

# Doubly Convolutional Neural Networks

CNN RESEARCH PROJECT

...

MNNIT-ALLAHABAD

ARYAN KARN (20185106)

# AIM

Parameter sharing is the major reason of success of building large models of CNN for deep neural networks. This paper introduces the idea of **Doubly Convolutional Neural Networks**, which significantly improves the performance of CNN with the same number of parameters and improving the speed.

# RESEARCH PROPOSED BY:

Shuangfei Zhai : Binghamton University

Yu Cheng : IBM T.J. Watson Research Center

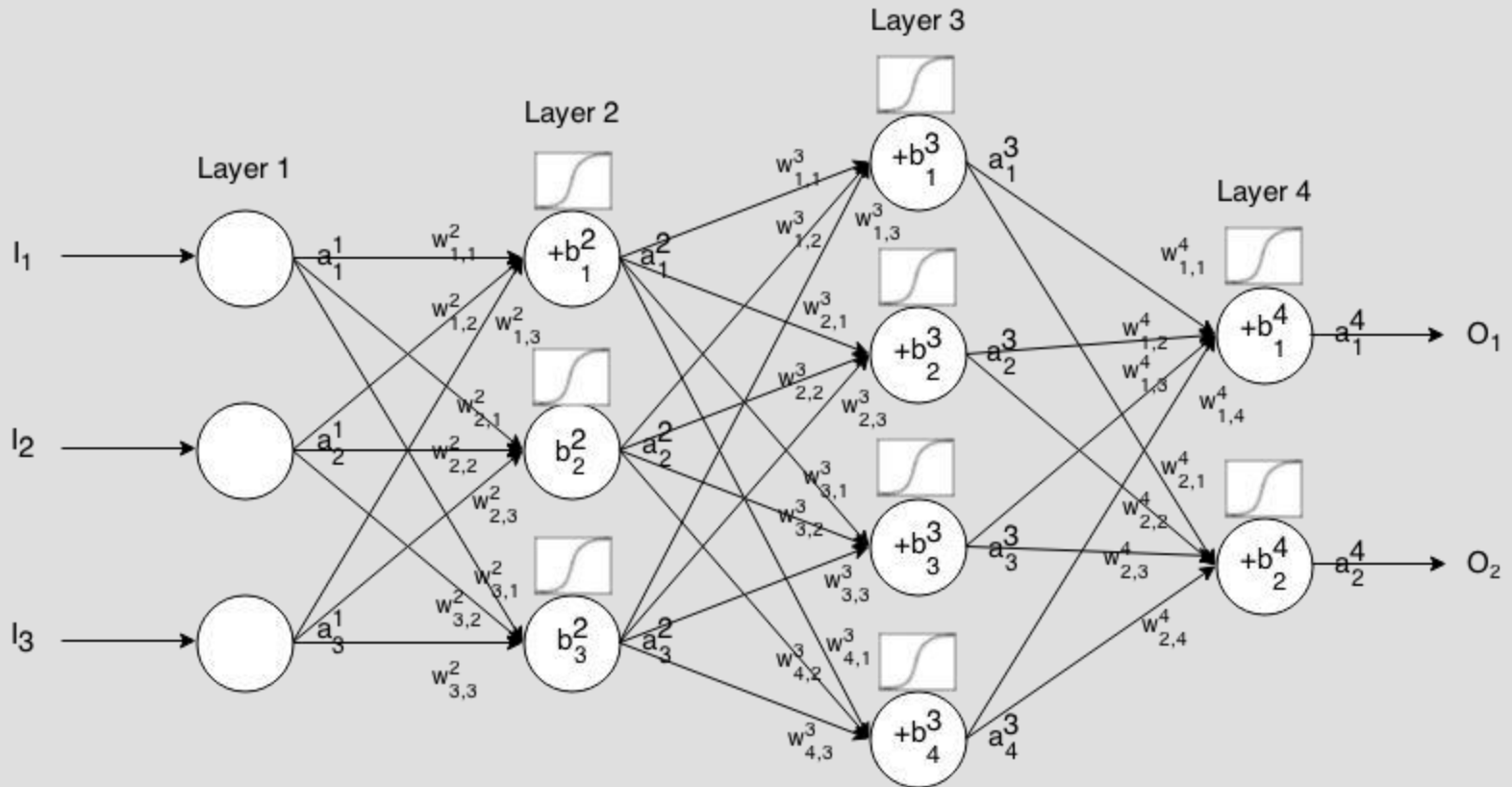
Weining Lu : Tsinghua University

Zhongfei (Mark) : Zhang Binghamton University

## ABSTRACT:

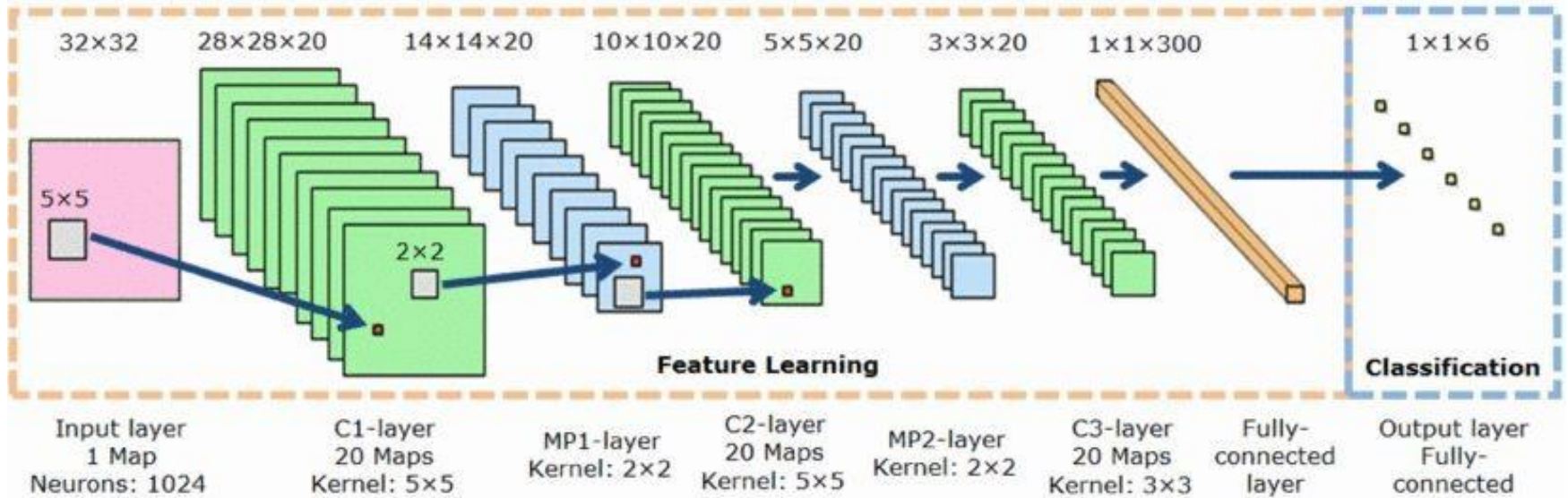
Building large models with parameter sharing accounts for most of the success of deep convolutional neural networks (CNNs). In this paper, we propose doubly convolutional neural networks (DCNNs), which significantly improve the performance of CNNs by further exploring this idea. Instead of allocating a set of convolutional filters that are independently learned, a DCNN maintains groups of filters where filters within each group are translated versions of each other. Practically, a DCNN can be easily implemented by a two-step convolution procedure, which is supported by most modern deep learning libraries.

# Neural Network



# Convolutional Neural network

CNNs are extremely parameter efficient due to exploring the translation invariant property of images, which is the key to training very deep models without severe overfitting.



# K-Translation Correlation

In well trained CNNs, many of the learned filters are slightly translated versions of each other.

K-translation correlation between two convolutional filters within same layer  $\mathbf{W}_i, \mathbf{W}_j$  is defined as:

$$\rho_k(\mathbf{W}_i, \mathbf{W}_j) = \max_{x,y \in \{-k, \dots, k\}, (x,y) \neq (0,0)} \frac{\langle \mathbf{W}_i, T(\mathbf{W}_j, x, y) \rangle_f}{\|\mathbf{W}_i\|_2 \|\mathbf{W}_j\|_2}$$

*Here,  $T(\cdot, x, y)$  denotes the translation of the first operand by  $(x, y)$  along its spatial dimensions.*

K-translation correlation between a pair of filters indicates the maximum correlation achieved by translating filters up to  $k$  steps along any spatial dimension.

