Lecture 5b

Convolutional networks

Convolution and correlation

correlation

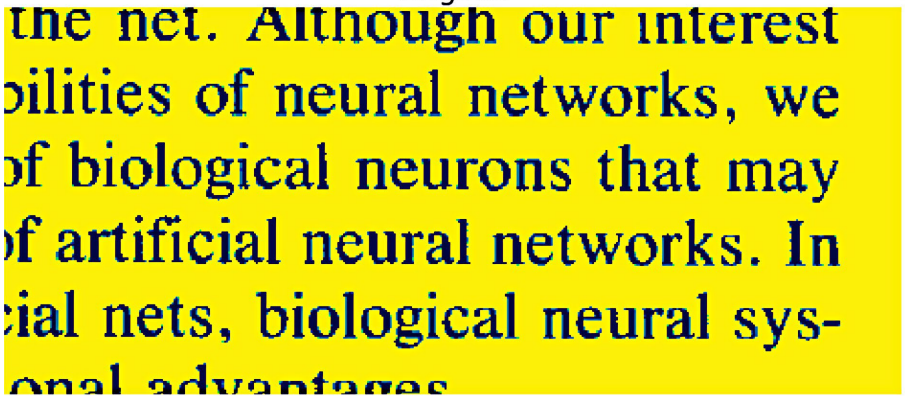
$$g(x',y') = \iint f(x,y)\,h(x-x',y-y')\,dx\,dy$$

convolution

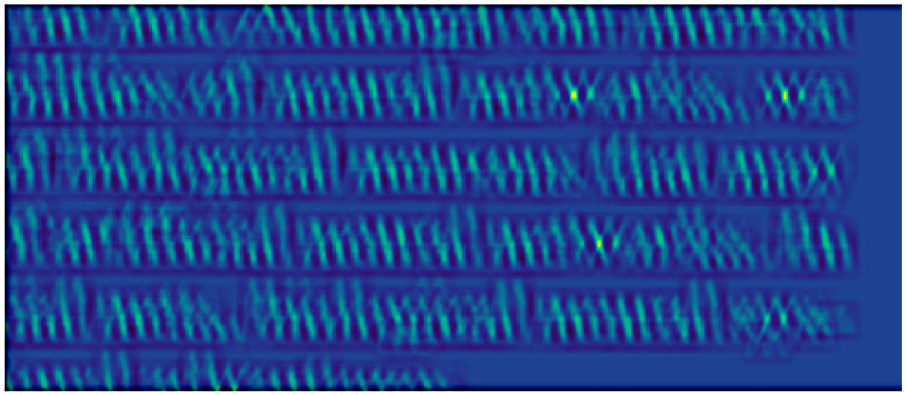
$$g(x',y') = \iint f(x,y)\,h(x'-x,y'-y)\,dx\,dy$$

Correlator

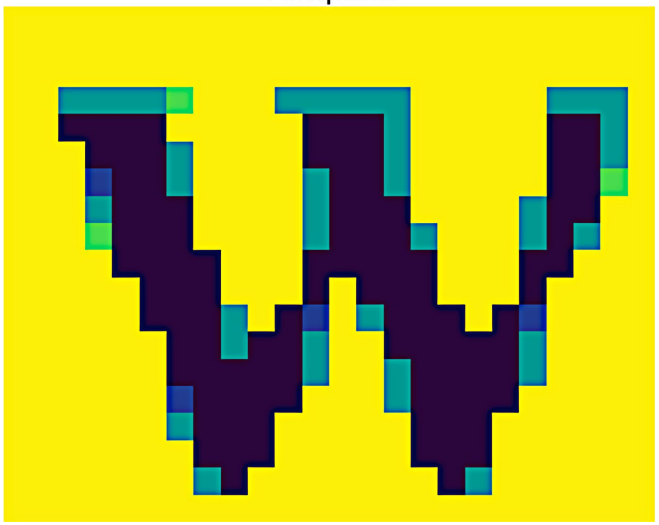
Original



Cross-correlation

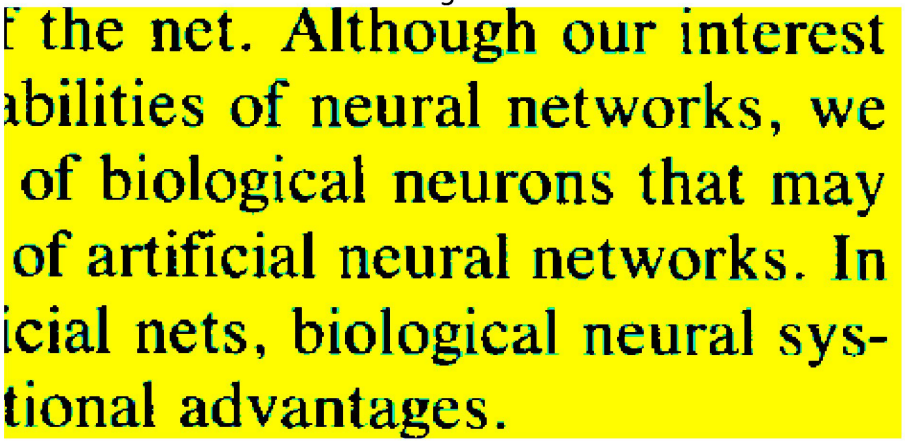


Template

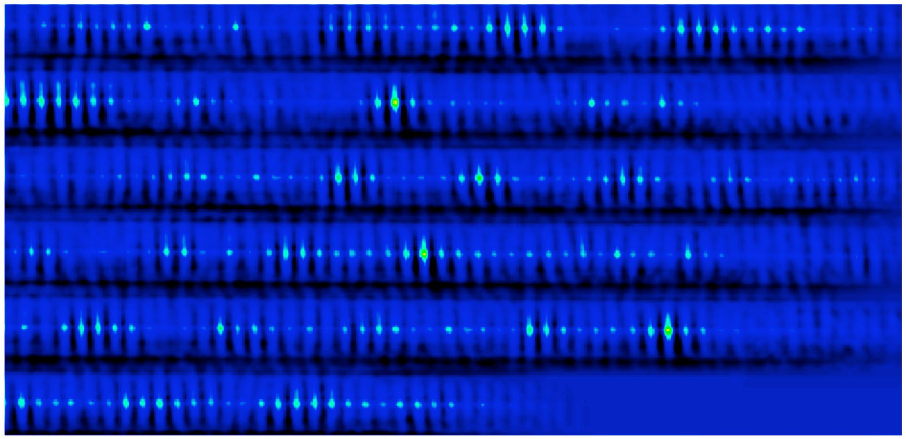


Correlator

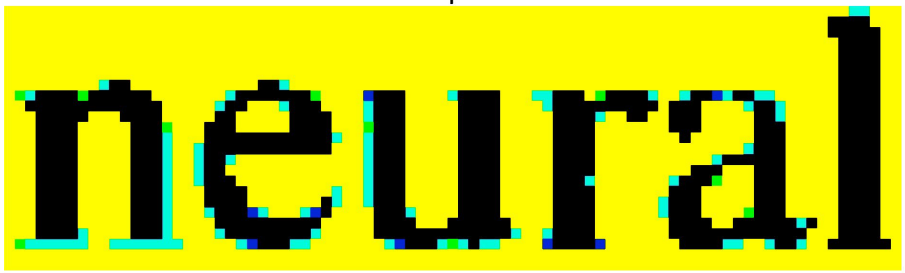
Original



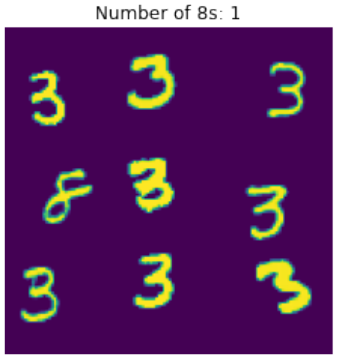
Cross-correlation

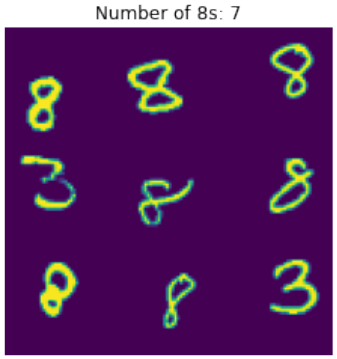


Template



Counting digits on an image





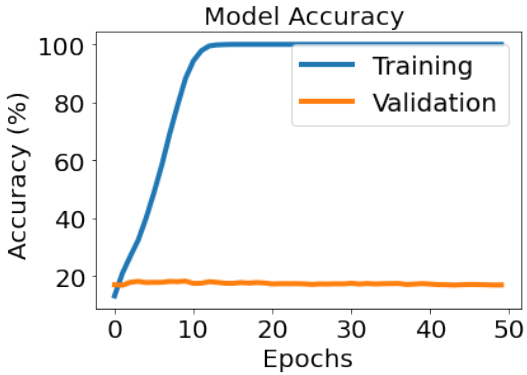
Dataset: 3's and 8's from MNIST, randomly shifted, placed in a 3x3 tile, final resolution: 120x120, 16000 training, 4000 test samples. Labels are the number of 8s in the image (0 to 9)

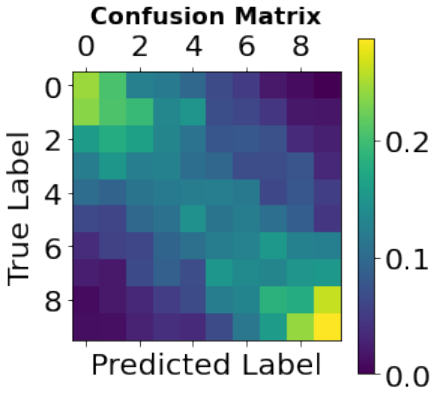
Fully Connected:

- 2 hidden layers with 100 neurons, sigmoid nonlinearity
- Output layer with 10 neurons, softmax nonlinearity

