\mathbf{U}$ during the learning process:

$$P(\mathbf{U}) = \sum_{i=1}^{M} \sum_{j=i+1}^{M} \frac{|\mathbf{u}_i \cdot \mathbf{u}_j|}{|\mathbf{u}_i| \cdot |\mathbf{u}_j|}. \tag{4}$$

In (Zhang et al., 2016), both unsupervised learning and supervised learning are studied for DNNs under the HOPE framework. The above orthogonal constraint is found to be equally important in both scenarios. In this paper, we will study how to supervisingly learn CNNs under the HOPE formulation and more specifically investigate how to introduce the orthogonality into the CNN model structure.

3. Our Proposed Method

In (Zhang et al., 2016), the authors have applied the HOPE model to the fully connected DNNs and have achieved good performance in experiments on small databases like MNIST. However, more widely used neural models in computer vision, i.e. convolutional neural networks (CNNs), have not been considered. Unlike DNNs, CNNs adopt some unique model structures and have achieved huge successes in many large-scale image classification tasks. Therefore, it is interesting to consider how to combine the HOPE model with CNNs to further improve image classification performance.

3.1. Applying the HOPE model to CNNs

To introduce the HOPE model to CNNs, the most straightforward solution is to split each convolution layer into a concatenation of a projection layer and a model layer and impose the orthogonal constraints onto the projection layer as in (Zhang et al., 2016). Assuming that we have a regular convolution layer in CNNs, which uses some $S \times S$ linear filters to map from C_i input feature maps to C_m output feature maps. As shown in Figure 2, under the HOPE framework, we propose to split this convolution layer into the two separate layers:

i) One linear orthogonal projection layer with the projection matrix $\mathbf{U}$: in the projection layer, we use each local region that is 'covered' by the orthogonal convolution filter in the input feature maps as the basic unit of the orthogonal projection. Specifically, the orthogonal convolution filter linearly maps a 3-dimension tensor with the size of $S \times S \times C_i$ into a vector $1 \times 1 \times C_p$, C_p denotes the number of feature maps to be used in the projection layer. As the projection filters convolve with the input layer, it generates a total of C_p feature maps in the projection layer. The projection filter itself is a 4-dimension tensor with the size of $S \times S \times C_i \times C_p$. Based on the definition of the convolution procedure and follow the formulation in (Zhang et al., 2016), we can reshape this 4-dimension tensor as a matrix $\mathbf{U}$ with the size of $(S \cdot S \cdot C_i) \times C_p$, as shown in Figure 2. Notice that we do not apply any non-linear activation function in the linear orthogonal projection layer.

ii) One non-linear model layer with the weight matrix W: it has exactly same structure as a regular convolutional layer, which maps the C_p projected feature maps into C_m output feature maps. Differing from (Zhang et al., 2016), instead of only mapping the projected vector, the proposed model layer here takes all projected vectors within each $S \times S$ region into account and map all projected features within this region into the final output feature maps. We have found that this modification is critical in CNNs for better performance in image classification since it helps the network to extract local features. In our implementation, we use ReLU as the non-linear activation function. Since we apply supervised learning method to learn HOPE CNNs, we do not need to explicitly define the mixture model, and the mixture model can be learned automatically from training data.

Figure 2 shows the whole structure of one HOPE layer in CNNs. Since the projection layer is linear, we may collapse these two layers to derive a normal convolution layer in CNNs. However, as argued in (Zhang et al., 2016), the HOPE framework provides many advantages by explicitly define the feature extraction stage and data modeling stage.

Note that C_p is always far less than $S \times S \times C_i$ in the above HOPE formulation, it implies that the orthogonal projection may help to remove irrelevant noises in this step.

In this paper, we only consider the supervised learning of CNNs under the HOPE framework. In this case, the model parameters in the model layer can be learned in the same way as in the convolutional CNNs. However, for the projection layers, we need to impose the orthogonal constraint, i.e. $\mathbf{U}\mathbf{U}^T = \mathbf{I}$, during the learning process. Following (Zhang et al., 2016), we cast this constraint as a penalty term in Eq. (4).