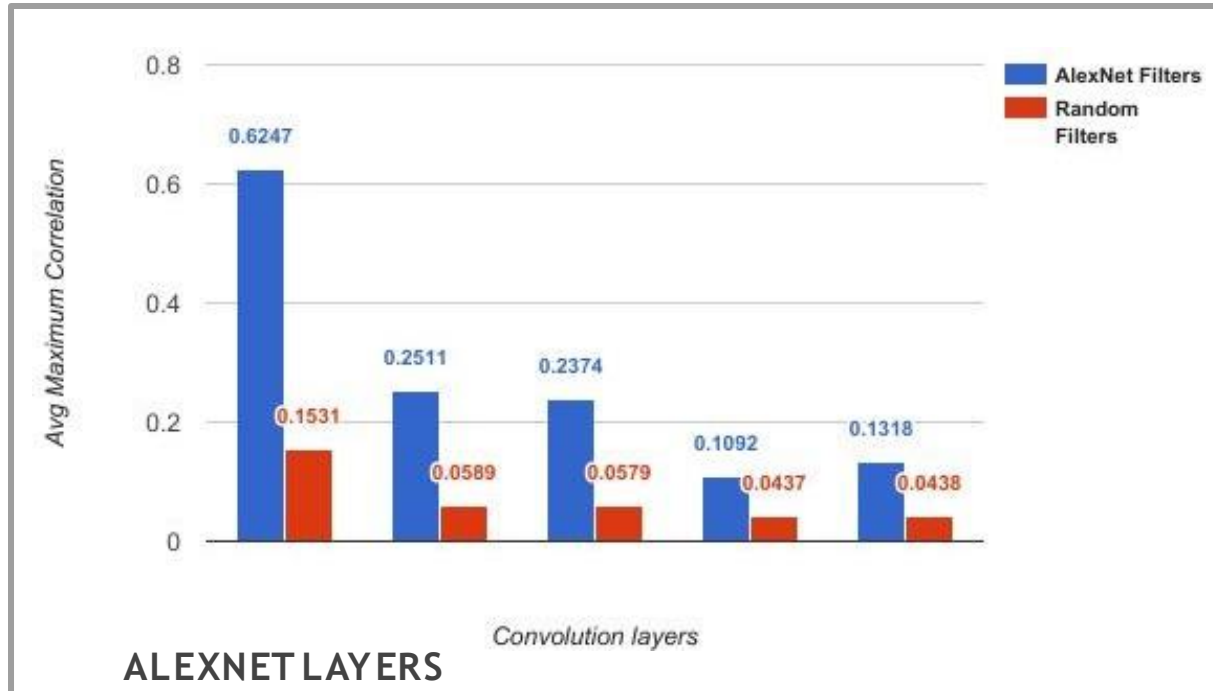
For deeper models, averaged maximum  $k$ -translation correlation of a layer  $\mathbf{W}$  is:

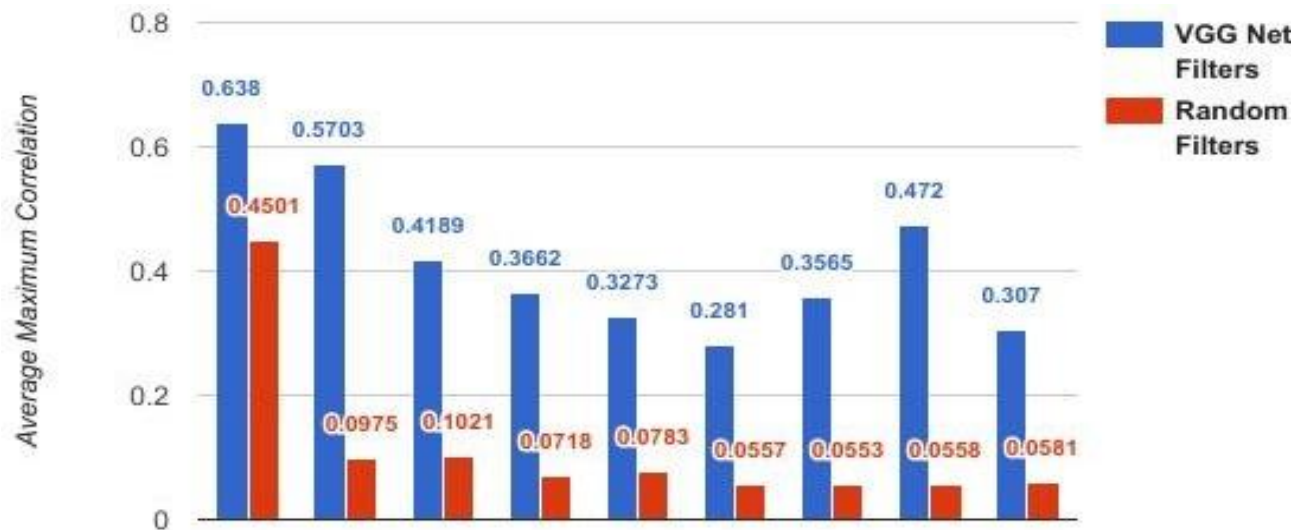
$$\bar{\rho}_k(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^N \max_{j=1, j \neq i}^N \rho_k(\mathbf{W}_i, \mathbf{W}_j)$$

*$N$  is the number of filters*

# Correlation Results

The averaged maximum 1-translational correlation of each layer for AlexNet and VGG Net are as follows. As a comparison, a filter bank with same shape filled with random gaussian samples has been generated.





VGG-19 first nine layers

Convolution Layers



# Idea of DCNN

Group filters which are translated versions of each other.