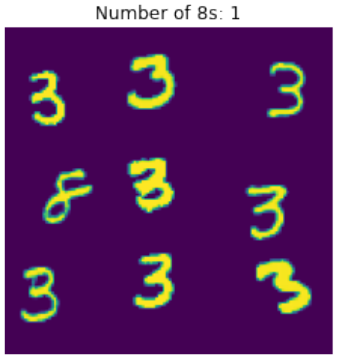
Total Number of Params: **1.45 M**

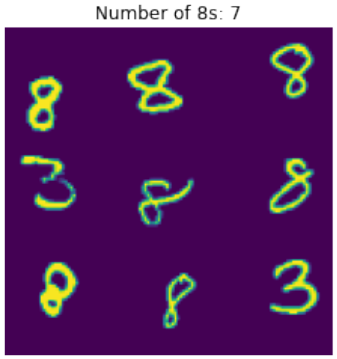
Test Accuracy: **17%**





Counting digits on an image





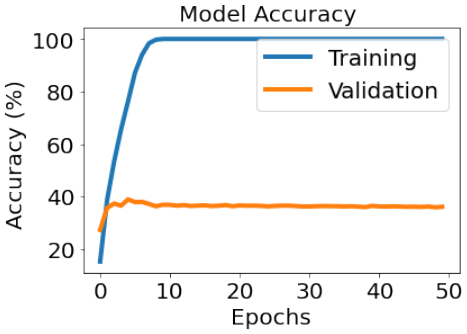
Dataset: 3's and 8's from MNIST, randomly shifted, placed in a 3x3 tile, final resolution: 120x120, 16000 training, 4000 test samples. Labels are the number of 8s in the image (0 to 9)

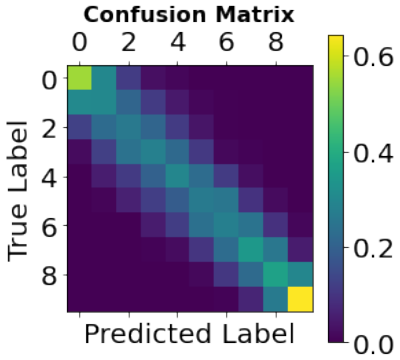
Convolutional:

- 1 conv. layer with 4, 28x28 kernels, relu nonlinearity, no stride
- Flattening
- Output layer with 10 neurons, softmax nonlinearity

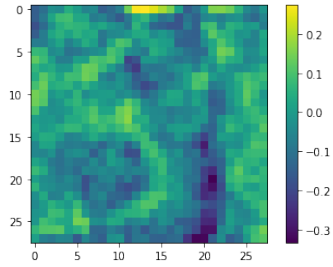
Total Number of Params: **0.35 M**

Test Accuracy: **36%**

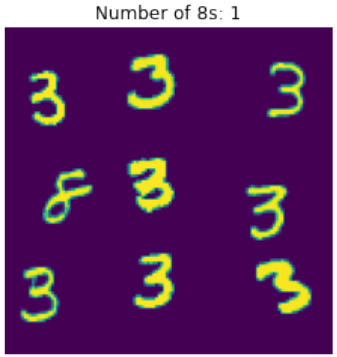


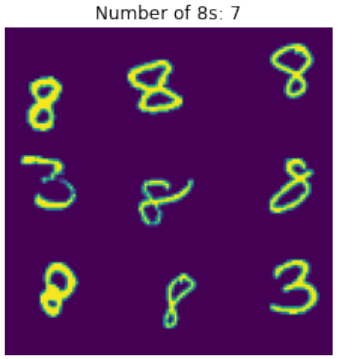


One of the conv. kernels



Counting digits on an image





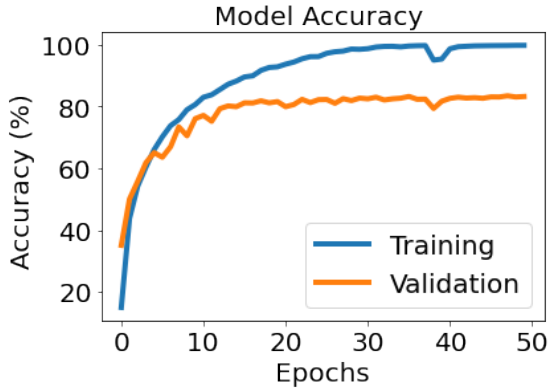
Dataset: 3's and 8's from MNIST, randomly shifted, placed in a 3x3 tile, final resolution: 120x120, 16000 training, 4000 test samples. Labels are the number of 8s in the image (0 to 9)

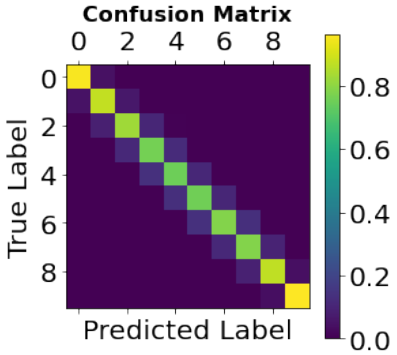
Convolutional:

- 3 conv. layers with 4,8 and 16 6x6 kernels, relu nonlinearity, 2x2 stride
- Flattening
- Fully connected layer with 100 units, sigmoid
- Output layer with 10 neurons, softmax nonlinearity

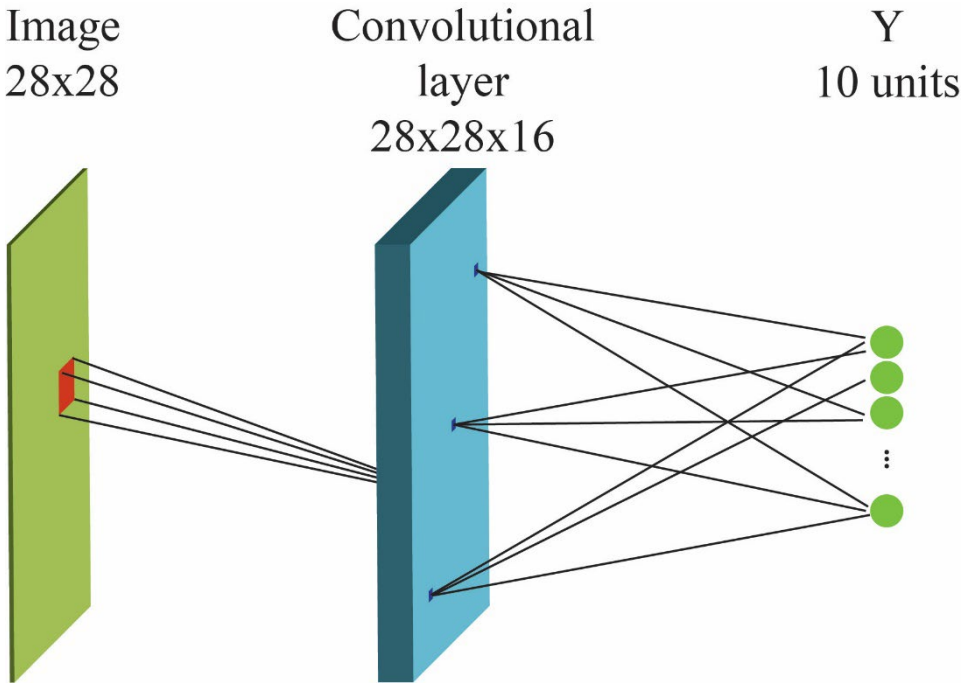
Total Number of Params: **0.15 M**

Test Accuracy: **83%**



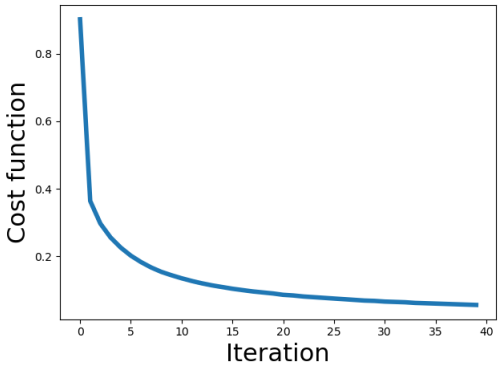


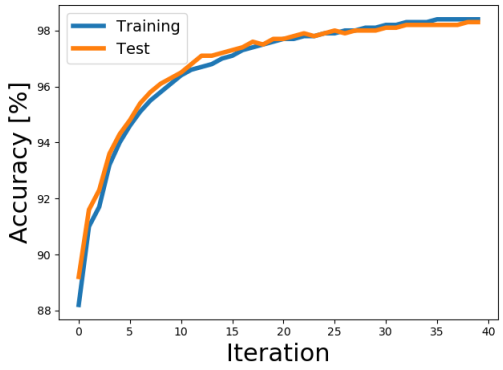
Two-layer convolutional network (digits MNIST)



60000 Training samples
10000 Test samples

- Optimizer: gradient descent, MSE
- Rectified linear activation function in the middle layer
- Softmax activation function in the final layer
- Receptive field : 20x20
- Batch size : 1000
- Epochs: 40
- Learning rate:0.05





Convolutional 1st layer
Training accuracy: 98.4%
Test accuracy: 98.3% 125K weights