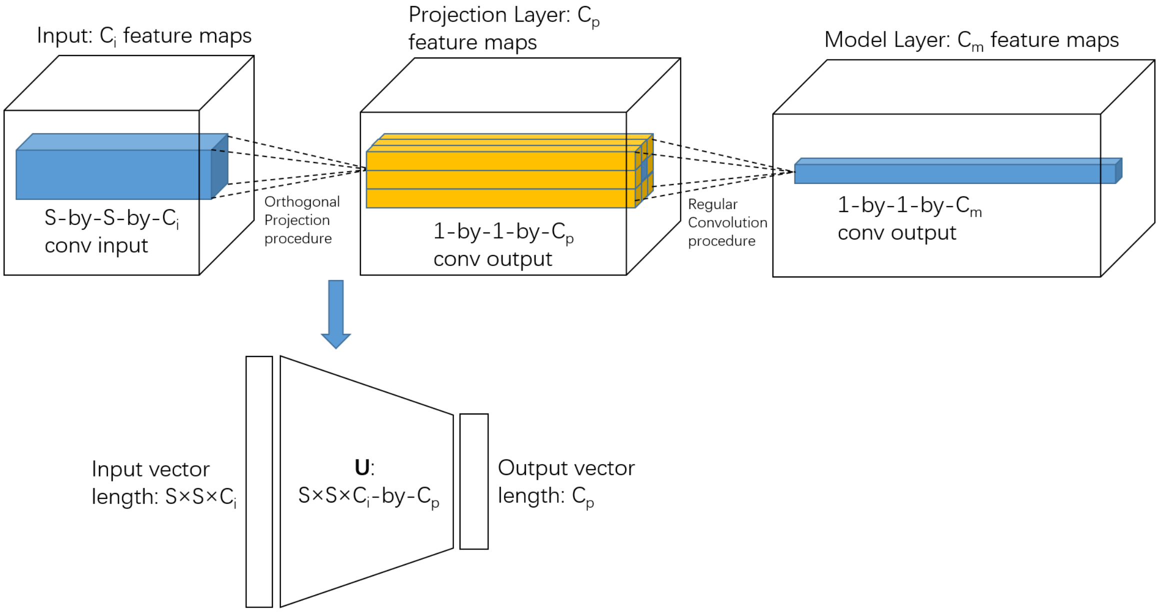
Input: C_i feature maps
Projection Layer: C_p feature maps
Model Layer: C_m feature maps
S-by-S-by-C_i conv input
Orthogonal Projection procedure
1-by-1-by-C_p conv output
Regular Convolution procedure
1-by-1-by-C_m conv output
Input vector length: S×S×C_i
U: S×S×C_i-by-C_p
Output vector length: C_p

Figure 2: A convolution layer in CNNs can be converted into a HOPE model. We do not need to explicitly define the data distribution of the model layer since it can be learned automatically via supervised learning criteria.

First of all, we need to derive the gradient of the penalty term $P(\mathbf{U})$ with respect to $\mathbf{U}$ as follows:

$$\frac{\partial P(\mathbf{U})}{\partial \mathbf{u}_i} = \sum_{j=1}^{M} (\frac{|\mathbf{u}_i \cdot \mathbf{u}_j|}{|\mathbf{u}_i| \cdot |\mathbf{u}_j|}) \cdot \Big((\frac{\mathbf{u}_j}{\mathbf{u}_i \cdot \mathbf{u}_j}) - (\frac{\mathbf{u}_i}{\mathbf{u}_i \cdot \mathbf{u}_i}) \Big) \tag{5}$$

To facilitate the above computation in GPUs, we may equivalently represent the above gradient computation as a matrix form, i.e., essentially a multiplication of the two matrices $\mathbf{D}$ and $\mathbf{B}$ as follows:

$$\frac{\partial P(\mathbf{U})}{\partial \mathbf{U}} = (\mathbf{D} - \mathbf{B})\mathbf{U} \tag{6}$$

where $\mathbf{D}$ is an M-by-M matrix of $d_{ij} = \frac{\text{sign}(\mathbf{u}_i \cdot \mathbf{u}_j)}{|\mathbf{u}_i| \cdot |\mathbf{u}_j|}$ $(1 < i, j < M)$ and $\mathbf{B}$ is another M-by-M diagonal matrix of $b_{ii} = \frac{\sum_j g_{ij}}{\mathbf{u}_i \cdot \mathbf{u}_i}$ with $g_{ij} = \frac{|\mathbf{u}_i \cdot \mathbf{u}_j|}{|\mathbf{u}_i| \cdot |\mathbf{u}_j|}$ $(1 < i, j < M)$.

Secondly, we can combine the above $\frac{\partial P(\mathbf{U})}{\partial \mathbf{U}}$ with the gradient $\Delta\mathbf{U}$, which is calculated from the objective function:

$$\widetilde{\Delta\mathbf{U}} = \Delta\mathbf{U} + \beta \cdot \frac{\partial P(\mathbf{U})}{\partial \mathbf{U}} \tag{7}$$

where β is a pre-defined parameter to balance the orthogonal penalty term. Finally, the projection matrix $\mathbf{U}$ can be updated as follows:

$$\mathbf{U}^{(n)} = \mathbf{U}^{(n-1)} - \gamma \cdot \widetilde{\Delta\mathbf{U}} \tag{8}$$

where γ is the learning rate for the weight update. During the learning process, $\mathbf{U}$ is gradually de-correlated and eventually becomes an orthogonal matrix. In Figure 3, we display all correlation coefficients, i.e., $\frac{|u_i \cdot u_j|}{|u_i| \cdot |u_j|}$, of the HOPE orthogonal projection matrix $\mathbf{U}$ and the corresponding linear projection matrix, in which the orthogonal constraints are removed. The two images in Figure 3 clearly show that the HOPE projection matrix removes most of the correlation and thus becomes orthogonal. As shown in the results, the proposed HOPE layer in CNNs may remove the noise signals from the feature maps.

