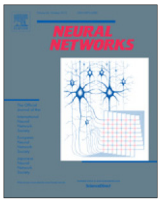
Neural Networks 105 (2018) 197–205

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Coupled convolution layer for convolutional neural network

Kazutaka Uchida[a,*], Masayuki Tanaka[a,b], Masatoshi Okutomi[a]

[a] Tokyo Institute of Technology, Tokyo, Japan
[b] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

ARTICLE INFO

Article history:
Received 5 March 2017
Received in revised form 20 April 2018
Accepted 1 May 2018
Available online 16 May 2018

Keywords:
Neural network
Learning and adaptive system
Classification

ABSTRACT

We propose a coupled convolution layer comprising multiple parallel convolutions with mutually constrained filters. Inspired by biological human vision mechanism, we constrain the convolution filters such that one set of filter weights should be geometrically rotated, mirrored, or be the negative of the other. Our analysis suggests that the coupled convolution layer is more effective for lower layer where feature maps preserve geometric properties. Experimental comparisons demonstrate that the proposed coupled convolution layer performs slightly better than the original layer while decreasing the number of parameters. We evaluate its effect compared to non-constrained convolution layer using the CIFAR-10, CIFAR-100, and PlanktonSet 1.0 datasets.

1. Introduction

Convolutional neural networks (CNN) have achieved great success in solving computer vision problems such as super-resolution (Dong, Loy, He, & Tang, 2014; He, Zhang, Ren, & Sun, 2016; Kim, Lee, & Lee, 2015a, b), tracking (Kristan et al., 2015), classification (Krizhevsky, Sutskever, & Hinton, 2012), image segmentation (Long, Shelhamer, & Darrell, 2015), and object recognition (Girshick, Donahue, Darrell, & Malik, 2014). Many studies have been performed to find key factors that improve performance by investigating the properties of networks (Bao & Zeng, 2016, 2017). Some studies have revealed that CNNs and the human vision mechanism have similar aspects. Jones and Palmer (1987) proved that some parts of the neural network of cat vision are activated by specific optical patterns, and they estimated that it has convolutional processes that are similar to Gabor filters. Trained filters in CNNs often form Gabor-like weights (Olshausen et al., 1996); thus the biological vision system and CNNs are considered to have similar properties when forming convolution filters.

Serre, Wolf, Bileschi, Riesenhuber, and Poggio (2007) introduced a cortex-like convolutional network that exploits Gabor filters in the first convolution layer by mimicking the primary visual cortex of primates. The Gabor filters are parameterized such that invariances to scale and rotation are formed as observed in the biological cortex.

Another attempt to apply human vision to a CNN exploits the behavior of retinal cells as an activation function. Kim, Kim, and Lee (2015) proposed an activation scheme called ON/OFF ReLU inspired by biological vision mechanism. ON/OFF ReLU mimics ON/OFF ganglion cell behavior that responds to positive and negative input separately.

Shang, Sohn, Almeida, and Lee (2016) achieved a similar activation scheme called concatenated ReLU (CReLU) based on their observation that filters in the lower layers of a CNN tend to form negatively-correlated pairs. By replacing ReLU with CReLU in CNN architectures, performance improved while training parameters were reduced by almost half.

In this paper, we extend the idea of ON/OFF ReLU and propose a coupled convolution layer that has multiple parallel convolutions with filter weights constrained by each other. First, we start from the human retina mechanism and model it as a retina-like convolution layer. Then, we generalize the retina-like convolution layer as a coupled convolution layer. We also analyze the trained filter of well-known networks to determine constraint types which efficiently reduce redundancy on filter weights for the coupled convolution layer. We evaluate the performance of models with coupled convolution layers using the CIFAR-10, CIFAR-100, and PlanktonSet 1.0 datasets in comparison with non-constrained layer models.

This paper is an extended version of work published in Uchida, Tanaka, and Okutomi (2016). We extend our previous work by making the notion more generic and giving thorough analysis on convolution weights.

In the previous paper, we performed only one type of coupled convolution layer with parametric constraint (i.e. negative constraint), that reduced the number of weight parameters in that

* Correspondence to: Department of Systems and Control Engineering, School of Engineering, Tokyo Institute of Technology, 2-12-1 O-okayama, Meguro-ku, Tokyo, 152-8550, Japan.
E-mail addresses: uchida@ok.sc.e.titech.ac.jp (K. Uchida), mtanaka@sc.e.titech.ac.jp (M. Tanaka), mxo@sc.e.titech.ac.jp (M. Okutomi).

https://doi.org/10.1016/j.neunet.2018.05.002
0893-6080/© 2018 Elsevier Ltd. All rights reserved.

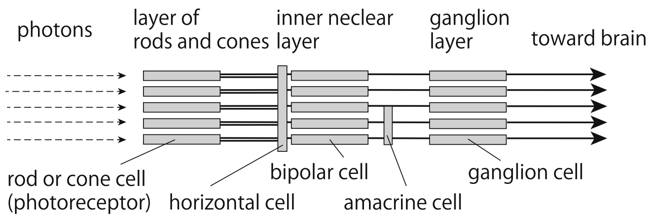
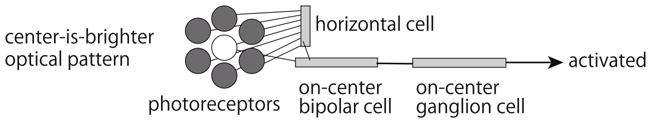
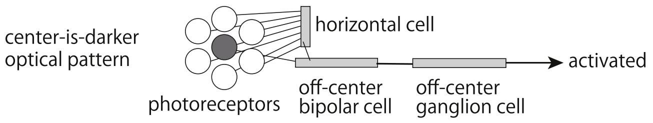


Fig. 1. Retina structure.



(a) On-center receptive field.



(b) Off-center receptive field.

Fig. 2. Two receptive field types.

layer by almost half. In this work, (1) we introduce geometric constraints such as rotation and mirroring as a variation of coupled convolution layer. (2) We confirmed through analysis that these geometric constraints are applicable to the coupled convolution layer. (3) With this insight, we introduce a coupled convolution layer with sixteen parallel coupled convolutions having sixteen types of constraint, and experimental result shows that the network with the proposed layer maintains its performance comparatively while the number of layer parameters are reduced significantly by one-sixteenth.

2. Retina-like convolution layer

We first present a retina-like convolution layer, derived from the human retina mechanism. We extend this notion to a coupled convolution layer in the next section.