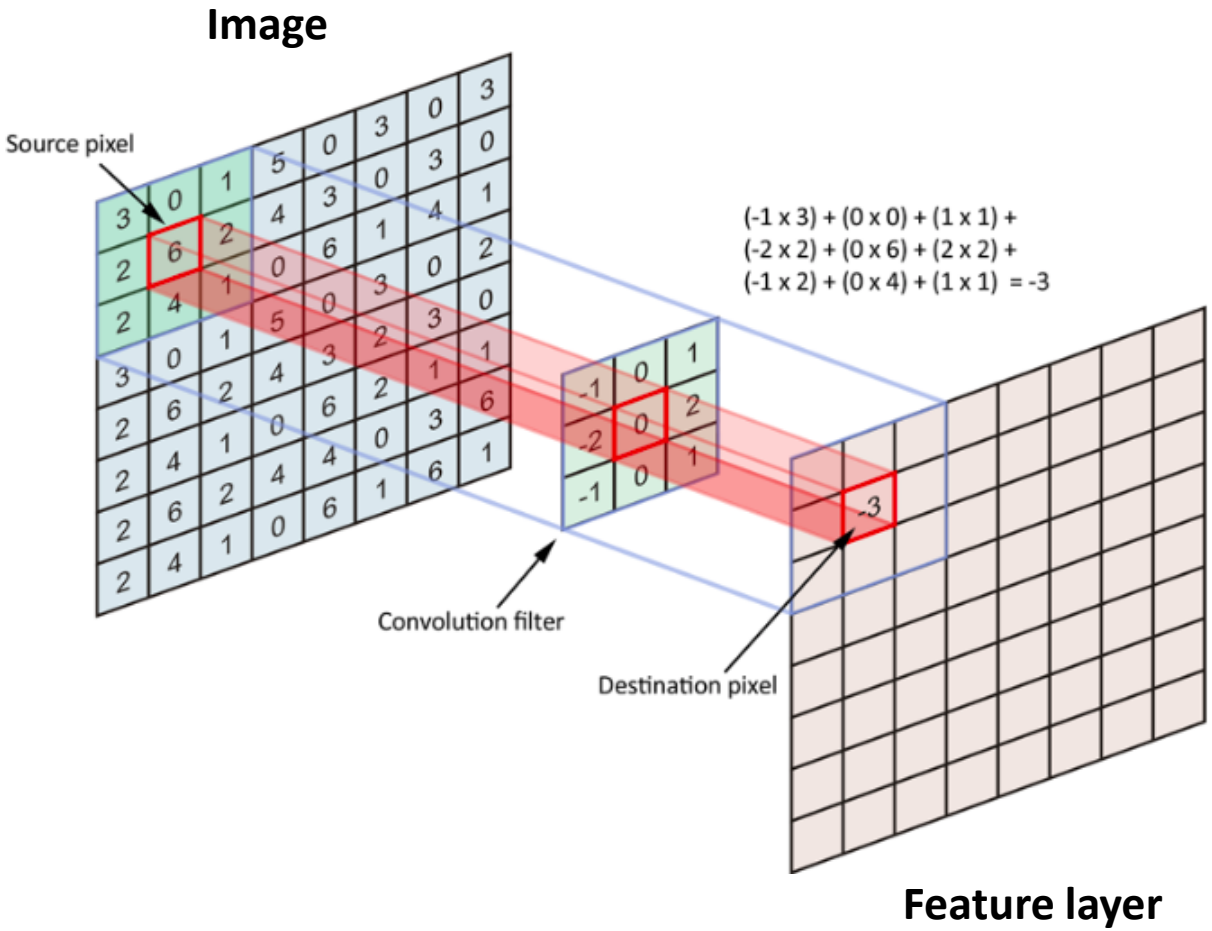
Fully connected 1st layer (20 HU)
Training accuracy: 97%
Test accuracy: 95.7% 16K weights

Fully connected network parameters: 100 epoch, 1 learning rate, sigmoid activation in the middle layer, 1000 batch size

The receptive fields of the convolutional layer are trained simultaneously for all positions

Feature detection using discrete 2-dimensional convolution with a filter

Template matching



Convolution of the image with a learnt filter (template) identifies the structure in the image and localises it in the feature layer

Convolution: Single channel input – single filter

Input

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

Input – ***image or feature layer from previous convolution block***
Size: $[L_x, L_y] = [8, 8]$
Channels: $[n_c] = 1$
Zero padded

0	1	0
0	1	0
0	1	0

Filter kernel
Filters = $n_f = 1$
Size: $[f_x, f_y] = [3, 3]$
Filter depth = $n_c = 1$
Filter bias = $b = 0$
Filter weights:
$= f_x \times f_y \times n_c + 1$
$= 3 \times 3 \times 1 + 1 = 10$
Stride: $[s_x, s_y] = [1, 1]$

Feature layer: 1/1

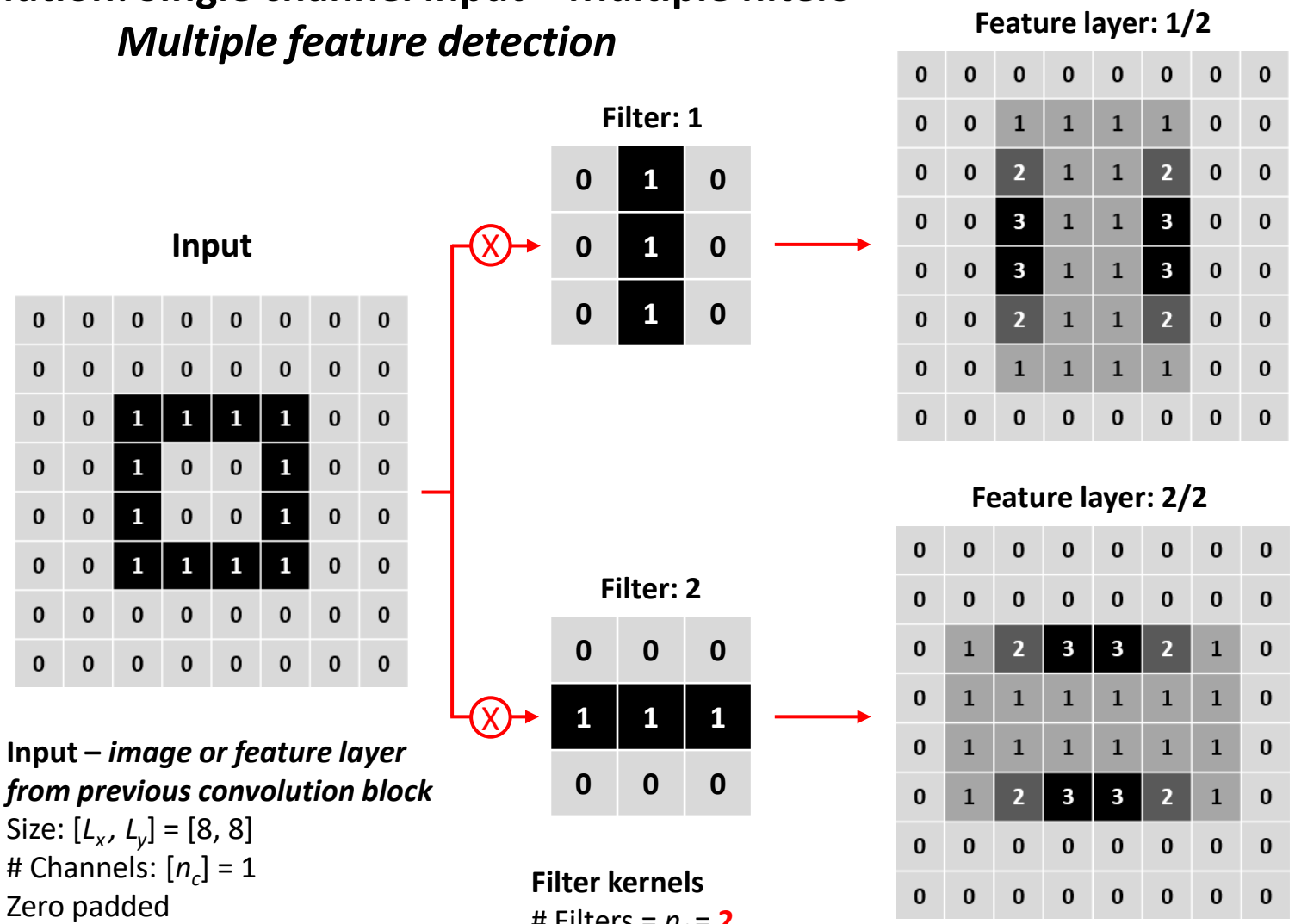
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	2	1	1	2	0	0
0	0	3	1	1	3	0	0
0	0	3	1	1	3	0	0
0	0	2	1	1	2	0	0
0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0

Output – ***feature layer***
Feature layers = $n_f = 1$
Size: $[F_x, F_y] = [8, 8]$
Channels / feature = 1

Detects vertical edges

Convolution: Single channel input – Multiple filters

Multiple feature detection



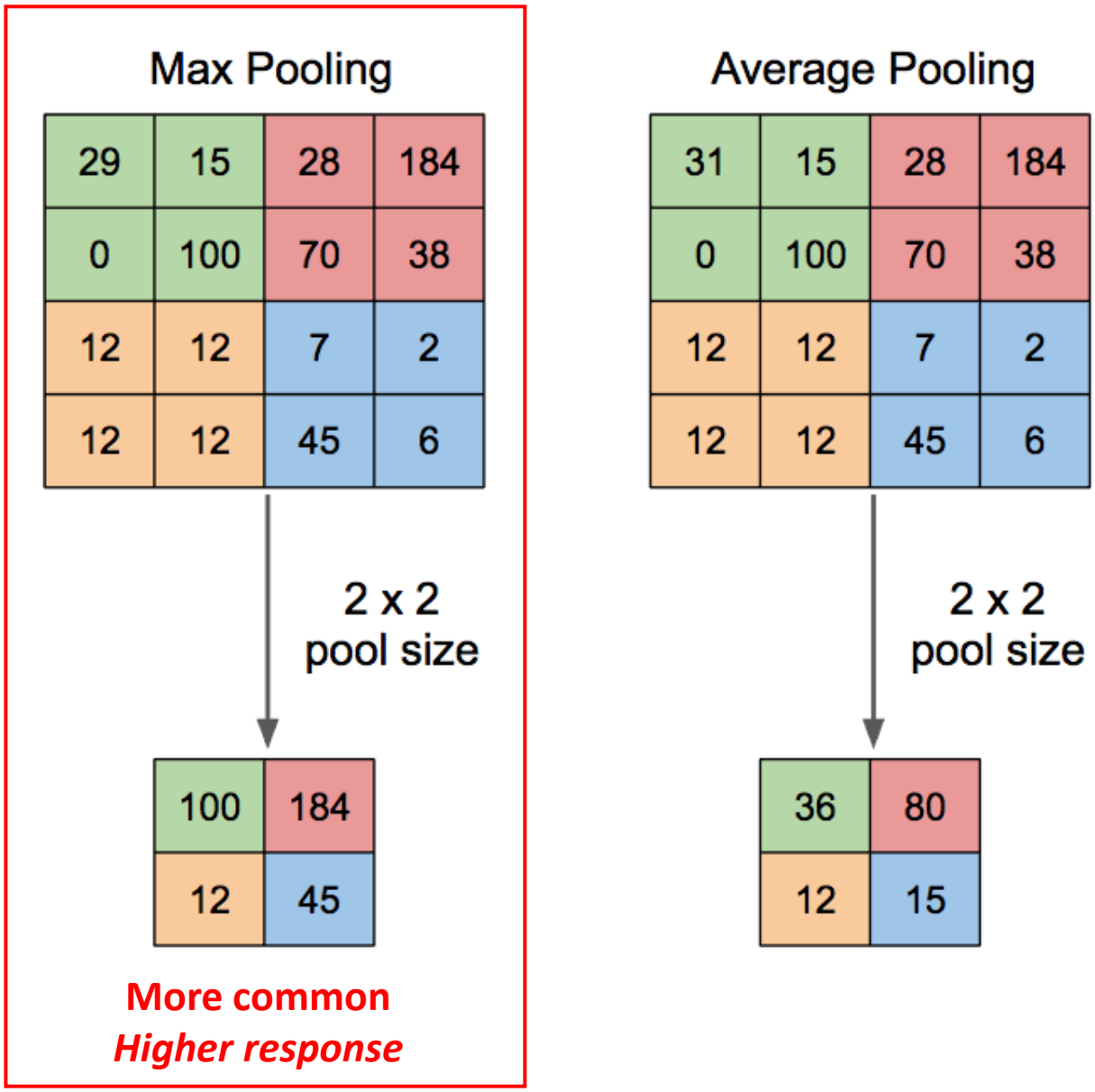
Input – ***image or feature layer from previous convolution block***
Size: $[L_x, L_y] = [8, 8]$
Channels: $[n_c] = 1$
Zero padded