DCNN allocates a set of meta filters

Convolve meta filters with identity kernel

Effective filters extracted

# Convolution

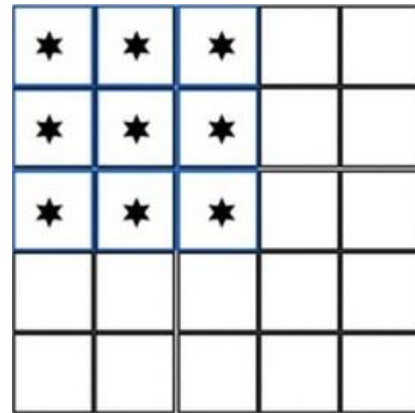
$$\mathcal{I}_{k,i,j}^{\ell+1} = \sum_{c' \in [1,c], i' \in [1,z], j' \in [1,z]} \mathbf{W}_{k,c',i',j'}^{\ell} \mathcal{I}_{c',i+i'-1,j+j'-1}^{\ell}$$

$$k \in [1, c^{\ell+1}], i \in [1, w^{\ell+1}], j \in [1, h^{\ell+1}].$$

Input image:  $\mathcal{I}^{\ell} \in R^{c^{\ell} \times w^{\ell} \times h^{\ell}}$

Set of  $c_{l+1}$  filters:  $\mathbf{W}^{\ell} \in R^{c^{\ell+1} \times c^{\ell} \times z \times z}$  each filter of shape:  $c' \times z \times z$

Output image:  $\mathcal{I}^{\ell+1} \in R^{c^{\ell+1} \times w^{\ell+1} \times h^{\ell+1}}$



# Double Convolution

$$\mathcal{O}_{i,j,k}^{\ell+1} = \mathbf{W}_k^\ell * \mathcal{I}_{:,i:(i+z-1),j:(j+z-1)}^\ell,$$

$$\mathcal{I}_{(nk+1):n(k+1),i,j}^{\ell+1} = \text{pool}_s(\mathcal{O}_{i,j,k}^{\ell+1}), n = \left(\frac{z' - z + 1}{s}\right)^2$$

$$k \in [1, c^{\ell+1}], i \in [1, w^{\ell+1}], j \in [1, h^{\ell+1}].$$

Input image:  $\mathcal{I}^\ell \in R^{c^\ell \times w^\ell \times h^\ell}$

Output image:  $\mathcal{I}^{\ell+1} \in R^{nc^{\ell+1} \times w^{\ell+1} \times h^{\ell+1}}$

Set of  $c^{\ell+1}$  meta filters:  $\mathbf{W}^\ell \in R^{c^{\ell+1} \times c^\ell \times z' \times z'}$  with filter size  $z \times z$ ,  $z' > z$