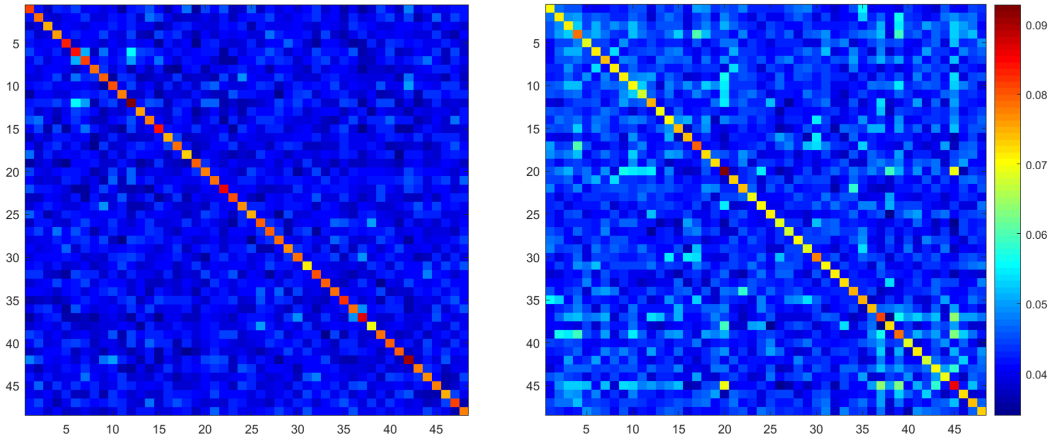

Figure 3: The correlation coefficients of the HOPE orthogonal projection matrix (left) and the corresponding linear projection matrix (right). We can see the HOPE projection matrix is more orthogonal compare with its linear counterpart. Here $S = 3$, $C_i = 64$ and $C_p = 48$.

3.2. The HOPE-Input Layer

The first way to apply the HOPE model to CNNs is to use the above HOPE layer to replace the first convolution layer right after the image pixel input. The HOPE formulation may help to de-correlate the raw image pixel inputs and filter out irrelevant noises in the first place. This is called as one HOPE-Input layer.

3.3. HOPE-Blocks

In many cases, simply applying one HOPE-Input layer is not enough to remove noise signals from features and achieve good performance. Therefore, we need to introduce more HOPE layers into the baseline CNN. In practice, one CNN can be divided into some building blocks, and each block may include several convolutional layers and end with one pooling layer. We can use these blocks as the basic units to introduce HOPE layers. Figure 4 shows an example of one HOPE-Block, and here we apply three HOPE layers to replace the corresponding convolutional layers (for the first convolutional layer, the projection layer is from the last block).