Filter kernels
Filters = n_f = **2**
Size: $[f_x, f_y] = [3, 3]$
Filter depth = $n_c = 1$
Filter bias = $b = 0$
Filter weights = 10
Stride: $[s_x, s_y] = [1, 1]$

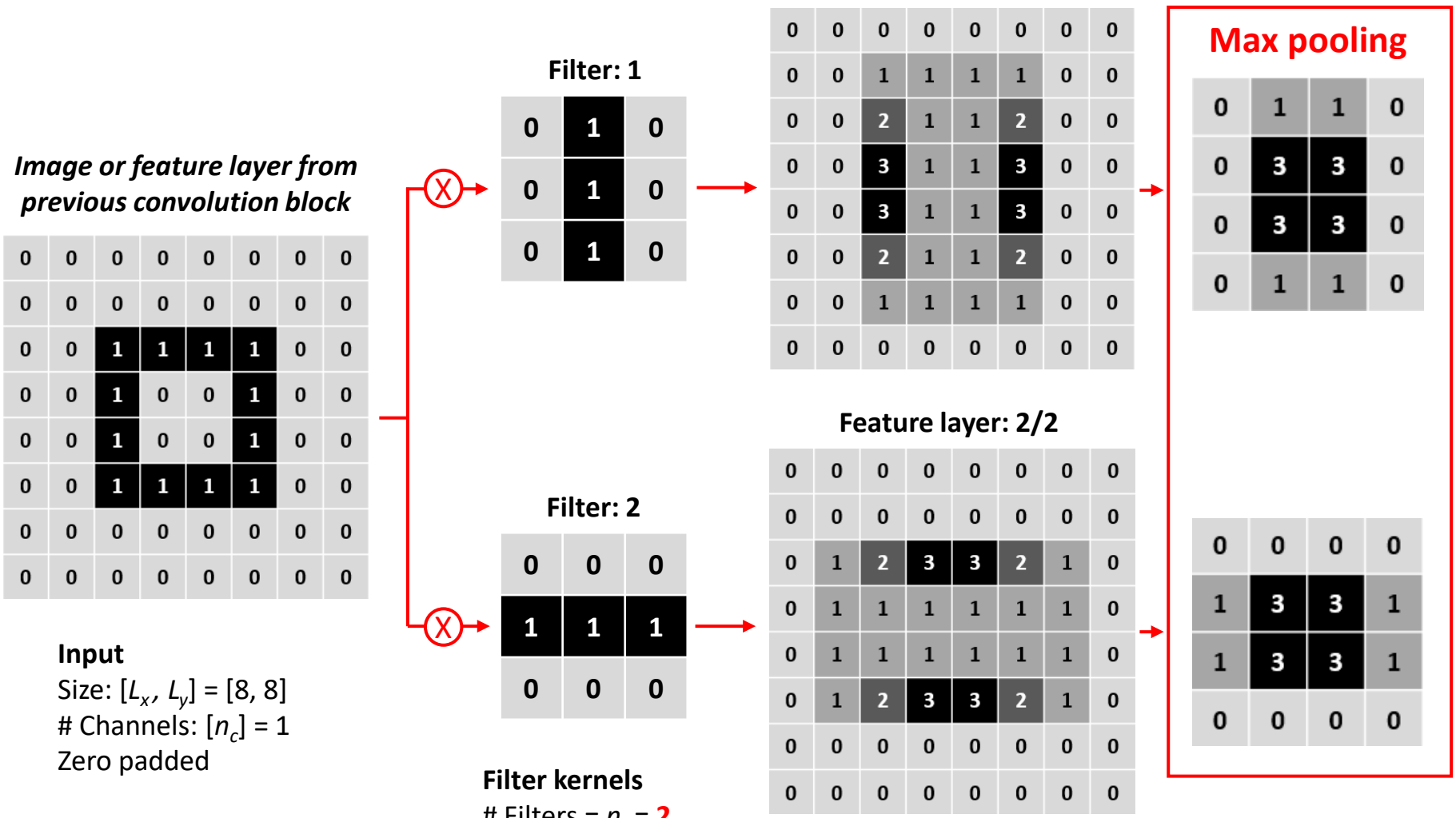
Output – ***feature layers***
Feature layers = n_f = **2**
Size: $[F_x, F_y] = [8, 8]$
Channels / feature = 1

Detects vertical and horizontal edges

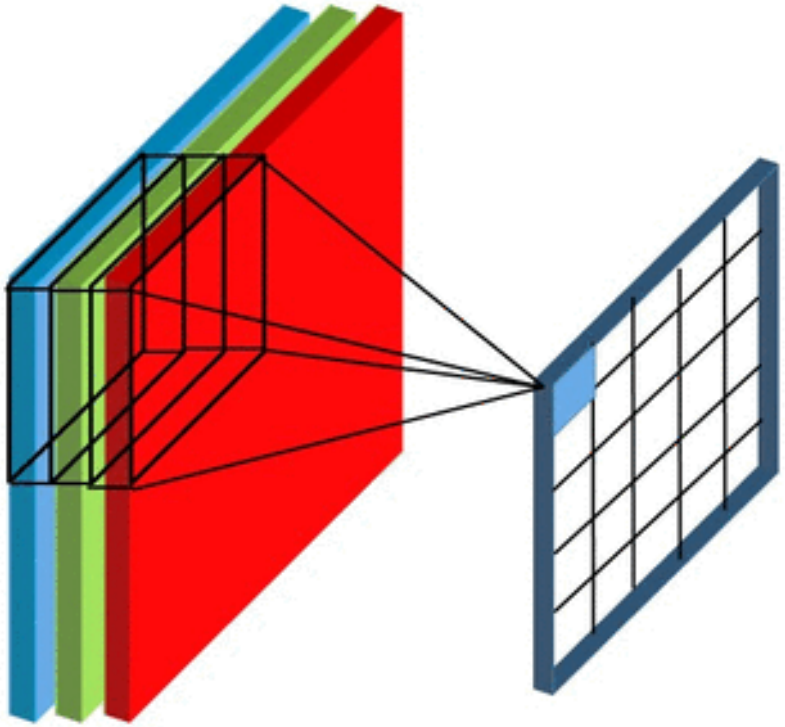
Spatial down-sampling using pooling functions