2.1. Human retina mechanism

There are three layers in the human retina, as shown in Fig. 1 (Wandell, 1995). The first layer is a layer of rods and cones, where photons coming through the lens of the eye are captured by photoreceptors, i.e., rod cells and cone cells. The next layer is the inner nuclear layer where signals from the photoreceptors are received. This layer has three types of neural cells, i.e., bipolar, horizontal, and amacrine cells. Bipolar and horizontal cells synapse to photoreceptors to receive signals. Horizontal cells have lateral connections with photoreceptors, which makes the receptive field wide. The last layer is the ganglionic layer, where signals from the previous layer are transmitted to the brain by ganglion cells.

There are two types of ganglion cell with respect to the receptive field, i.e., on-center ganglion and off-center ganglion cells as shown in Fig. 2. On-center ganglion cells are activated by center-is-brighter optical patterns and off-center ganglion cells respond to center-is-darker optical pattern.

Relative to a CNN, the retina mechanism can be interpreted as follows. The photoreceptors can be considered as input pixels, the connections between photoreceptors and horizontal and bipolar cells are considered a filter, and the behavior of the ganglion cell

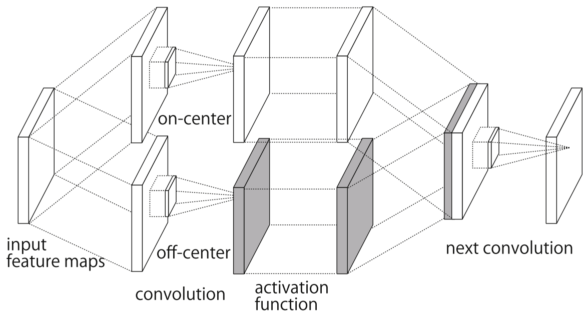


Fig. 3. Retina-like convolution layer.

is an activation function. Considering that there are two types of ganglion cell that react to center-is-brighter and center-is-darker optical patterns, there should be two types of convolutional filter with inversed weights.

2.2. Retina-like convolution layer

As mentioned previously, the retinal mechanism can be modeled as a convolution layer. Fig. 3 shows the retina-like convolution layer. The two parallel convolutions represent the on-center and off-center receptive fields. The two sets of output feature maps are merged for further convolutions.

The convolution layer, whose size of input and output feature map is $I \times J$, the number of input and output channel is C and K, and the size of filter is $S \times T$, can be represented as follows:

$$\begin{cases} y_{(+)ij}^{(k)} &= \sum_{c}\sum_{s,t} w_{(+)st}^{(k,c)} x_{i+s,j+t}^{(c)} + b_{(+)}^{(k)} \\ y_{(-)ij}^{(k)} &= \sum_{c}\sum_{s,t} w_{(-)st}^{(k,c)} x_{i+s,j+t}^{(c)} + b_{(-)}^{(k)}, \end{cases} \tag{1}$$

where $x_{ij}^{(c)}$ is an input value at position (i, j) of the cth input feature map ($i \in \{1, \dots, I\}, j \in \{1, \dots, J\}, c \in \{1, \dots, C\}$), $y_{(+)ij}^{(k)}$ and $y_{(-)ij}^{(k)}$ are the corresponding on-center and off-center output in the kth feature map ($k \in \{1, \dots, K\}$), $w_{(+)st}^{(k,c)}$ and $w_{(-)st}^{(k,c)}$ represent weights for computing the on-center and off-center outputs at (s, t) in the cth to kth kernel mapping channels ($s \in \{1, \dots, S\}, t \in \{1, \dots, T\}$), and $b_{(+)}^{(k)}$ and $b_{(-)}^{(k)}$ are the biases for the kth on-center and off-center channels, respectively.

After parallel convolutions, activation functions $f_{(+)}$ and $f_{(-)}$ map $y_{(+)ij}$ and $y_{(-)ij}$ as follows:

$$\begin{cases} z_{(+)ij} &= f_{(+)}(y_{(+)ij}) \\ z_{(-)ij} &= f_{(-)}(y_{(-)ij}), \end{cases} \tag{2}$$

where $f_{(+)}$ and $f_{(-)}$ can be any piecewise activation function such as sigmoid (Chen & Cao, 2009, 2016; Costarelli, Spigler, et al., 2013, 2014; Costarelli & Vinti, 2016; Hong & Hahm, 2016) or ReLU.

$z_{(+)ij}$ and $z_{(-)ij}$ are the on-center and off-center outputs from input x that correspond to retinal ganglion cell response.

From the perspective of retina-like convolution, it is reasonable to assign a constraint on the weights such that $w_{(+)ij}^{(k,c)} = -w_{(-)ij}^{(k,c)}$ can explicitly represent the on/off-center characteristics of the convolution. Consequently, Eq. (1) can be rewritten as follows:

$$\begin{cases} y_{(+)ij}^{(k)} &= \sum_{c}\sum_{s,t} w_{st}^{(k,c)} x_{i+s,j+t}^{(c)} + b_{(+)}^{(k)} \\ y_{(-)ij}^{(k)} &= -\sum_{c}\sum_{s,t} w_{st}^{(k,c)} x_{i+s,j+t}^{(c)} + b_{(-)}^{(k)}. \end{cases} \tag{3}$$

Eq. (3) is equivalent to a convolution layer with Biased ON/OFF ReLU (Uchida et al., 2016) as its activation schema, whereas corresponds to a convolution layer with ON/OFF ReLU (Kim et al., 2015) or CReLU (Shang et al., 2016) when $b^{(k)}_{(+)} = 0$ and $b^{(k)}_{(-)} = 0$.

3. Coupled convolution layer

Here, we generalize the retina-like convolution layer as a coupled convolution layer that has several parallel convolutions with mutually-constrained filter weights.

3.1. Filter weight constraint

In Eq. (1), the first and second convolutions are constrained through weight parameters. We refer to such convolution layers as coupled convolution layers.