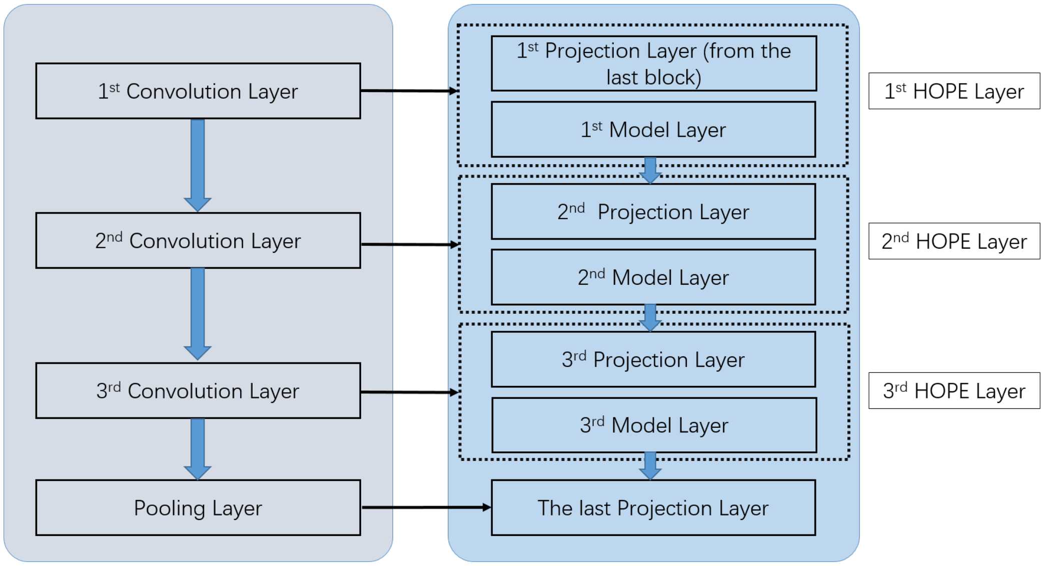

Figure 4: One HOPE-Block

For the pooling layer, we may need to consider a slightly different way to apply HOPE model. In CNNs, the pooling layers (Krizhevsky et al., 2012b) are traditionally considered as an important part for good performance. (Springenberg et al., 2014) has shown that the pooling layers result in the reduction of feature dimensionality, which help the CNNs to *view* much larger regions of the input feature maps, and generate more stable and invariant high level features.

Since the projection layer in HOPE models shares the similar objection with pooling layers, i.e., reducing the feature dimensionality, remove noise and increase the stability of the feature, we can just use one HOPE projection layer to replace one pooling layer, and view the next convolutional layer as the model layer. Comparing with the regular pooling layers, we believe that the HOPE projection layer may be advantageous in feature extraction since the learnable linear orthogonal projection may help to de-correlate the input feature maps more precisely and generate better features for the upper layers.

In practice, we can just introduce one HOPE-Block to replace the first building block in the baseline CNN (single-HOPE-Block), or apply multiple HOPE-Blocks (multi-HOPE-Blocks).

4. Experiments

In this paper, we use three widely used image classification databases, namely CIFAR-10. CIFAR-100 (Krizhevsky and Hinton, 2009) and ImageNet (Deng et al., 2009), to evaluate the performance of our proposed HOPE methods [1].

4.1. Databases

CIFAR-10 and CIFAR-100 are two popular databases that are widely used in computer vision. Both databases contain 50,000 32-by-32 RGB images for training and 10,000 images for validation. The

[1] The codes of the proposed method can be downloaded via: https://github.com/mowangphy/HOPE-CNN.

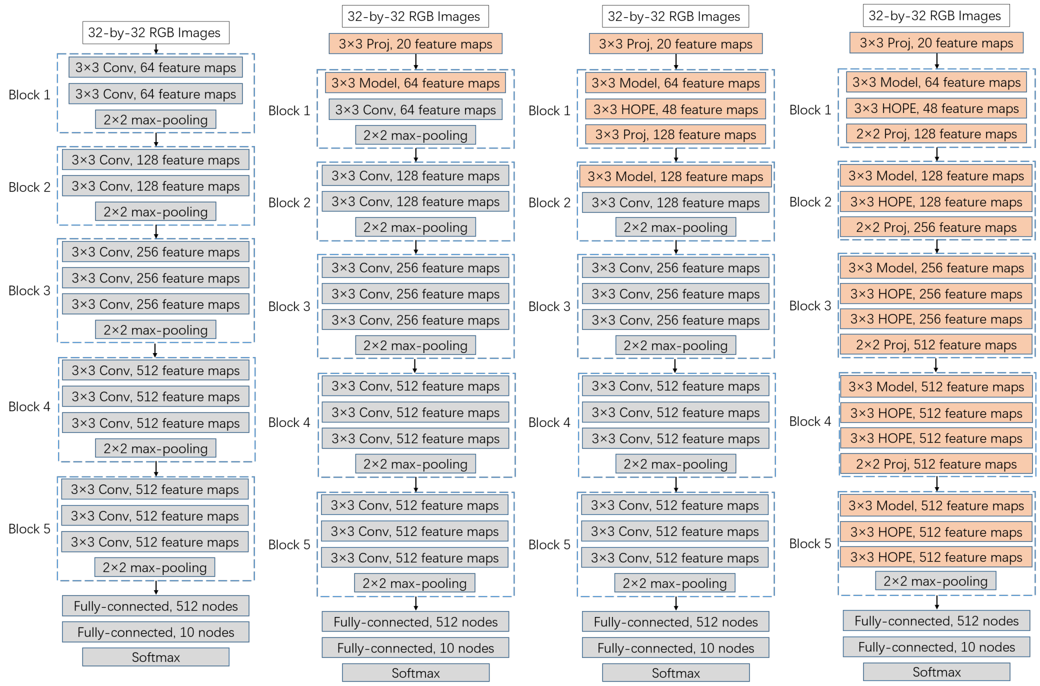
32-by-32 RGB Images
Block 1
3×3 Conv, 64 feature maps
3×3 Conv, 64 feature maps
2×2 max-pooling
Block 2
3×3 Conv, 128 feature maps
3×3 Conv, 128 feature maps
2×2 max-pooling
Block 3
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
2×2 max-pooling
Block 4
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Block 5
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Fully-connected, 512 nodes
Fully-connected, 10 nodes
Softmax
32-by-32 RGB Images
3×3 Proj, 20 feature maps
Block 1
3×3 Model, 64 feature maps
3×3 Conv, 64 feature maps
2×2 max-pooling
Block 2
3×3 Conv, 128 feature maps
3×3 Conv, 128 feature maps
2×2 max-pooling
Block 3
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
2×2 max-pooling
Block 4
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Block 5
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Fully-connected, 512 nodes
Fully-connected, 10 nodes
Softmax
32-by-32 RGB Images
3×3 Proj, 20 feature maps
Block 1
3×3 Model, 64 feature maps
3×3 HOPE, 48 feature maps
3×3 Proj, 128 feature maps
Block 2
3×3 Model, 128 feature maps
3×3 Conv, 128 feature maps
2×2 max-pooling
Block 3
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
3×3 Conv, 256 feature maps
2×2 max-pooling
Block 4
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Block 5
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
3×3 Conv, 512 feature maps
2×2 max-pooling
Fully-connected, 512 nodes
Fully-connected, 10 nodes
Softmax
32-by-32 RGB Images
3×3 Proj, 20 feature maps
Block 1
3×3 Model, 64 feature maps
3×3 HOPE, 48 feature maps
2×2 Proj, 128 feature maps
Block 2
3×3 Model, 128 feature maps
3×3 HOPE, 128 feature maps
2×2 Proj, 256 feature maps
Block 3
3×3 Model, 256 feature maps
3×3 HOPE, 256 feature maps
3×3 HOPE, 256 feature maps
2×2 Proj, 512 feature maps
Block 4
3×3 Model, 512 feature maps
3×3 HOPE, 512 feature maps
3×3 HOPE, 512 feature maps
2×2 Proj, 512 feature maps
Block 5
3×3 Model, 512 feature maps
3×3 HOPE, 512 feature maps
3×3 HOPE, 512 feature maps
2×2 max-pooling
Fully-connected, 512 nodes
Fully-connected, 10 nodes
Softmax