Max pooling of feature layers

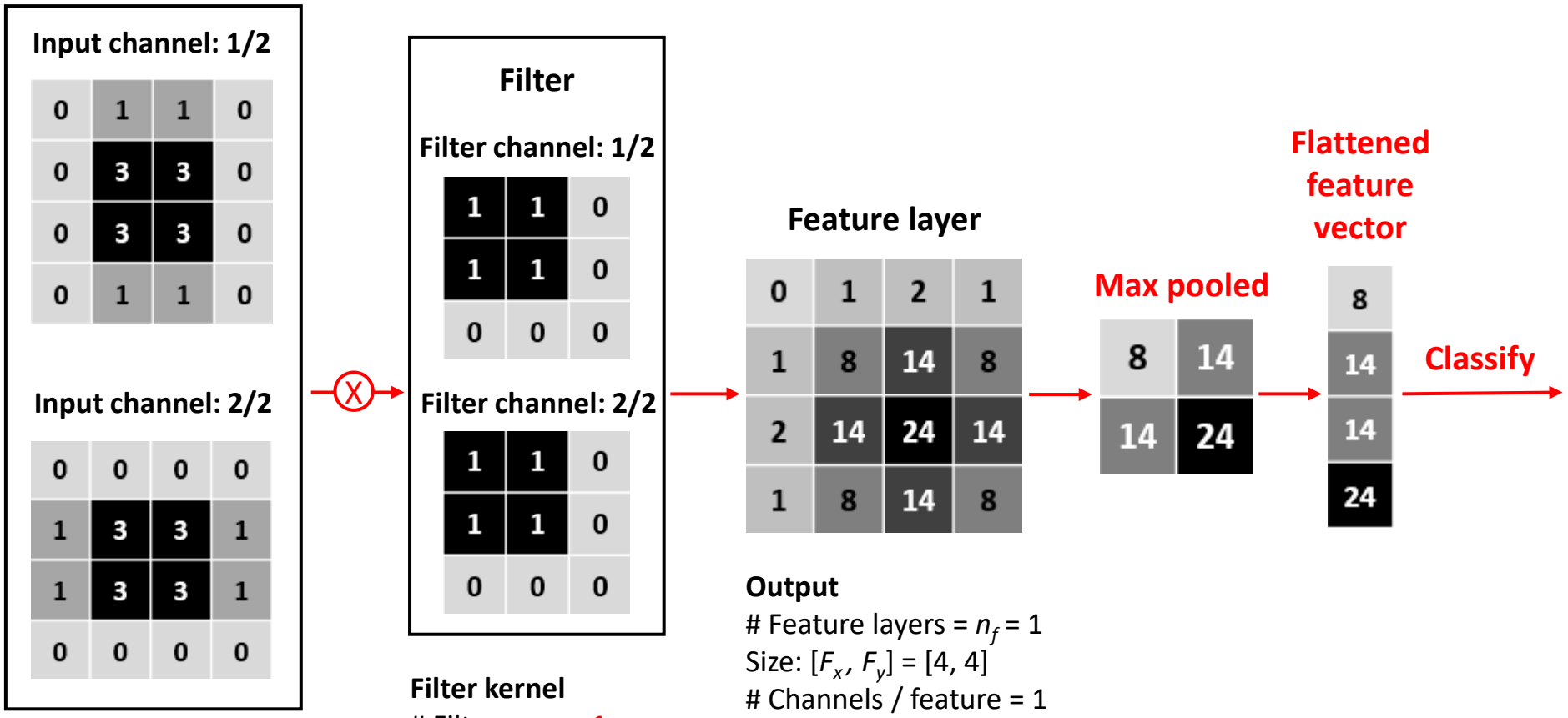


Convolution: Multiple channel input – Single filter



Note: no activation function has been applied in this case – linear demo

Convolution: Multiple channel input – Single filter



Input
Size: $[L_x, L_y] = [4, 4]$
Channels: $[n_c] = \mathbf{2}$
Zero padded

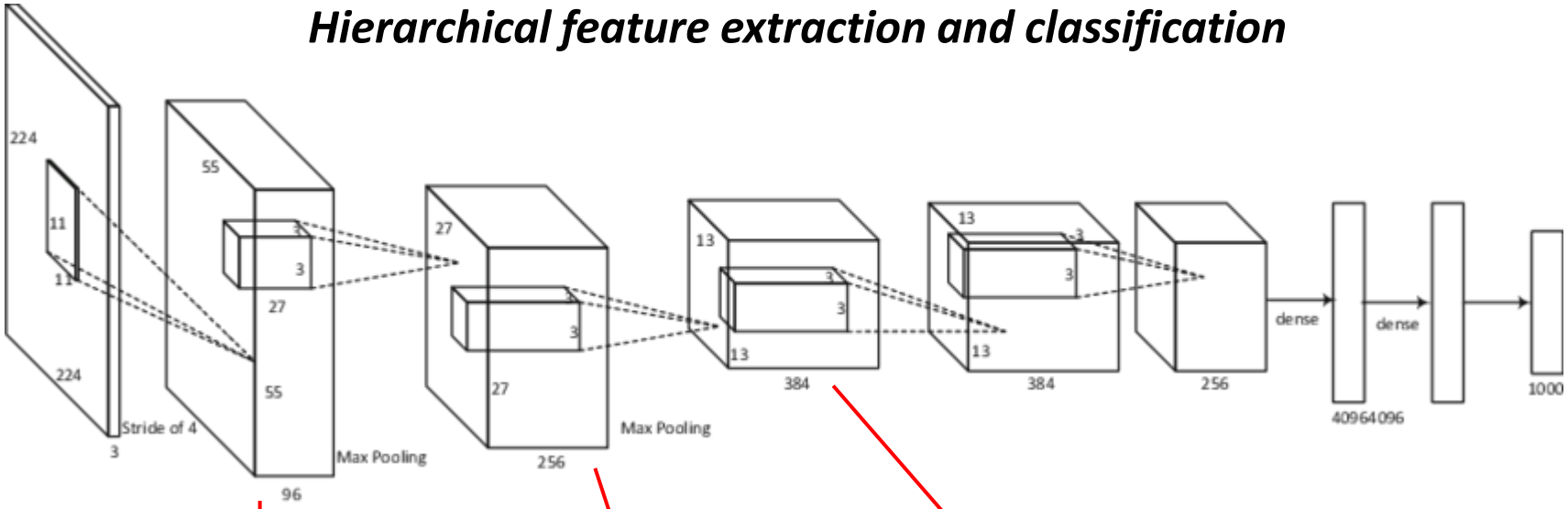
Filter kernel
Filters = $n_f = \mathbf{1}$
Size: $[f_x, f_y] = [3, 3]$
Filter depth = $n_c = \mathbf{2}$
Filter bias = $b = 0$
Filter weights:
$= f_x \times f_y \times n_c + 1$
$= 3 \times 3 \times 2 + 1 = 19$
Stride: $[s_x, s_y] = [1, 1]$

Output
Feature layers = $n_f = 1$
Size: $[F_x, F_y] = [4, 4]$
Channels / feature = 1

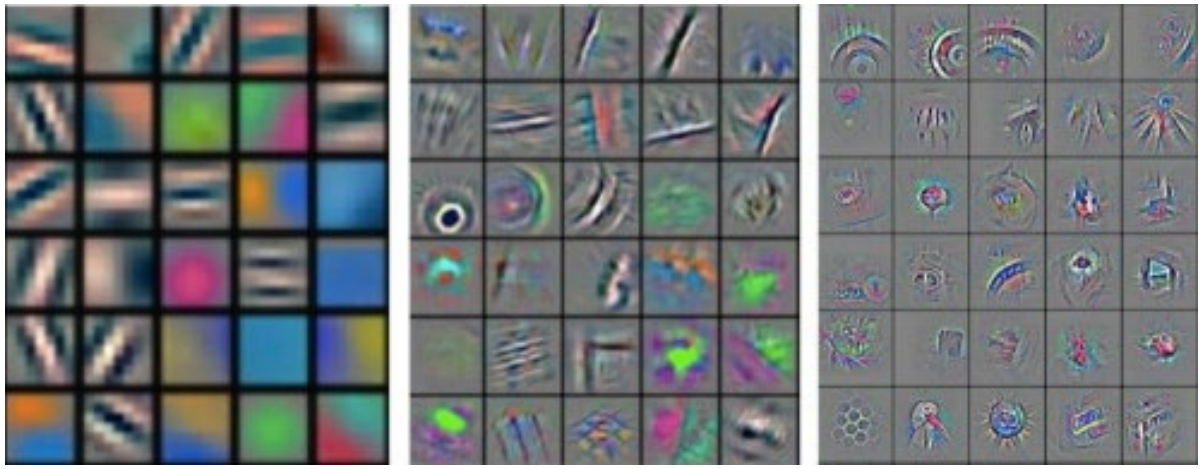
Detects a square

nb: no activation function has been applied after max pooling

VGG type convolutional neural network (CNN):
Hierarchical feature extraction and classification



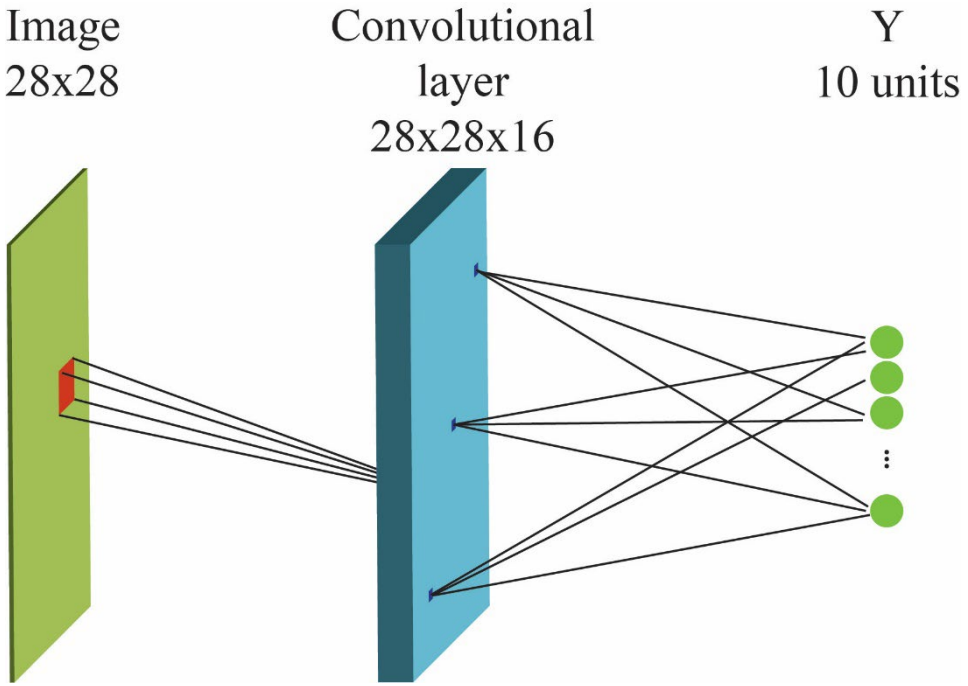
Fully connected classifier frontend



Low-level features Mid-level features High-level features

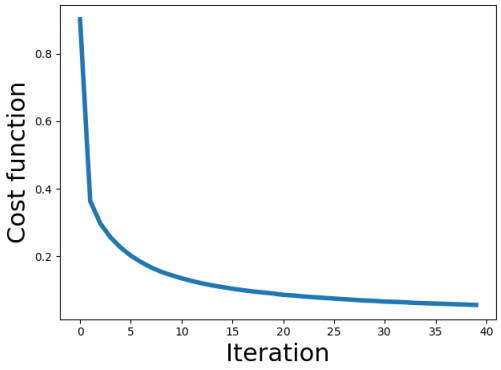
Convolutional feature extraction backend

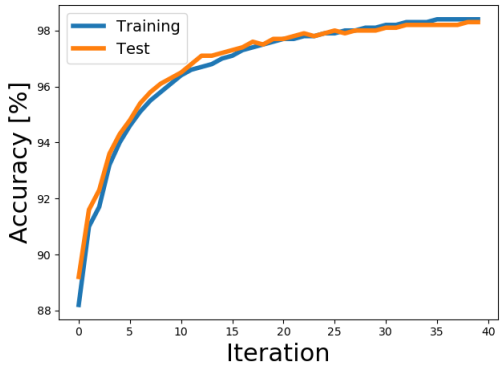
Two-layer convolutional network (digits MNIST)