Eq. (1) is generalized as a coupled convolution layer with N convolutions as follows:

$$\begin{cases} y^{(k)}_{(n)ij} &= \sum_{c} \sum_{s,t} w^{(k,c)}_{(n)st} x^{(c)}_{i+s,j+t} + b^{(k)}_{(n)} \\ \mathbf{w}^{(k,c)}_{(m)} &= \mathbf{H}^{(n,m)} \mathbf{w}^{(k,c)}_{(n)}, \end{cases} \tag{4}$$

where $\mathbf{w}^{(k,c)}_{(n)}$ is a matrix representation of the convolution kernel, and $\mathbf{H}^{(n,m)}$ is a one-to-one weight mapping matrix between the nth and mth convolution with geometric or parametric constraint ($n, m \in \{1, \dots, N\}$), which is described in the next subsection. Elements in $\mathbf{H}^{(n,m)}$ are set to $\{-1, 0, 1\}$ for simple one-to-one mapping.

With this constraint, the output channels on the nth and mth convolutions are associated with the constrained weights $\mathbf{w}_{(m)}$ and $\mathbf{w}_{(n)}$: thus, we refer to such convolution layers as coupled convolution layers.

Multiple constraints can also be applied such that:

$$\begin{aligned} \mathbf{w}_{(1)} &= \mathbf{H}^{(1)} \mathbf{w}_{(0)} \\ \mathbf{w}_{(2)} &= \mathbf{H}^{(2)} \mathbf{w}_{(0)} \\ &\cdots \\ \mathbf{w}_{(N)} &= \mathbf{H}^{(N)} \mathbf{w}_{(0)}, \end{aligned}$$

where $\mathbf{w}_{(0)}$ is a hidden trainable filter weights vector (canonical filter) and $\mathbf{H}^{(n)}$ is a weight constraint matrix for the nth convolution mapping from the canonical filter. Multiple constraints with N constraint matrix $\mathbf{H}$ can be interpreted as N pairs of convolutions: thus, we also refer to multi-constraint convolution layers as coupled convolution layers.

Constraint matrix $\mathbf{H}$ can be a mirrored, rotational, and geometric mapping of the weight parameters. It is also possible to impose a negative constraint by taking $\mathbf{H}$ as a negative identity matrix.

Fig. 4 shows a coupled convolution layer.

Filter weights that use a rotational constraint have been proposed by Marcos, Volpi, and Tuia (2016) to increase invariance to rotation. With a shallow CNN to classify texture patterns, they tie convolution filters by rotating a canonical filter from 0 to 2π radians at a step of $2\pi/S$ radians, where $S = 32$ in their experiments. Due to the small rotation step, which requires higher resolution in the filters, the filter size must be relatively large, such as 35×35, to prevent loss of accuracy.

In our context, the constraint matrix $\mathbf{H}$ is a rotational matrix with float numbers as its elements. On the other hand, our constraint matrix contains only $\{-1, 0, 1\}$ as an element, targeting simple geometrical patterns. As a result, the filter size can be small without loss of filter resolution.

Another difference with their network is that they introduce a max-pooling layer right after the convolution layer to explicitly obtain invariance to rotation, whereas our coupled convolution layer is not followed by a max-pooling layer. The proposed method attempts to generate filters trained symmetrically with respect to the given constraint as a means of regularization. Thus, the coupled convolution layer does not restrict any configuration of the network beyond the layer.

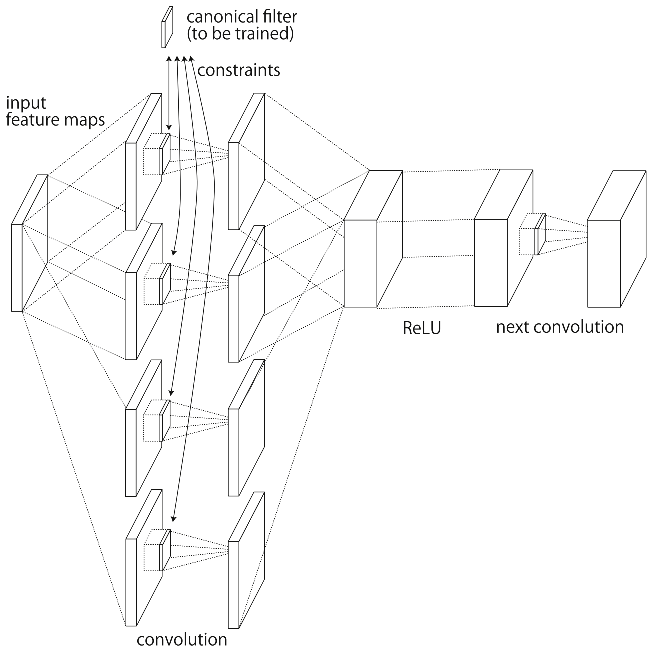


Fig. 4. Coupled convolution layer.

3.2. Constraint matrix

Constraint matrix $\mathbf{H}$ is a simple one-to-one mapping matrix that represents a geometric or parametric constraint between a canonical filter and an applied filter for convolutional operation.

Basic constraint patterns can be rotations, mirroring, the inverse of the sign of a weight (negative constraint), and combinations thereof. To avoid cropping a constrained filter while rotating a squared canonical filter, the rotation angles should be 0, 90, 180 or 270 degrees; thus, the rotations have four patterns. Mirroring has two patterns, i.e., mirrored and not mirrored. Obviously, the inverse of the sign has two patterns, i.e., positive and negative. Thus, there are 16 basic constraint patterns that are represented by $\mathbf{H}$ with elements of $\{-1, 0, 1\}$.

Table 1 lists the 16 constraint matrices, where $\mathbf{I}$ is an identity matrix, $\mathbf{R}$ is a 90-degree clockwise rotation matrix, and $\mathbf{M}$ is a horizontal mirroring matrix. Each $\mathbf{H}$ can be calculated as a product of $\mathbf{I}$, $\mathbf{R}$, and $\mathbf{M}$.

These 16 constraints are considered reasonable especially for lower level feature extraction because local textures has often symmetrical patterns. Fig. 5 shows an example of 16 constrained filters with a canonical filter. The filters have mutually unique weight patterns and are expected to work to extract different features.

Other constraint matrix patterns can be applied, such as random one-to-one mapping; however, they do not perform well (Sections 4 and 5).

Mathematical expressions of constraint matrix for each type are described in Appendix A.

3.3. Weight optimization

For the optimization of weight parameters, general optimization algorithms such as stochastic gradient descent (SGD) and

Table 1
Constraint matrix.