Figure 5: From Left to Right: Baseline CNN, HOPE-Input CNN, single-HOPE-Block CNN, and multi-HOPE-Blocks CNN (Using CIFAR experiments as examples), where *Proj* denotes one HOPE projection layer, *Model* denotes one HOPE model layer, and *HOPE* denotes one whole HOPE layer (includes one projection layer and one model layer).

main difference between these two databases is that CIFAR-10 only divides all images into 10 coarse classes, but CIFAR-100 divides them into 100 fine classes. All CIFAR data should be zero-mean normalized, and we did not apply data whitening to pre-process the training and test data.

To expand the training sample size and reduce over-fitting, we also consider to use data augmentation techniques on the two databases. Specifically, In each mini-batch, we will randomly select half of the images to apply four kinds of augmentation methods respectively:

- **Translation:** The selected images will be randomly translate horizontally and vertically for at most 5 pixels.
- **Rotation:** The selected images will be randomly rotated by -5 to 5 degrees. The rotated images should be cropped to keep the original size.
- **Scaling:** We firstly randomly extract a patch from the input image (the patch size is pre-defined), and resize the patch to the original image size. This procedure equals to zoom-in.
- **Color space translation:** For each channel of one image, we define a translation matrix. Each element in the translation matrix is corresponding to one pixel in the corresponding color channel, and the value lays between 0.95 and 1.05. The image will element-wise multiply with the translation matrix for the color space translation.

Figure 6 displays some examples of the selected data augmentation methods.