60000 Training samples
10000 Test samples

- Optimizer: gradient descent, MSE
- Rectified linear activation function in the middle layer
- Softmax activation function in the final layer
- Receptive field : 20x20
- Batch size : 1000
- Epochs: 40
- Learning rate:0.05





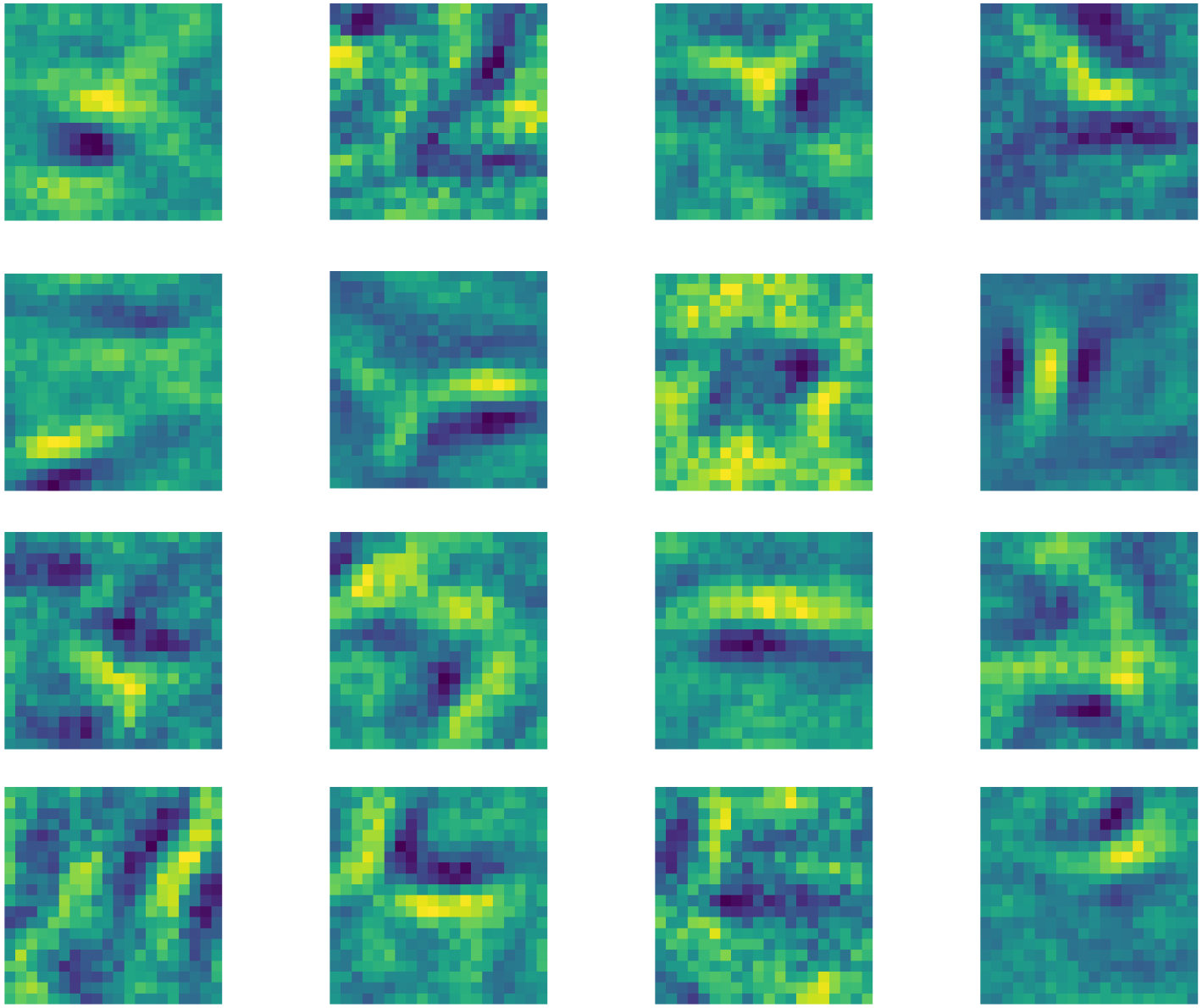
Convolutional 1st layer
Training accuracy: 98.4%
Test accuracy: 98.3% 125K weights

Fully connected 1st layer (20 HU)
Training accuracy: 97%
Test accuracy: 95.7% 16K weights

Fully connected network parameters: 100 epoch, 1 learning rate, sigmoid activation in the middle layer, 1000 batch size

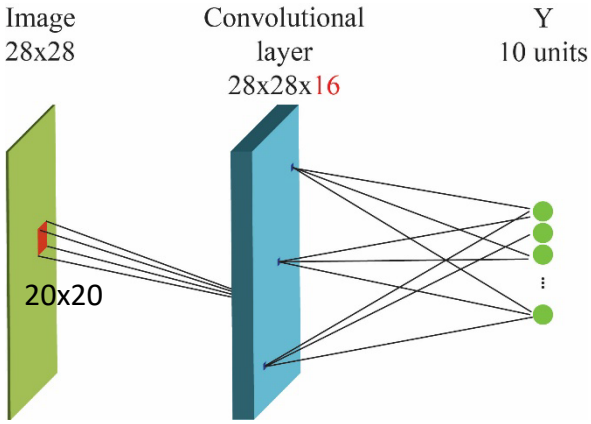
The receptive fields of the convolutional layer are trained simultaneously for all positions