Annotation	Constraint	Constraint matrix $\mathbf{H}$
Org	Identical to canonical filter	$\mathbf{I}$
Rot90	90 degree rotation	$\mathbf{R}$
Rot180	180 degree rotation	$\mathbf{RR}$
Rot270	270 degree rotation	$\mathbf{RRR}$
Mir	Mirroring (horizontal)	$\mathbf{M}$
MirRot90	Mirroring and 90 degree rotation	$\mathbf{RM}$
MirRot180	Mirroring and 180 degree rotation	$\mathbf{RRM}$
MirRot270	Mirroring and 270 degree rotation	$\mathbf{RRRM}$
Neg	Negative of canonical filter	$-\mathbf{I}$
Rot90Neg	90 degree rotation and negative	$-\mathbf{R}$
Rot180Neg	180 degree rotation and negative	$-\mathbf{RR}$
Rot270Neg	270 degree rotation and negative	$-\mathbf{RRR}$
MirNeg	Mirroring and negative	$-\mathbf{M}$
MirRot90Neg	Mirroring, 90 degree rotation and negative	$-\mathbf{RM}$
MirRot180Neg	Mirroring, 180 degree rotation and negative	$-\mathbf{RRM}$
MirRot270Neg	Mirroring, 270 degree rotation and negative	$-\mathbf{RRRM}$

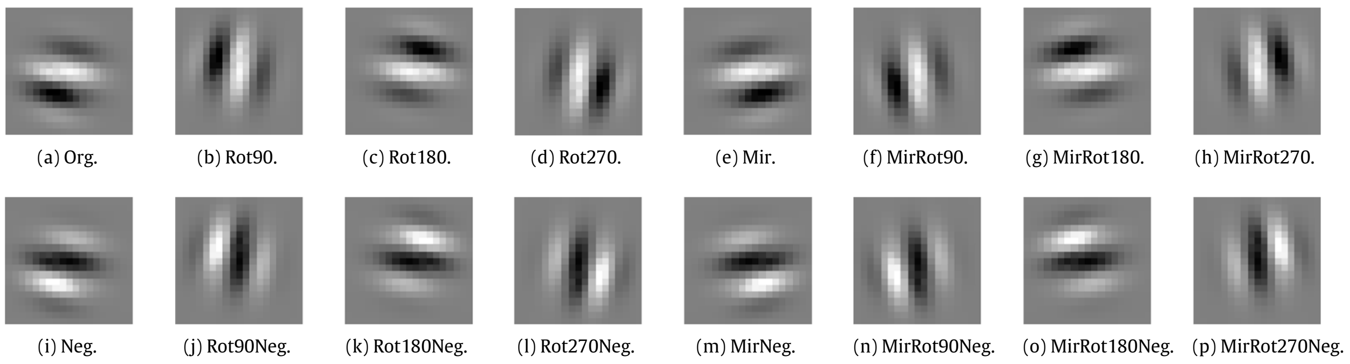


Fig. 5. Example of Constrained Filters: these filters are obtained by applying the 16 constraint matrices to a canonical filter.

Adam (Kingma & Ba, 2014) can be used by calculating the gradient as follows:

$$\nabla E(\mathbf{w}_{(0)}) = \frac{1}{N}\sum_{n=1}^{N} \mathbf{H}^{(n)^{-1}} \nabla E(\mathbf{w}_{(n)}),$$

where $\nabla E(\mathbf{w}_{(n)})$ is the gradient for loss with respect to weight $\mathbf{w}_{(n)}$ in the nth convolution of the network for each mini-batch, $\nabla E(\mathbf{w}_{(0)})$ is the calculated gradient used to update the canonical filter weight $\mathbf{w}_{(0)}$, and $\mathbf{H}^{-1}$ is the inverse of matrix $\mathbf{H}$.

3.4. Computational algorithm

Implementation of coupled convolution layer for inference phase and backpropagation phase can be realized based on the following algorithm.

For inference phase,

1. calculate each filter weight by applying $\mathbf{w}_{(n)} = \mathbf{H}^{(n)}\mathbf{w}_{(0)}$
2. execute convolution with kernel $\mathbf{w}_{(n)}$ and bias $\mathbf{b}_{(n)}$ for each input channels
3. concatenate output channels of each convolution as output feature maps for the layer

For backpropagation phase,

1. with given gradient for each weight $\nabla E(\mathbf{w}_{(n)})$, calculate gradient with respect to the canonical filter as $\mathbf{H}^{(n)^{-1}}\nabla E(\mathbf{w}_{(n)})$
2. Average the each gradient as $\nabla E(\mathbf{w}_{(0)}) = \frac{1}{N}\sum_{n=1}^{N}\mathbf{H}^{(n)-1}$ $\nabla E(\mathbf{w}_{(n)})$
3. Update weights of canonical filter $\mathbf{w}_{(0)}$ using the summed gradient:

   $$\mathbf{w}_{(0)}^{(t+1)} = \mathbf{w}_{(0)}^{(t)} - \eta\nabla E(\mathbf{w}_{(0)}^{(t)}),$$

   where $\mathbf{w}_{(0)}^{(t)}$ is a weight vector of the canonical filter at iteration t, and η is the learning rate.

4. Filter weight analysis

Here, we investigate the trained filters of several networks to observe redundancy between filters under several geometric and parametric relations. In addition, we examine the difference among convolution layers with respect to depth in the network.

This analysis is important to determine what type of constraint should be used and to which layers the given constraint should be applied.

4.1. Analysis method

To find effective constraint weight parameters, we investigate weight parameters trained without constraint to evaluate how likely it is that our constraint candidates will work. If the same pattern as our constraint candidates is observed in the trained weights, the candidate constraint is expected to work properly.

Shang et al. (2016) conducted weight analysis using AlexNet (Krizhevsky et al., 2012) with respect to CReLU, which is equivalent to a negative constraint. We extend their analysis as follows.

Given a unit length vector for each filter $\boldsymbol{\phi}_i$ and a converted filter using constraint matrix $\mathbf{H}\boldsymbol{\phi}_j$, the correlation between these filters is denoted as follows:

$$\rho = \langle \boldsymbol{\phi}_i, \mathbf{H}\boldsymbol{\phi}_j \rangle.$$

If a filter pair $\boldsymbol{\phi}_i$, $\boldsymbol{\phi}_j$ with high correlation exists, the constraint matrix $\mathbf{H}$ is more likely to work because such a filter pair already exists in a well-trained network. On the other hand, if a filter

Table 2
Random constraints.

Annotation	Constraint	Constraint matrix **H**
Rand	Random geometric one-to-one mapping	$\mathbf{Q}$
RandNeg	Random geometric one-to-one mapping and negative	$-\mathbf{Q}$

Table 3
AlexNet: max correlation for each constraint candidate (max correlations over 0.75 are shown in bold).