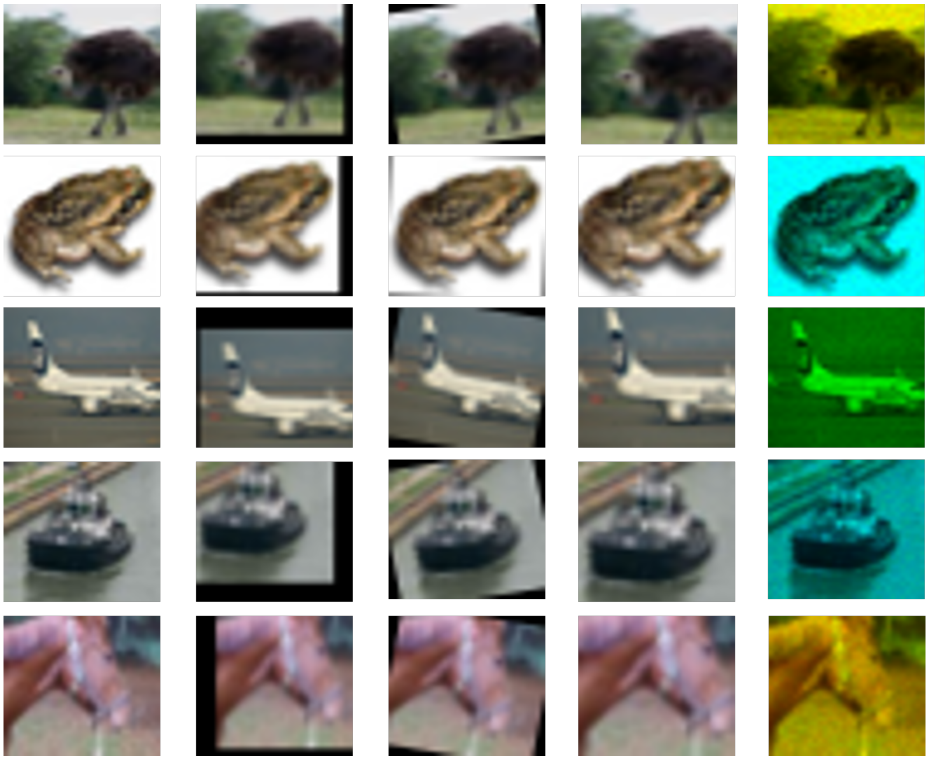

Figure 6: From Left to Right: original images, translated images, rotated images, scaled images and color space translated images.

Comparing with CIFAR-10 and CIFAR-100, ImageNet is an image database with much larger sample size (nearly 1.3 million training images and 50,000 validation images in the classification tasks), and all images are divided into 1000 classes. During the experiments, all images should be re-scaled to 224-by-224 to feed into CNNs.

4.2. CIFAR Experiments

In our CIFAR-10 and CIFAR-100 experiments, we consider several different CNN structures as specified in Figure 5 in detail. Firstly, we follow the CNN structure that is defined by Sergey Zagoruyko as our baseline CNNs.[2] Then we evaluate the HOPE-Input CNN, single-HOPE-Block and multi-HOPE-Block CNNs as discussed in section 3, and compare them with the baseline models. In Figure 5, we have provided the detailed description of the structure of 4 CNNs (baseline, HOPE-Input, single-HOPE-Block and multi-HOPE-Blocks) used in our experiments.

[2] See https://github.com/szagoruyko/cifar.torch for more information. According to the website, without using data augmentation, the best performance on the CIFAR-10 test set is 8.7% in error rate. By using RGB color channel instead of YUV, our reproduced baseline performance is 8.30% in this paper.

In Figure 5, for the HOPE-Input layer, we use 20 feature maps in the projection layer. For the whole HOPE layer in the Block 1 (single-HOPE-Block and multi-HOPE-Blocks CNNs), the projection layer contains 48 feature maps. For the rest HOPE layers in the multi-HOPE-Blocks CNN, the projection layers should have the same number of feature maps as the corresponding model layers. In practice, the feature map number of the input layer and the first block should be tuned very carefully. The reason of this fact is that the shallow layers tend to have highly correlated data, which may more sensitive to the size of projection layers. It is easy to learn that too many feature maps or too few feature maps can both lead to worse performance. We did several tests on HOPE-Input and single-HOPE-Block CNNs to determine the best scale of the projection layers, and the results are shown in Table 1.

Table 1: The relationship of the size of projection layers and classification performance (using CIFAR-10 tests as examples).

	Proj Layer 1	Proj Layer 2	Test Err
HOPE-Input	10	-	8.01%
	15	-	7.86%
	20	-	7.77%
	25	-	7.92%
Single-HOPE-Block	20	24	8.12%
	20	32	7.71%
	20	48	**7.06%**
	20	64	7.57%

To further investigate the performance of the HOPE CNNs (HOPE-Input, single-HOPE-Block and multi-HOPE-Blocks), we also consider the model configurations called as LIN CNNs (Lin-Input, single-Lin-Block and multi-Lin-Blocks), which uses the same model structure as the HOPE CNNs except that the orthogonal constraint in Eq. (4) is NOT applied in training.

In all CIFAR experiments, we use the mini-batch SGD with a batch size of 100 images to perform 350 epochs of network training. The initial learning rate is 0.065, and the learning rate should be halved after every 30 epochs. We also use momentum of 0.9 and weight decay rate of 0.0005. In batch normalization (Ioffe and Szegedy, 2015), we set $\epsilon = 0.001$. For the HOPE layers, we use an initial β that equals to 0.15, and the β should be divided by 1.75 after every 25 epochs. As shown in Table 2, the choices hyper parameter β and its decay rate may have some impacts on the final classification performance. Generally speaking, it is not very sensitive once we choose a good value for β. All weights in CNNs are initialized by using the method proposed by He et al (He et al., 2015). For the multi-HOPE-Blocks CNN experiments, we introduce 5 HOPE-Blocks (overall 6 blocks), since the last block only includes fully connected layers. In CNNs, fully-connected layers are applied at the end of the network. Our previous experiments show that adding HOPE layers on fully-connected layers cannot bring about positive influences on classification accuracy. One important reason is that the previous CNN layers (or HOPE CNN layers) already remove most of data correlation.