2-layer convolutional network
16 Receptive fields

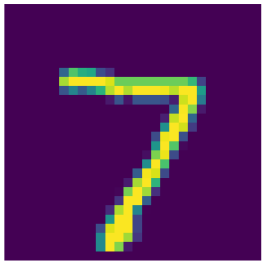


Two-layer convolutional network

Output of the convolutional layer: 16 receptive fields

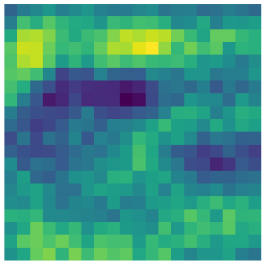


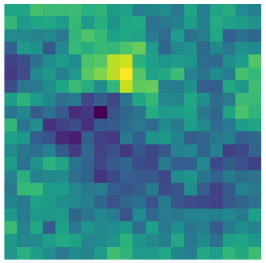
Input image



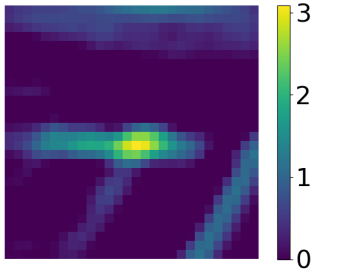
Input image

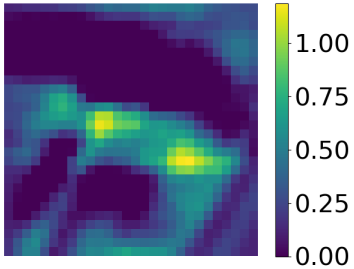
Receptive field
2 out of 16



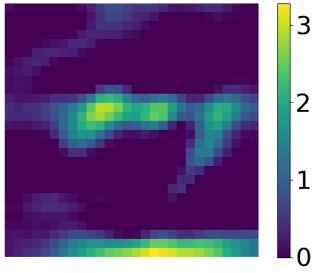


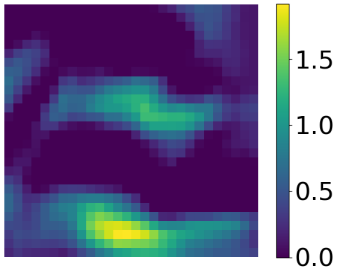
Convolutional layer output





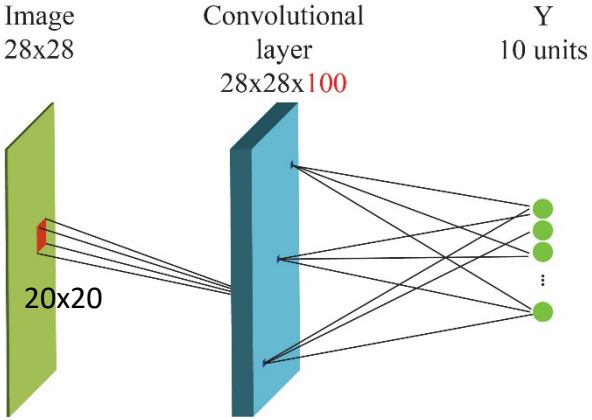
Convolutional layer output



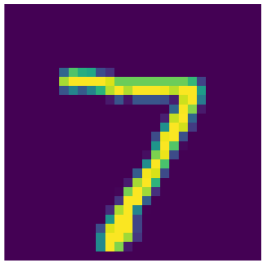


Two-layer convolutional network

Output of the convolutional layer: 100 receptive fields

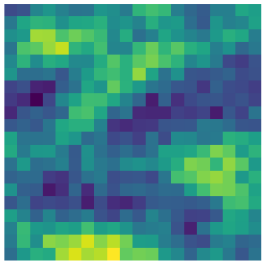


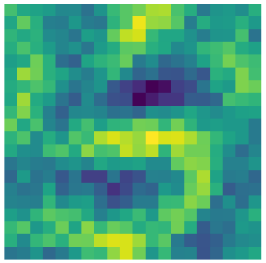
Input image



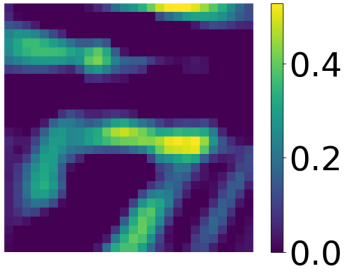
Input image

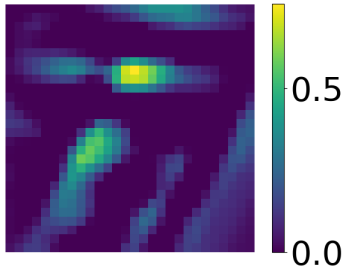
Receptive field



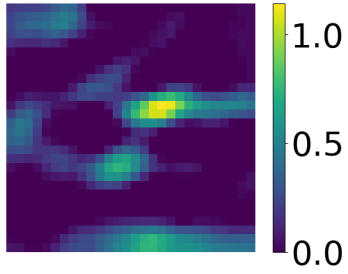


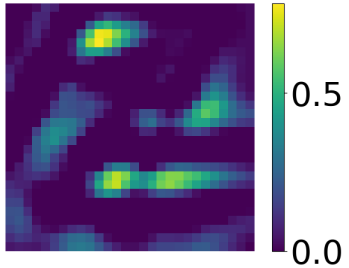
Convolutional layer output





Convolutional layer output



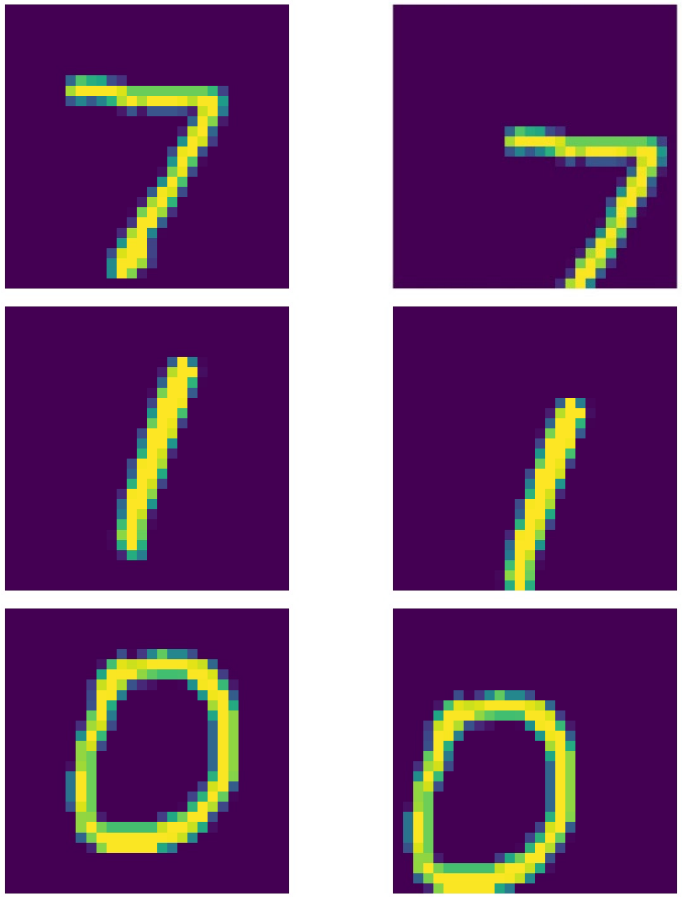


Fully connected and Convolutional network

Shifted digits

Examples

Original Shifted



Convolutional neural network

- Optimizer: gradient descent, MSE
- Rectified linear activation function in the middle layer
- Softmax activation function in the final layer
- Receptive field : 20x20 with 16 receptive fields
- Batch size : 1000
- Epochs: 40
- Learning rate:0.05

Fully connected network

- Optimizer: gradient descent, MSE
- Sigmoid activation function in the middle layer
- Softmax activation function in the final layer
- Hidden units: 20
- Batch size : 1000
- Epochs: 100
- Learning rate: 1

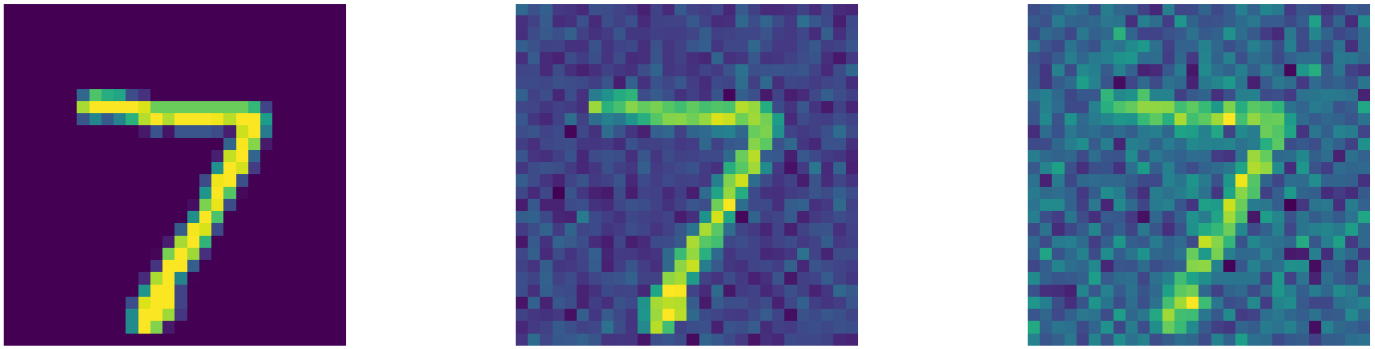
Fully connected and Convolutional network
Shifted digits

	Shifted images are not included in the training		Shifted images are included in the training	
	Test accuracy (%) Unshifted images	Test accuracy (%) Shifted images	Test accuracy (%) Unshifted images	Test accuracy (%) Shifted images
Convolutional neural network				
2 receptive fields	97.2	42.9	97.2	90.6
16 receptive fields	98.3	53.4	98.5	96
100 receptive fields	98.4	54	98.7	96.7
Fully connected network				
20 hidden units	95.7	35.8	93.7	80.8

Two-layer convolutional network – Additive noise

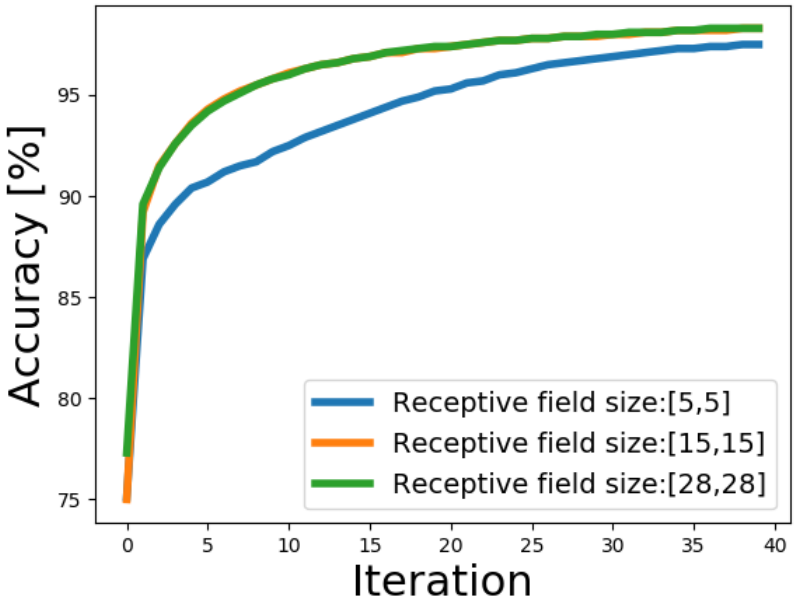
In this example, noisy images are not included in the training.