Relation	conv1	conv2	conv3	conv4	conv5
Rot90	**0.9278**	0.6686	0.4777	0.2812	0.2905
Rot180	**0.9765**	**0.7518**	0.4567	0.4843	0.2499
Rot270	**0.9395**	0.7335	0.4411	0.2641	0.2801
Mir	**0.9689**	**0.7570**	0.5032	0.3152	0.3129
MirRot90	**0.9130**	0.6781	0.4525	0.2806	0.2651
MirRot180	**0.9215**	0.6585	0.4748	0.3018	0.2812
MirRot270	**0.9238**	0.6260	0.4657	0.2803	0.2728
Neg	**0.9172**	0.6665	0.3803	0.6399	0.2324
Rot90Neg	**0.9363**	0.6274	0.3673	0.2022	0.1943
Rot180Neg	**0.8673**	0.6368	0.3829	0.1660	0.2080
Rot270Neg	**0.9203**	0.6388	0.3547	0.2246	0.1953
MirNeg	**0.9451**	0.6295	0.3624	0.2007	0.2107
MirRot90Neg	**0.9121**	0.6570	0.3649	0.1963	0.1872
MirRot180Neg	**0.9472**	0.6517	0.3737	0.1962	0.2255
MirRot270Neg	**0.9116**	0.6056	0.3833	0.1833	0.1964
Rand	0.4753	0.3787	0.4503	0.2302	0.2090
RandNeg	0.4669	0.4765	0.3586	0.1822	0.1956

pair with high correlation does not exist, the constraint matrix is considered unsuitable for that convolution layer.

To simplify the analysis, the maximum correlation $\rho^* = \max\langle\boldsymbol{\phi}_i, \mathbf{H}\boldsymbol{\phi}_j\rangle$ among any filter pair is observed.

For constraint candidates, we apply all constraint matrices listed in Table 1. For comparison, a random constraint matrix is also applied (Table 2), where $\mathbf{Q}$ is a random geometric one-to-one mapping matrix.

We explore convolution layers in AlexNet (Krizhevsky et al., 2012), GoogLeNet (Szegedy et al., 2015), and Network in Network (NIN) (Lin, Chen, & Yan, 2013) with batch normalization (Ioffe & Szegedy, 2015).

4.2. Alexnet

AlexNet was proposed by Krizhevsky et al. (2012) and it comprises five convolution layers.

Table 3 shows the max correlations for each constraint candidate and each layer. High ρ indicates that the constraint is likely to work because the constraint pattern is already formed while the constraint is not posed during training.

As observed, paired filters with high ρ exist in lower layers such as conv1 and conv2 with all geometric constraints. On the other hand, high ρ is not observed in deeper layers, such as conv4 and conv5, for any constraint candidate. These tendencies are also observed with negative constraints (Shang et al., 2016).

Histograms of ρ for each constraint and convolution layer are shown in Fig. B.7.

4.3. GoogLeNet

GoogLeNet is a 22-layer network proposed by Szegedy et al. (2015) at the ImageNet Large-Scale Visual Recognition Challenge 2014. It consists of two convolution layers followed by nine inception modules.

A max correlation analysis is shown in Table 4. Max correlation is high at conv1 for all constraint candidates, with the exception of the Rand and RandNeg constraints. For the other layers, high ρs is detected for some constraints, such as Rot180 and Mir; however, no obvious tendency is observed.

4.4. NIN with batch normalization

NIN, which was proposed by Min et al. (Lin et al., 2013), includes three convolution layers with multilayer perceptrons. Table 5 shows the network structure. Note that batch normalization layers are inserted (Chang & Chen, 2015).

Table 6 shows histograms of ρ for each convolution layer and each constraint. High ρ is observed only in conv1. Note that geometric properties are not preserved after passing through a multilayer perceptron.

4.5. Analytical conclusion

As observed in the three networks, all constraint candidates, except Rand and RandNeg, are likely to work effectively in lower convolution layers, particularly in the first convolution layer.

The results suggest that filters in the first convolution layer are likely to be redundant because there originally exists mirrored and rotated filter pairs along with negative filter pairs. Using coupled convolution layer with mirrored, rotated, and negative constraint, the redundancy can be eliminated by deriving each filter weights from a single set of canonical filter.

Table 4
GoogLeNet: max correlation for each constraint candidate (max correlations over 0.75 are shown in bold).

Relation	conv1	conv2	3(a)	3(b)	4(a)	4(b)	4(c)	4(d)	4(e)	5(a)	5(b)
Rot90	**0.9371**	0.6561	0.7135	0.5348	0.6528	0.6574	0.5370	0.5631	0.6363	0.5147	0.5514
Rot180	**0.9197**	**0.8048**	**0.8420**	0.6080	0.7075	**0.8035**	0.4950	0.4897	0.5821	**0.7844**	0.5016
Rot270	**0.8909**	0.4926	0.6734	0.5042	0.7241	0.6655	0.5157	0.4811	0.6308	0.4941	0.5730
Mir	**0.9575**	0.6563	0.7319	0.5187	**0.7754**	**0.7539**	0.7372	0.7333	**0.8863**	**0.9348**	0.6300
MirRot90	**0.9055**	**0.7901**	0.6453	0.4372	0.7007	0.7300	0.5089	0.4410	0.5124	0.5047	0.5700
MirRot180	**0.8297**	**0.7510**	0.7003	0.5557	0.5758	**0.7549**	0.5514	0.5612	0.6004	0.5055	0.5089
MirRot270	**0.9381**	0.7040	**0.8160**	0.7014	0.6772	0.7258	0.5600	0.5799	0.6526	0.5069	0.5561
Neg	**0.9030**	0.7452	**0.7677**	0.6500	0.6031	0.5696	0.5043	0.4161	0.3578	0.6461	0.3011
Rot90Neg	**0.8833**	0.6681	0.6489	0.3656	0.6537	0.5253	0.3558	0.2769	0.2927	0.3523	0.2885
Rot180Neg	**0.8093**	0.6450	0.5350	0.3486	0.6349	0.5337	0.3089	0.2871	0.3718	0.4112	0.3027
Rot270Neg	**0.9186**	0.6481	0.7094	0.4139	0.6381	0.5027	0.3568	0.2237	0.3599	0.3626	0.2693
MirNeg	**0.8854**	0.6660	0.6641	0.4403	0.6176	0.5008	0.3850	0.2852	0.3065	0.3765	0.2998
MirRot90Neg	**0.9284**	0.6630	0.6598	0.2705	0.6350	0.5229	0.3910	0.2843	0.3622	0.3513	0.2689
MirRot180Neg	**0.9238**	0.6656	0.6785	0.3348	0.6201	0.5602	0.4757	0.3624	0.3729	0.4739	0.3015
MirRot270Neg	**0.9013**	0.6496	0.6883	0.2629	0.6345	0.5410	0.3513	0.2147	0.2988	0.3656	0.2918
Rand	0.4142	0.3701	0.2356	0.2676	0.4021	0.2897	0.2275	0.1771	0.2767	0.3307	0.5150
RandNeg	0.4095	0.6716	0.2368	0.2055	0.1957	0.2710	0.2003	0.2210	0.2493	0.3431	0.2711

Table 5
Structure of Network in Network (NIN) (Lin et al., 2013).