Table 2: The corresponding test error of different combination of initial β and its decay rate (the β should be decayed every 25 epochs).

Initial β	0.015	0.05	0.1	0.15	0.15	0.15	0.2
Decay Rate	1.75	1.75	1.75	1.50	1.75	2.00	1.75
Test Error	8.43%	8.31%	8.06%	7.46%	**7.06%**	7.37%	7.63%

4.2.1. LEARNING SPEED

We firstly consider the computational efficiency of the proposed HOPE methods in learning CNNs. Our computing platform includes Intel Xeon E5-1650 CPU (6 cores), 64 GB memory and a Nvidia Geforce TITAN X GPU (12 GB memory). Our method is implemented with MatConvNet (Vedaldi and Lenc, 2015), which is a CUDA based CNN toolbox in Matlab. The learning speed of all CNNs are listed in Table 3.

From Table 3, we can see that using the more complicated HOPE layers in CNNs will slow down the computation of CNNs in GPUs, but the speeds are still reasonably good. Moreover, the learning speed of the HOPE methods is similar with the corresponding LIN methods, which implies that the computational overhead for the orthogonal projection constraint is negligible in training.

Table 3: The learning speed of different CNNs on CIFAR experiments.

Methods	Learning Speed
Baseline	220 images/s
LIN-Input	206 images/s
HOPE-Input	203 images/s
Single-LIN-Block	200 images/s
Single-HOPE-Block	195 images/s
Multi-LIN-Blocks	149 images/s
Multi-HOPE-Blocks	141 images/s

4.2.2. PERFORMANCE ON CIFAR-10 AND CIFAR-100

We use the classification error rate on the test sets of the selected databases to evaluate the performance of all CNN models. Besides the 7 CNN configurations we mentioned above, we also include several well-known CNN models from the previous work to compare with our methods, including Tree-Pooling (Lee et al., 2015), Deep Networks for Global Optimization (DNGO) (Snoek et al., 2015), Fitnet4-LSUV (Mishkin and Matas, 2015), RCNN (Liang and Hu, 2015), ALL-CNN (Springenberg et al., 2014), Maxout Networks (Goodfellow et al., 2013) and Network in Network (Lin et al., 2013). All selected models have the comparable model size with our proposed CNNs.

From all results summarized in Table 4, we can see that the proposed HOPE-based CNNs models work well in both CIFAR-10 and CIFAR-100 databases. In the cases that the data augmentation methods are not applied, the single-HOPE-Block CNNs can achieve the best performances on both CIFAR-10 and CIFAR-100, which is the state-of-the-art performance without data augmentation,

Table 4: The classification error rates of all examined CNNs on the test set of CIFAR-10 and CIFAR-100. CIFAR-10+ and CIFAR-100+ denote the results using data augmentation.

	CIFAR-10	CIFAR-10+	CIFAR-100	CIFAR-100+
Baseline	8.30%	6.96%	30.71%	29.38%
LIN-Input	7.97%	6.88%	30.13%	28.91%
HOPE-Input	7.77%	6.65%	29.96%	28.79%
Single-LIN-Block	8.30%	7.21%	31.85%	30.27%
Single-HOPE-Block	**7.06%**	**5.89%**	**29.47%**	**26.99%**
Multi-LIN-Blocks	9.29%	8.16%	39.28%	35.52%
Multi-HOPE-Blocks	7.88%	6.47%	31.01%	27.59%
Tree-Pooling (Lee et al., 2015)	7.62%	6.05%	32.37%	-
DNGO (Snoek et al., 2015)	-	6.37%	-	27.40%
Fitnet4-LSUV (Mishkin and Matas, 2015)	-	6.06%	-	27.66%
RCNN (Liang and Hu, 2015)	8.69%	7.09%	31.75%	-
ALL-CNN (Springenberg et al., 2014)	9.08%	7.25%	33.71%	-
Maxout (Goodfellow et al., 2013)	11.68%	9.38%	34.54%	-
Network in Network (Lin et al., 2013)	10.41%	8.81%	35.68%	-

and much better compare with the selected strong baselines. Moreover, we can see that HOPE-Input, single-HOPE-Block and multi-HOPE-Blocks CNNs consistently outperform the counterpart LIN models that do not use the orthogonal constraints. This implies that the orthogonality introduced by the HOPE methods is quite useful to improve the performance of CNNs in the image classification tasks.