Spatial pooling function with pooling size  $s \times s$

## Working of DCNN

Set of  $c^{l+1}$  meta filters  
size  $(z' \times z')$

Image patches size  $(z \times z)$   
convolved with each meta filter

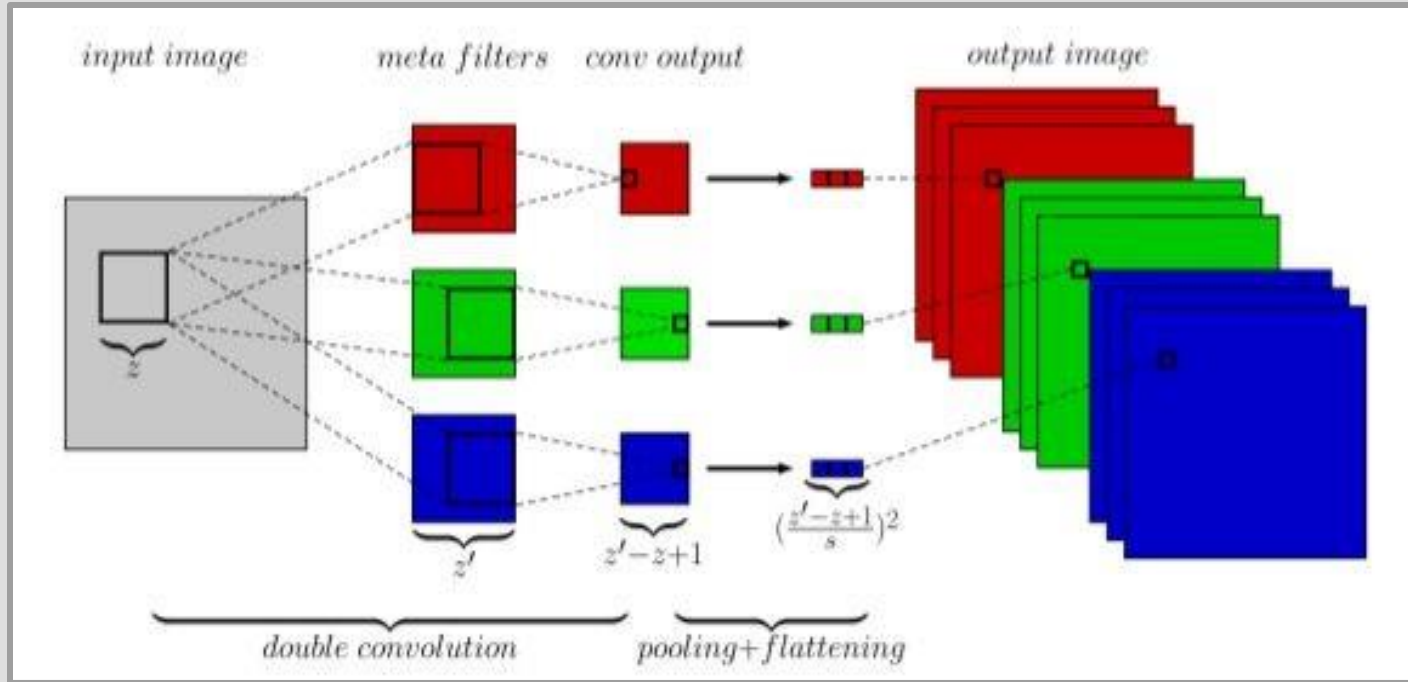
Output size  $(z'-z+1) \times (z'-z+1)$

Spatial pooling with size  $(s \times s)$

Output flattened to column vector

Feature map with  
 $nc^{l+1}$  channels

# Double Convolution: 2 step convolution



**STEP1:** An image patch is convolved with a metafilter.

**STEP2:** Meta filters slide across to get different patches, i.e. convolved with the image.

**ALGORITHM-**

---

**Algorithm 1** Implementation of double convolution with convolution.

---

**Input** : Input image  $I^l \in R^{c^l * w^l * h^l}$ , meta filters  $W^l \in R^{c^{l+1} * c^l * z' * z'}$ ,  
effective filter size  $z * z$ , pooling size  $s * s$ .

**Output:** Output image  $I^{l+1} \in R^{nc^{l+1} * w^{l+1} * h^{l+1}}$ , with  $n = \frac{(z'-z+1)^2}{s^2}$ .

**function** DOUBLE CONVOLUTION



1.  $I^l \leftarrow IdentityMatrix(c^l z^2)$ ;
2. Reorganize  $I^l$  to shape  $c^l z^2 * c^l * z * z$ ;
3.  $\tilde{W}^l \leftarrow W^l * I^l$ ; /\* output shape:  $c^{l+1} * c^l z^2 * (z' - z + 1) * (z' - z + 1) *$  /
4. Reorganize  $\tilde{W}^l$  to shape  $c^{l+1} (z' - z + 1)^2 * c^l * z * z$ ;
5.  $O^{l+1} \leftarrow I^l * \tilde{W}^l$ ; /\* output shape:  $c^{l+1} (z' - z + 1)^2 * w^{l+1} * h^{l+1} *$  /
6. Reorganize  $O^{l+1}$  to shape  $c^{l+1} w^{l+1} h^{l+1} * (z' - z + 1) * (z' - z + 1)$ ;
7.  $I^{l+1} \leftarrow pool_s(O^{l+1})$ ; /\* output shape:  $c^{l+1} w^{l+1} h^{l+1} * \frac{z'-z+1}{s} * \frac{z'-z+1}{s} *$  /
8. Reorganize  $I^{l+1}$  to shape  $c^{l+1} (\frac{z'-z+1}{s})^2 * w^{l+1} * h^{l+1}$ ;

**end function**

---

1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1

**Identity (8x8)**



---

**Algorithm 1** Implementation of double convolution with convolution.

---

**Input** : Input image  $I^l \in R^{c^l * w^l * h^l}$ , meta filters  $W^l \in R^{c^{l+1} * c^l * z' * z'}$ ,  
effective filter size  $z * z$ , pooling size  $s * s$ .

**Output:** Output image  $I^{l+1} \in R^{nc^{l+1} * w^{l+1} * h^{l+1}}$ , with  $n = \frac{(z'-z+1)^2}{s^2}$ .

**function** DOUBLE CONVOLUTION

1.  $I^l \leftarrow IdentityMatrix(c^l z^2)$ ;
2. Reorganize  $I^l$  to shape  $c^l z^2 * c^l * z * z$ ;
3.  $\tilde{W}^l \leftarrow W^l * I^l$ ; /\* output shape:  $c^{l+1} * c^l z^2 * (z' - z + 1) * (z' - z + 1) *$  /
4. Reorganize  $\tilde{W}^l$  to shape  $c^{l+1} (z' - z + 1)^2 * c^l * z * z$ ;
5.  $O^{l+1} \leftarrow I^l * \tilde{W}^l$ ; /\* output shape:  $c^{l+1} (z' - z + 1)^2 * w^{l+1} * h^{l+1} *$  /
6. Reorganize  $O^{l+1}$  to shape  $c^{l+1} w^{l+1} h^{l+1} * (z' - z + 1) * (z' - z + 1)$ ;
7.  $I^{l+1} \leftarrow pool_s(O^{l+1})$ ; /\* output shape:  $c^{l+1} w^{l+1} h^{l+1} * \frac{z'-z+1}{s} * \frac{z'-z+1}{s} *$  /
8. Reorganize  $I^{l+1}$  to shape  $c^{l+1} (\frac{z'-z+1}{s})^2 * w^{l+1} * h^{l+1}$ ;