Layer	Patch, stride	Input maps	Output	Activation function pooling/dropout
data	–	–	$32 \times 32 \times 3$	–
conv1	5×5	3	$32 \times 32 \times N$	ReLU
cccp1	1×1	N	$32 \times 32 \times 160$	ReLU
cccp2	1×1	160	$32 \times 32 \times 96$	ReLU
pool1	3×3, /2	96	$16 \times 16 \times 96$	maxout
drop3	–	96	$16 \times 16 \times 96$	Dropout ($p = 0.5$)
conv2	5×5	96	$16 \times 16 \times 192$	ReLU
cccp3	1×1	192	$16 \times 16 \times 192$	ReLU
cccp4	1×1	192	$16 \times 16 \times 192$	ReLU
pool2	3×3, /2	192	$8 \times 8 \times 192$	average
drop6	–	192	$8 \times 8 \times 192$	Dropout ($p = 0.5$)
conv3	3×3	192	$8 \times 8 \times 192$	ReLU
cccp5	1×1	192	$8 \times 8 \times 192$	ReLU
cccp6	1×1	192	$8 \times 8 \times P$	ReLU
pool3	8×8	P	$1 \times 1 \times P$	average

Table 6
NIN: max correlation for each constraint candidate (max correlations over 0.75 are shown in bold).

Relation	conv1	conv2	conv3
Rot90	**0.7716**	0.2305	0.2329
Rot180	0.7348	0.2709	0.2444
Rot270	**0.7844**	0.3772	0.2208
Mir	**0.7783**	0.3114	0.3182
MirRot90	**0.7808**	0.2350	0.2011
MirRot180	0.7425	0.2716	0.2275
MirRot270	**0.8275**	0.3777	0.2258
Neg	**0.8091**	0.2904	0.2543
Rot90Neg	**0.7734**	0.2251	0.2020
Rot180Neg	0.7360	0.2071	0.2102
Rot270Neg	**0.8259**	0.1985	0.1919
MirNeg	**0.7992**	0.2257	0.2360
MirRot90Neg	**0.8206**	0.2244	0.2049
MirRot180Neg	**0.7559**	0.2357	0.2542
MirRot270Neg	**0.7712**	0.2143	0.2104
Rand	0.6341	0.1612	0.2618
RandNeg	0.7173	0.1660	0.2296

We have also observed that no highly correlated filter pairs with Rand and RandNeg relation exist in any convolution layer. In addition, mirrored, rotated, and negative related filter pairs does not exist in deeper convolution layer. This indicates that mirrored, rotated, and negative related filter pairs are unique property of filters for natural images. In other words, it implies that natural images are isotropic, however, feature maps in deeper network are getting more anisotropic.

5. Experiments

We evaluate the effect of coupled convolution layers using the CIFAR-10, CIFAR-100, and Plankton1.0 datasets on classification tasks.

In the experiments, the network structure is based on NIN with batch normalization, as shown in Table 5. Parameters N and P for NIN differ depending on the applied constraints and datasets. As discussed in the previous section, NIN has high ρ primarily in the first convolution layer (conv1); thus, a coupled convolution layer is only applied to conv1.

We used three sets of model[1] and parameter settings, as shown in Tables 7–9. The first set has the same number of independent filters (canonical filters), i.e., the number of parameters does not increase while the total number of filters and output channels increases proportionally relative to the number of constraints. Eighteen filters were allocated for each constraint. The number of canonical filters was determined empirically such that the network could perform as a classification system while avoiding overfitting. The second set has the same number of filters in total, i.e., the number of independent filters depends on the number of constraints. In the experiments, 288 filters were allocated in total. The third set has the same constraint matrix, i.e., positive and negative constraints, and the different numbers of independent filters.

SGD was used to optimize the network. In each trial, 15,000 iterations (mini-batch size of 128) were executed, and the lowest error rate was taken as the result. The learning rate was 0.1 until iteration 10,000 and 0.01 afterward. The momentum is set to 0.9 and weight decay is 0.0001. Cross-entropy loss function is adapted as the cost of network to be minimized. We ran five trials for each model and calculated the average and standard deviation of the test error rate. As of the cost of the network to be minimized, cross-entropy loss function is used.

All experiments were implemented using the Caffe framework (Jia et al., 2014) and executed on Nvidia TitanX GPUs. The source code is available at http://www.ok.sc.e.titech.ac.jp/res/DL/CCL/.

5.1. CIFAR-10

The CIFAR-10 dataset consists of 50,000 training images and 10,000 validation images (32×32 pixels, RGB format). Each image is labeled with one of ten classes that represents the main object in the image.

The image set is preprocessed with global contrast normalization (GCN) and zero-phase component analysis (ZCA) whitening for training and validation in the same manner as Goodfellow, Warde-farley, Mirza, Courville, and Bengio (2013).

Table 10 shows the result for the model set with 18 independent filters. The models with constraints, i.e., Neg36 and MirRotNeg288, outperform the original model when the number of parameters in conv1 is equal.

The results for the model set with 288 filters is shown in Table 11. The Neg288 model with a negative constraint performs better despite the fact that conv1 has half the number of parameters. MirRotNeg288 performs comparable to the non-constrained model None288 even though the number of parameters in conv1 is limited by only one-sixteenth.

The result on the effect of the number of independent filters is shown in Table 12. It is observed that models having more independent filters perform better.