Table 4 also shows that after data augmentation the proposed HOPE method can also achieve state-of-the-art performance on both CIFAR-10 and CIFAR-100 databases.

In the supervised learning of HOPE CNNs, the projection layers in the shallow layers (from input layer to the first block) are most important since those layers can remove most of residual noises and data correlation from the raw data. Therefore, adding more HOPE layers (multi-HOPE-Blocks) may not bring about obvious improvements. Moreover, introducing HOPE layers in the top may introduce a lot of extra model parameters due to the large number of feature maps used in those layers. This may further lead to over-fitting and decrease the performance on the test sets. This can explain why single-HOPE-Block CNNs show better performance compare with multi-HOPE-Blocks CNNs.

4.3. ImageNet Experiments

In our ImageNet experiments, we directly apply VGG-16 (Simonyan and Zisserman, 2014) as the baseline network [3] Then we evaluate the HOPE-Input CNN and single-HOPE-Block CNN as discussed in section 3, and compare them with the original VGG-16 model. Similar to CIFAR experiments, we also take the corresponding LIN CNNs into account to further demonstrate the effectiveness of HOPE.

[3] See http://http://www.vlfeat.org/matconvnet/pretrained/ for more information. According to the website, the top-5 error of VGG-16 on ImageNet validation set is 9.5% on MatConvNet platform.

Table 5: The learning speed and top-5 classification error (validation set) of different CNNs on ImageNet experiments.

Methods	Learning Speed	Top-5 Error
VGG-16	58 images/s	9.5%
LIN-Input	56 images/s	9.5%
HOPE-Input	55 images/s	9.3%
Single-LIN-Block	53 images/s	9.6%
Single-HOPE-Block	52 images/s	**9.0%**

In all ImageNet experiments, we use 128 mini-batch size to train the CNNs for 50 epochs. The initial learning rate and β are 0.01 and 0.02 respectively, and both of them should be halved after every 5 epochs. The momentum and weight decay rate are 0.9 and 0.0005 respectively. In batch normalization (Ioffe and Szegedy, 2015), we set $\epsilon = 0.001$. All weights in CNNs will be initialized by using the method proposed by He et al (He et al., 2015).

4.3.1. Learning Speed

Since ImageNet database has very large sample size, and the CNNs are much deeper comparing with their counterparts in CIFAR experiments, we use 4 Nvidia Geforce TITAN X GPUs to do network training, and the learning speeds are listed in Table 5. In our platform, one single training round of ImageNet takes nearly 2 weeks. Therefore, we can not try all possible combinations of HOPE model as CIFAR experiments, but just the best configuration obtained from CIFAR (Single-HOPE-Block).

4.3.2. Performance on ImageNet

From Table 5, we can also see that HOPE models also work well on the ImageNet database. And Single-HOPE-Block CNN shows the best performance among all selected models. HOPE-Input CNN can also yield improvement over the VGG-16 baseline model.

5. Conclusion and Future Work

In this paper, we have proposed several methods to apply the recent HOPE model to CNNs for image classification. Experimental results on the CIFAR-10, CIFAR-100 and ImageNet databases have shown that our proposed HOPE methods work well with CNNs, and can yield the state-of-the-art classification performance in these databases. This study has confirmed that the orthogonal constraints imposed by the HOPE models can significantly improve the performance of CNNs in these image classification tasks.

Due to the limitation of time and computing resources, we haven't applied the propose HOPE CNN methods to the more recent deep residual nets (He et al., 2016). However, the application of HOPE CNN is straightforward since the HOPE model can also be inserted in the shortcut connections in deep residual nets. We will continue to investigate along this line as a possible direction to extend this work in the future.

References

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1319–1327, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

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

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012a.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012b.

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361, 1995.

Yann LeCun, Boser B., JS Denker, D Henderson, Richard E. Howard, W. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems (NIPS)*. Citeseer, 1990.

Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. *arXiv preprint arXiv:1509.08985*, 2015.

Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375, 2015.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

Hengyue Pan and Hui Jiang. A fast method for saliency detection by back-propagating a convolutional neural network and clamping its partial outputs. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 3585–3592. IEEE, 2017.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*. 2015.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mostofa Ali, Ryan P Adams, et al. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, 2015.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *arXiv preprint arXiv:1409.4842*. 2014.

Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.

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

Shiliang Zhang, Hui Jiang, and Lirong Dai. Hybrid orthogonal projection and estimation (HOPE): A new framework to learn neural networks. *Journal of Machine Learning Research*, 17(37):1–33, 2016. URL http://jmlr.org/papers/v17/15-335.html.