**end function**

---

Applying All 8 Masks for fix position

	4	3	2
	7	1	5
	6	9	8
7	8	9	
2	6	3	
1	4	5	

Meta Filter (2x3x3)  
Filter (2x2x2) (Blue)

\*

0	0
0	0
1	0
0	0

Rearranged 1<sup>st</sup> Row

7

---

**Algorithm 1** Implementation of double convolution with convolution.

---

**Input** : Input image  $I^l \in R^{c^l * w^l * h^l}$ , meta filters  $W^l \in R^{c^{l+1} * c^l * z' * z'}$ ,  
effective filter size  $z * z$ , pooling size  $s * s$ .

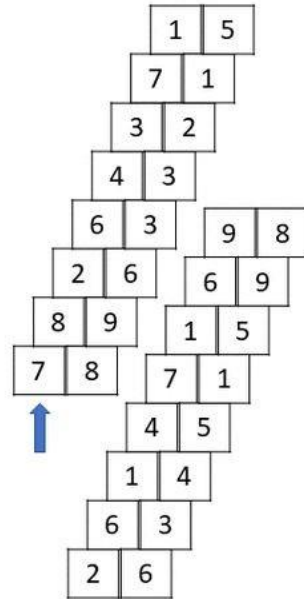
**Output:** Output image  $I^{l+1} \in R^{nc^{l+1} * w^{l+1} * h^{l+1}}$ , with  $n = \frac{(z'-z+1)^2}{s^2}$ .

**function** DOUBLE CONVOLUTION

1.  $I^l \leftarrow IdentityMatrix(c^l z^2)$ ;
2. Reorganize  $I^l$  to shape  $c^l z^2 * c^l * z * z$ ;
3.  $\tilde{W}^l \leftarrow W^l * I^l$ ; /\* output shape:  $c^{l+1} * c^l z^2 * (z' - z + 1) * (z' - z + 1) *$  /
4. Reorganize  $\tilde{W}^l$  to shape  $c^{l+1}(z' - z + 1)^2 * c^l * z * z$ ;
5.  $O^{l+1} \leftarrow I^l * \tilde{W}^l$ ; /\* output shape:  $c^{l+1}(z' - z + 1)^2 * w^{l+1} * h^{l+1} *$  /
6. Reorganize  $O^{l+1}$  to shape  $c^{l+1} w^{l+1} h^{l+1} * (z' - z + 1) * (z' - z + 1)$ ;
7.  $I^{l+1} \leftarrow pool_s(O^{l+1})$ ; /\* output shape:  $c^{l+1} w^{l+1} h^{l+1} * \frac{z'-z+1}{s} * \frac{z'-z+1}{s} *$  /
8. Reorganize  $I^{l+1}$  to shape  $c^{l+1}(\frac{z'-z+1}{s})^2 * w^{l+1} * h^{l+1}$ ;

**end function**

---



Rearranging

4	3
7	1
7	8
2	6

---

**Algorithm 1** Implementation of double convolution with convolution.

---

**Input** : Input image  $I^l \in R^{c^l * w^l * h^l}$ , meta filters  $W^l \in R^{c^{l+1} * c^l * z' * z'}$ ,  
effective filter size  $z * z$ , pooling size  $s * s$ .

**Output:** Output image  $I^{l+1} \in R^{nc^{l+1} * w^{l+1} * h^{l+1}}$ , with  $n = \frac{(z'-z+1)^2}{s^2}$ .

**function** DOUBLE CONVOLUTION

1.  $I^l \leftarrow IdentityMatrix(c^l z^2)$ ;
2. Reorganize  $I^l$  to shape  $c^l z^2 * c^l * z * z$ ;
3.  $\tilde{W}^l \leftarrow W^l * I^l$ ; /\* output shape:  $c^{l+1} * c^l z^2 * (z' - z + 1) * (z' - z + 1) *$  /
4. Reorganize  $\tilde{W}^l$  to shape  $c^{l+1} (z' - z + 1)^2 * c^l * z * z$ ;
5.  $O^{l+1} \leftarrow I^l * \tilde{W}^l$ ; /\* output shape:  $c^{l+1} (z' - z + 1)^2 * w^{l+1} * h^{l+1} *$  /
6. Reorganize  $O^{l+1}$  to shape  $c^{l+1} w^{l+1} h^{l+1} * (z' - z + 1) * (z' - z + 1)$ ;
7.  $I^{l+1} \leftarrow pool_s(O^{l+1})$ ; /\* output shape:  $c^{l+1} w^{l+1} h^{l+1} * \frac{z'-z+1}{s} * \frac{z'-z+1}{s} *$  /
8. Reorganize  $I^{l+1}$  to shape  $c^{l+1} (\frac{z'-z+1}{s})^2 * w^{l+1} * h^{l+1}$ ;