Computational cost in inference phase is same as the original network if $\mathbf{w}_{(n)}$ is calculated from the canonical filter $\mathbf{w}_{(0)}$ in advance. However, the cost in back-propagation phase is increased by 5.4% compared to the original network because computation with constraint matrix costs additionally in our implementation.

5.2. CIFAR-100

The CIFAR-100 dataset (Krizhevsky & Hinton, 2009) is the same as CIFAR-10 but has 100 labels ($P = 100$), whereas the number of images is equal to that of CIFAR-10. Consequently, only one-tenth of the images for each class are available for training.

We compared the same networks used in the CIFAR-10 experiment for CIFAR-100 except for parameter P. We also preprocessed images using GCN and ZCA whiting, as we did for CIFAR-10.

Table 13 shows the results for the model set with 18 independent filters. The MirRotNeg288 model performs better among models with the same number of parameters in conv1.

The results for the model set with 288 filters is shown in Table 14. The non-constraint model None288 performs better;

[1] Note that Neg192 and Neg384 are equivalent models of biased ON/OFF ReLU proposed in Uchida et al. (2016) denoted as BiasOnOff192 and BiasOnOff384.

Table 7
Comparing models with 18 independent filters.

Model	Constraint matrix	N	# of independent filters (N/# of constraints)	# of parameters of conv1 (w/o BN)
None18	**I**	18	18	1,368
Neg36	**I**, −**I**	36	18	1,368
MirRotNeg288	**I**, **R**, **RR**, **RRR**, **M**, **RM**, **RRM**, **RRRM**, −**I**, −**R**, −**RR**, −**RRR**, −**M**, −**RM**, −**RRM**, −**RRRM**	288	18	1,368
Rand36	**I**, **Q**	36	18	1,368

Table 8
Comparing models with 288 filters.

Model	Constraint matrix	N	# of independent filters (N/# of constraints)	# of parameters of conv1 (w/o BN)
None288	**I**	288	288	21,888
Neg288	**I**, −**I**	288	144	10,944
MirRotNeg288	**I**, **R**, **RR**, **RRR**, **M**, **RM**, **RRM**, **RRRM**, −**I**, −**R**, −**RR**, −**RRR**, −**M**, −**RM**, −**RRM**, −**RRRM**	288	18	1,368
Rand288	**I**, **Q**	288	144	10,944

Table 9
Comparing models with the different numbers of filters.

Model	Constraint matrix	N	# of independent filters (N/# of constraints)	# of parameters of conv1 (w/o BN)
Neg192	**I**, −**I**	192	96	7,296
Neg288	**I**, −**I**	288	144	10,944
Neg384	**I**, −**I**	384	192	14,592

Table 10
Results for CIFAR-10 (18 filters for each constraint).

Model	Error	Training loss
None18	10.36% ± 0.30%	0.0093 ± 0.0029
Neg36	9.90% ± 0.10%	0.0141 ± 0.0076
MirRotNeg288	**9.61% ± 0.17%**	0.0033 ± 0.0012
Rand36	15.82% ± 6.85%	0.0680 ± 0.0826

Table 11
Result for CIFAR-10 (288 filters for the convolution layer).

Model	Error	Training loss
None288	9.61% ± 0.15%	0.0063 ± 0.0046
Neg288	**9.36% ± 0.06%**	0.0176 ± 0.0196
MirRotNeg288	9.61% ± 0.17%	0.0033 ± 0.0012
Rand288	15.21% ± 6.80%	0.0512 ± 0.0466

however, the MirRotNeg288 model is also competitive considering that it has significantly fewer parameters in conv1.

Table 15 shows the effect of the number of independent filters. The Neg384 model which has more independent filters performs better.

5.3. *Plankton1.0*

The PlanktonSet 1.0 (Cowen, Sponaugle, Robinson, & Luo, 2015) dataset comprises 30,336 grayscale labeled images of various sizes of planktons for training and 130,400 images for testing. The number of classes to be categorized is 121 ($P = 121$). Because the dataset was used in a competition (Kaggle, 2015), the ground truth of the test data is not available to the public. Therefore, we divided the training dataset into 25,000 images for training and 5,336 images for validation (Xu, Wang, Chen, & Li, 2015).

Table 12
Result for CIFAR-10 (effect on the number of filters for the convolution layer).

Model	Error	Training loss
Neg192	9.53% ± 0.08%	0.0135 ± 0.0086
Neg288	9.36% ± 0.06%	0.0176 ± 0.0196
Neg384	**9.02% ± 0.06%**	0.0034 ± 0.0014

Table 13
Result for CIFAR-100 (18 filters for each constraint).

Model	Error	Training loss
None18	35.21% ± 0.23%	0.1386 ± 0.0400
Neg36	34.81% ± 0.39%	0.1378 ± 0.0227
MirRotNeg288	**34.25% ± 0.15%**	0.1163 ± 0.0220
Rand36	41.93% ± 6.54%	0.2728 ± 0.0654

Table 14
Result for CIFAR-100 (288 filters).

Model	Error	Training loss
None288	**34.16% ± 0.21%**	0.0880 ± 0.0229
Neg288	34.39% ± 0.27%	0.0848 ± 0.0289
MirRotNeg288	34.25% ± 0.15%	0.1163 ± 0.0220
Rand288	40.61% ± 6.64%	0.2923 ± 0.1567

Table 15
Result for CIFAR-100 (effect on the number of filters).

Model	Error	Training loss
Neg192	$34.41\% \pm 0.33\%$	0.2059 ± 0.0340
Neg288	$34.39\% \pm 0.27\%$	0.0848 ± 0.0289
Neg384	$\mathbf{33.02\% \pm 0.33\%}$	0.0443 ± 0.0108

Table 16
Result for Plankton1.0 (18 filters for each constraint).

Model	Error	Training loss
None18	$30.70\% \pm 0.26\%$	0.4743 ± 0.0670
Neg36	$30.43\% \pm 0.22\%$	0.4758 ± 0.0931
MirRotNeg288	$\mathbf{30.24\% \pm 0.10\%}$	0.4498 ± 0.0531
Rand36	$30.94\% \pm 0.26\%$	0.5055 ± 0.0669

Table 17
Result for Plankton1.0 (288 filters).

Model	Error	Train loss
None288	$30.26\% \pm 0.21\%$	0.4224 ± 0.1105
Neg288	$30.64\% \pm 0.26\%$	0.3769 ± 0.0904
MirRotNeg288	$\mathbf{30.24\% \pm 0.10\%}$	0.4498 ± 0.0531
Rand288	$30.36\% \pm 0.27\%$	0.4428 ± 0.0586

Table 18
Result for Plankton1.0 (effect on the number of filters).

Model	Error	Train loss
Neg192	$30.67\% \pm 0.24\%$	0.4587 ± 0.0406
Neg288	$30.64\% \pm 0.26\%$	0.3769 ± 0.0904
Neg384	$\mathbf{30.06\% \pm 0.32\%}$	0.4149 ± 0.0506

For data augmentation, we randomly rotated, translated and scaled the images, and then resized them to 32×32 pixels. Consequently, 250,000 images were available for training.

We trained the same models as those of CIFAR-10 and CIFAR-100.

Table 16 shows the results for the model set with 18 independent filters. The MirRotNeg288 model performs better among models with the same number of parameters in conv1.

The results for the model set with 288 filters are shown in Table 17. The MirRotNeg288 model performs better despite the small number of parameters in conv1.

Table 18 shows the results for the effect on the number of independent filters. The model with more independent filters, i.e., Neg384, performs better.

6. Conclusion

We have proposed a coupled convolution layer that comprises pairs of filters that are mutually constrained.

We evaluated the coupled convolution layer with geometric constraints and negative weight constraints in a comparison with non-constrained convolution layers. Our analysis revealed that filter weights in trained neural networks, such as AlexNet and GoogLeNet, are formed with geometrically and parametrically symmetric patterns. This property supports the introduction of the constraints.

The experimental results show that a network with a coupled convolution layer outperforms non-constrained networks overall, thereby indicating that proper constraint of filter weights improves network performance.

Moreover, constraining filter weights reduces the number of parameters. This is beneficial because the number of parameters in the convolution layer can be reduced while improving performance.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neunet.2018.05.002.

References

Bao, G., & Zeng, Z. (2016). Global asymptotical stability analysis for a kind of discrete-time recurrent neural network with discontinuous activation functions. *Neurocomputing*, *193*(Suppl. C), 242–249. http://dx.doi.org/10.1016/j.neucom.2016.02.017.

Bao, G., & Zeng, Z. (2017). Region stability analysis for switched discrete-time recurrent neural network with multiple equilibria. *Neurocomputing*, *249*(Suppl. C), 182–190. http://dx.doi.org/10.1016/j.neucom.2017.03.065.

Chang, J.-R., & Chen, Y.-S. (2015). Batch-normalized Maxout Network in Network, arXiv preprint arXiv:1511.02583.

Chen, Z., & Cao, F. (2009). The approximation operators with sigmoidal functions. *Computers and Mathematics with Applications*, *58*(4), 758–765.

Chen, Z., & Cao, F. (2016). Scattered data approximation by neural networks operators. *Neurocomputing*, *190*, 237–242.

Costarelli, D., Spigler, R., et al. (2013). Solving volterra integral equations of the second kind by sigmoidal functions approximation. *Journal of Integral Equations and Applications*, *25*(2), 193–222.

Costarelli, D., Spigler, R., et al. (2014). A collocation method for solving nonlinear Volterra integro-differential equations of neutral type by sigmoidal functions. *Journal of Integral Equations and Applications*, *26*(1), 15–52.

Costarelli, D., & Vinti, G. (2016). Max-product neural network and quasi-interpolation operators activated by sigmoidal functions. *Journal of Approximation Theory*, *209*, 1–22.

Cowen, R. K., Sponaugle, S., Robinson, K., & Luo, J. (2015). Planktonset 1.0: plankton imagery data collected from f.g. walton smith in straits of florida from 2014-06-03 to 2014-06-06 and used in the 2015 national data science bowl (nodc accession 0127422). noaa national centers for environmental information. dataset. http://dx.doi.org/10.7289/v5d21vjd, Oregon State University and Hatfield Marine Science Center (2015).

Dong, C., Loy, C. C., He, K., & Tang, X. (2014). Learning a deep convolutional network for image super-resolution. In *European conference on computer vision* (pp. 184–199). Springer.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).

Goodfellow, I., Warde-farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. In *Proceedings of the 30th international conference on machine learning* (pp. 1319–1327).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*.

Hong, B. I., & Hahm, N. (2016). A note on neural network approximation with a sigmoidal function. *Applied Mathematical Sciences*, *10*(42), 2075–2085.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. *Computing Research Repository*, *abs/1502.03167*. http://arxiv.org/abs/1502.03167.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., & Girshick, R., et al. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding, arXiv preprint arXiv:1408.5093.

Jones, J. P., & Palmer, L. A. (1987). An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, *58*(6), 1233–1258.

Kaggle, (2015). National data science bowl competition. URL https://www.kaggle.com/c/datasciencebowl.

Kim, J., Kim, S., & Lee, M. (2015). Convolutional neural network with biologically inspired on/off relu. In *Neural information processing* (pp. 316–323). Springer.

Kim, J., Lee, J. K., & Lee, K. M. (2015a). Accurate image super-resolution using very deep convolutional networks, arXiv preprint arXiv:1511.04587.

Kim, J., Lee, J. K., & Lee, K. M. (2015b). Deeply-recursive convolutional network for image super-resolution, arXiv preprint arXiv:1511.04491.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *CoRR*, *abs/1412.6980*. http://arxiv.org/abs/1412.6980.

Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Cehovin, L., & Fernandez, G., et al. (2015). The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE international conference on computer vision workshops* (pp. 1–230.

Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images* (Master's thesis), Department of Computer Science, University of Toronto.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *Computing Research Repository (CoRR)*, *abs/1312.4400*. http://arxiv.org/abs/1312.4400.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Marcos, D., Volpi, M., & Tuia, D. (2016). Learning rotation invariant convolutional filters for texture classification. In *Proceedings of the 23rd international conference on pattern recognition* (pp. 2013–2018).

Olshausen, B. A., et al. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*(6583), 607–609.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., & Poggio, T. (2007). Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*, 411–426.

Shang, W., Sohn, K., Almeida, D., & Lee, H. (2016). Understanding and improving convolutional neural networks via concatenated rectified linear units, arXiv preprint arXiv:1603.05201.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., & Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Uchida, K., Tanaka, M., & Okutomi, M. (2016). Coupled convolution layer for convolutional neural network. In *Proceedings of the 23rd international conference on pattern recognition* (pp. 3537–3542).

Wandell, B. A. (1995). *Foundations of vision*. Stanford University.

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network, arXiv preprint arXiv:1505.00853.