**end function**

---

# Implementation & Results



# MNIST DATASET

Input: 1x28x28 (GrayScale Image)

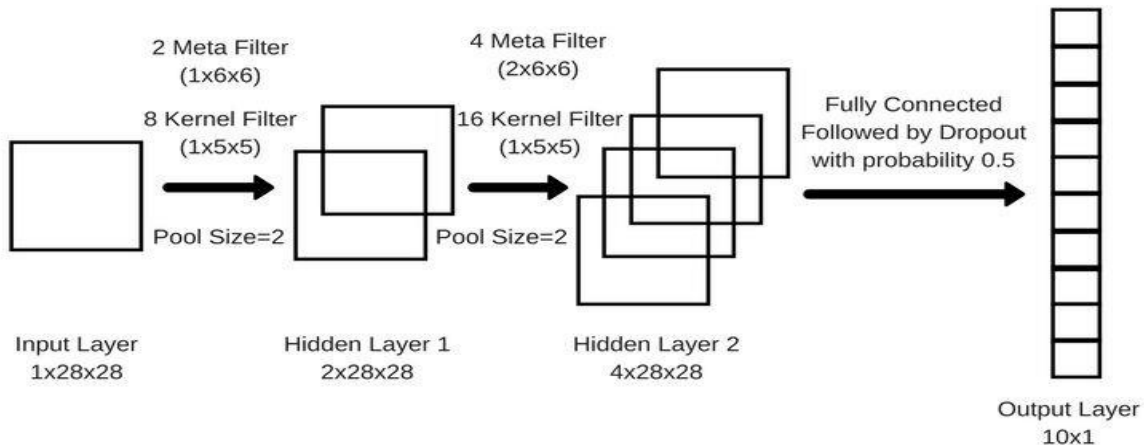
Class: 10 (0,1,2,...,9)

Train Samples: 60,000

Test Samples: 10,000

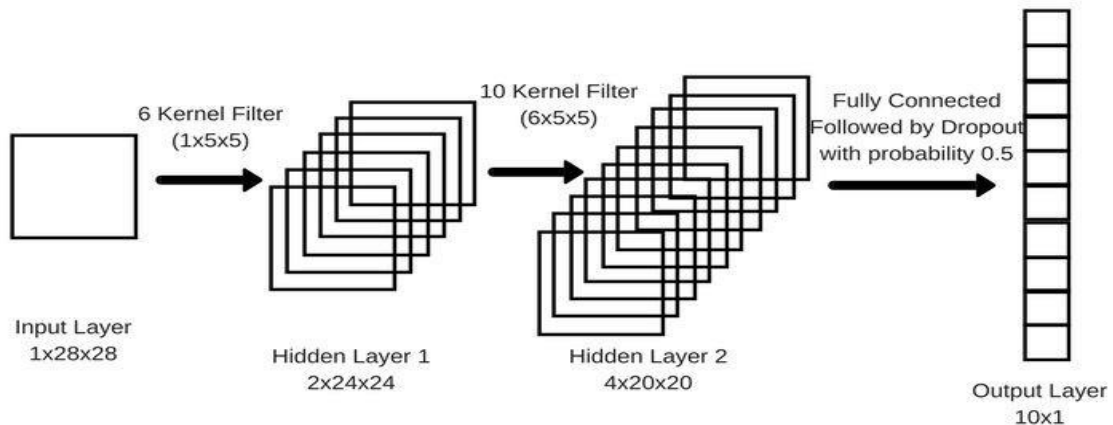
## DCNN Model

$$\begin{aligned}\text{Total Parameters} &= 2 * 1 * 6 * 6 + 4 * 2 * 6 * 6 \\ &= 360\end{aligned}$$



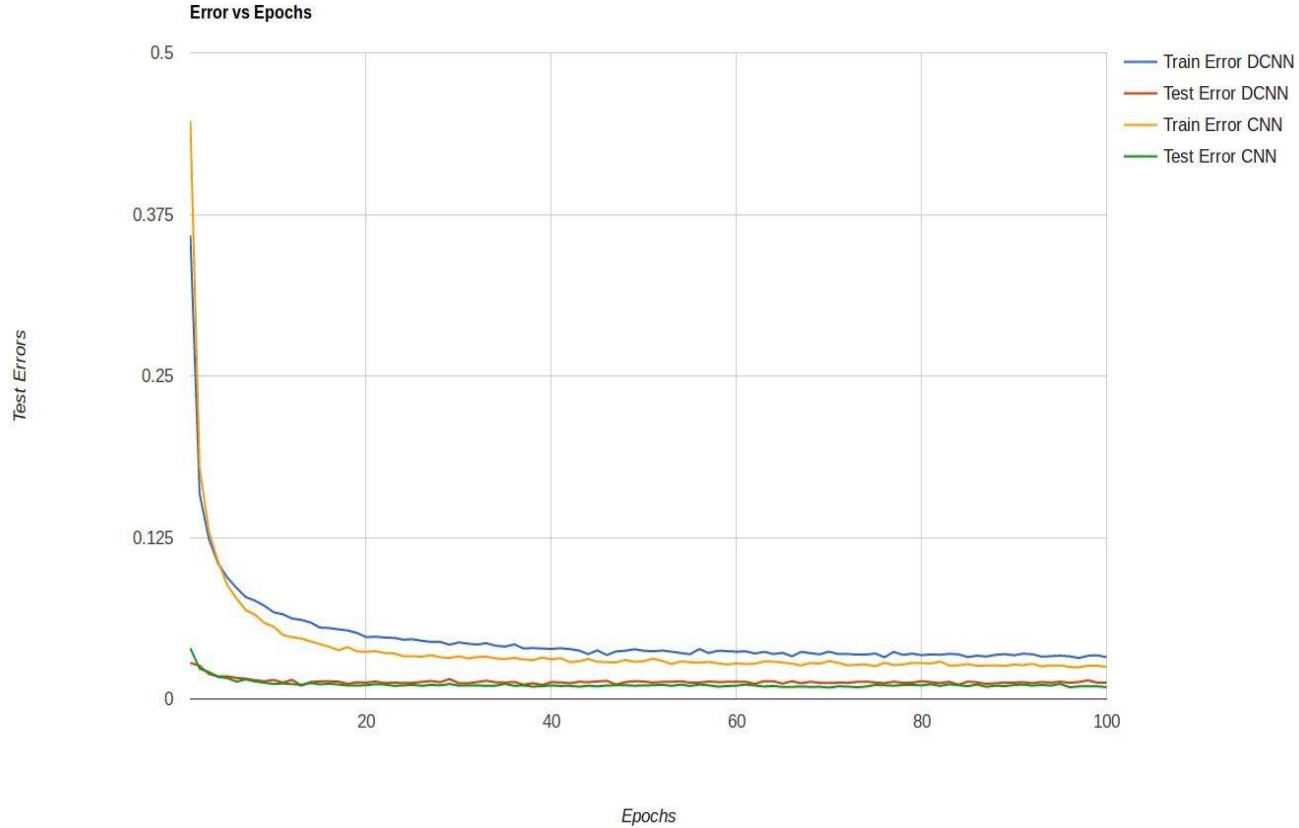
## CNN Model

$$\begin{aligned}\text{Total Parameters} &= 6 * 1 * 5 * 5 + 10 * 6 * 5 * 5 \\ &= 1650\end{aligned}$$





Batch Size: 200 Epochs: 100 Dropout: Yes



Minimum Error Values:

DCNN Train: 0.032 at 97

DCNN Test: 0.01 at 13

CNN Train: 0.025 at 97

CNN Test: 0.009 at 70

# DCNN vs CNN

Epochs	Pool	Batch Size	Dropout	Test Error DCNN	Test Error CNN
10	2	200	No	0.0137	0.019
9	1	100	No	0.018	0.017
10	2	200	Yes	0.0153	0.0171

## Conclusion:

Even though DCNN has 360 params compare to CNN which has 1650 params, Test Error is almost comparable.

Forward Pass Run is Faster in DCNN.

Convergence for DCNN is much faster and after that overfitting happens quickly compare to CNN

# Variants of DCNN

## Standard CNN

$$z' = z$$

DCNN is generalisation of CNN

## Concat DCNN

$$s = 1$$

Maximally parameter efficient

With the same amount of parameters produces  $\frac{(z' - z + 1)^2 z^2}{z'^2}$

times more channels for a single layer.

## Maxout DCNN

$$s = z' - z + 1$$

Output image channel size equal to the number of meta filters.

Yields a parameter efficient implementation of maxout network.

# What's Next?

- Instead of translational correlation modeling for Rotational Correlation.
- Mechanism to decide number of meta filters and its size.

# References

- GitHub: Aryan05
- Doubly Convolutional Neural Networks (NIPS 2016) by Shuangfei Zhai, Yu Cheng, Weining Lu and Zhongfei (Mark) Zhang  
<https://papers.nips.cc/paper/6340-doubly-convolutional-neural-networks.pdf>
- Getting Started with Lasagne: <http://luizgh.github.io/libraries/2015/12/08/getting-started-with-lasagne/>
- Lasagne Docs: <https://lasagne.readthedocs.io/en/latest/>
- Theano Docs: <http://deeplearning.net/software/theano/library/index.html>