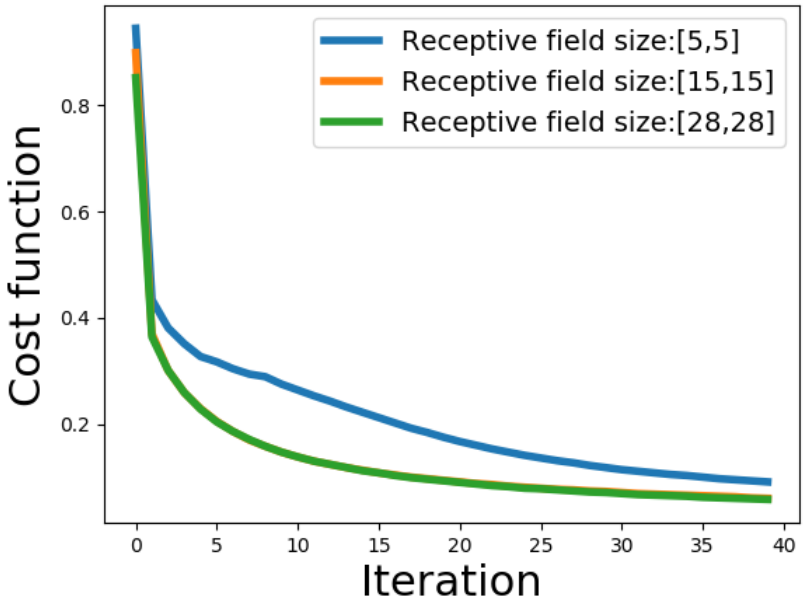
Variance of random distribution



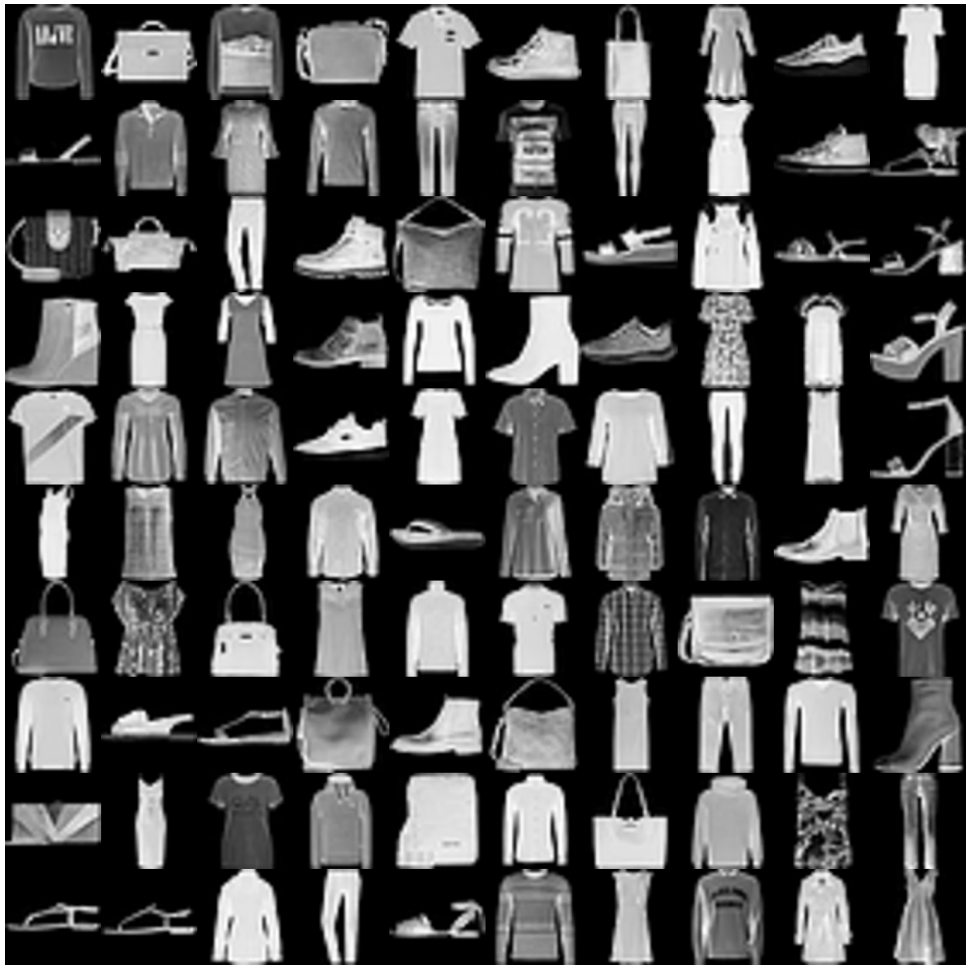
	0	0.01	0.05
Convolutional neural network		**Accuracy on the test set (%)**	
2 receptive fields 20x20 pixels each	97.2	44.3	16.00
16 receptive fields	98.3	47.47	25.63
100 receptive fields	98.4	68.21	34.65
Fully connected network			
20 hidden units	95.7	27	16.9

Two-layer convolutional network - Receptive fields



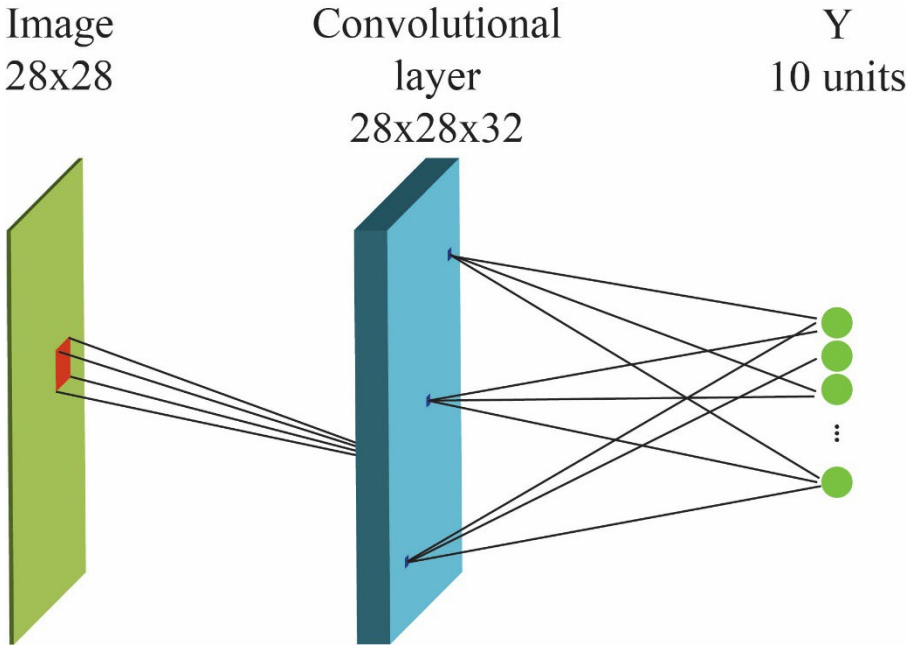


MNIST Fashion



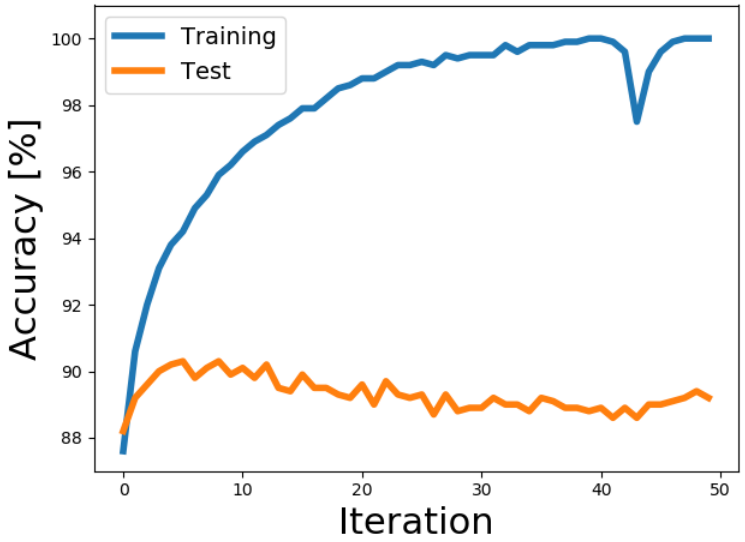
- 10 different classes
- 60000 training samples
- 10000 test samples

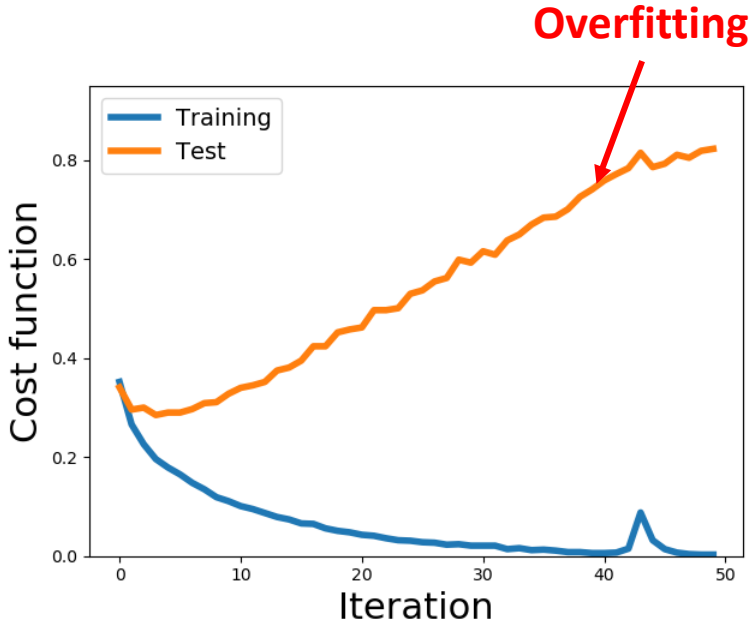
MNIST Fashion: 2-layer network



60000 Training samples
10000 Test samples

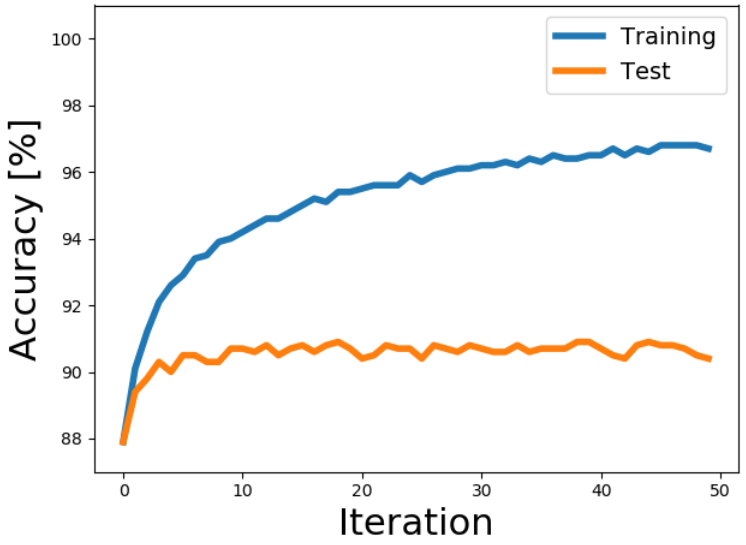
- Optimizer: gradient descent
- Rectified linear activation function in the middle layer
- Softmax activation function in the final layer
- Receptive field : 3x3
- Batch size : 1000

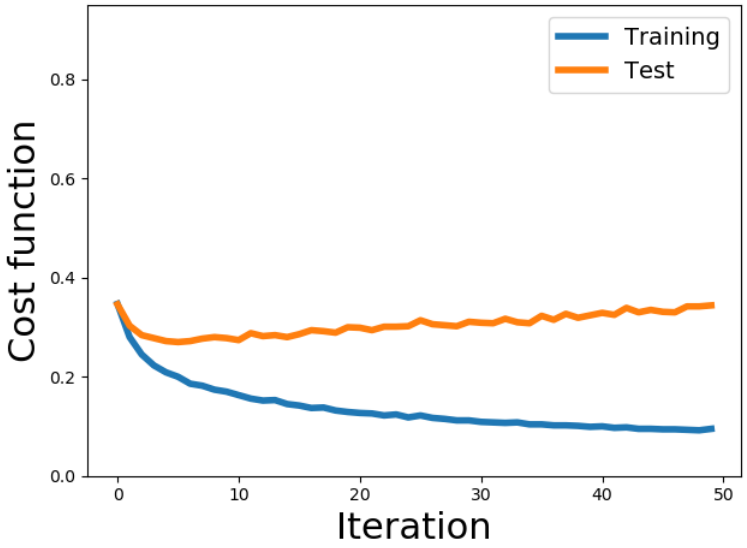




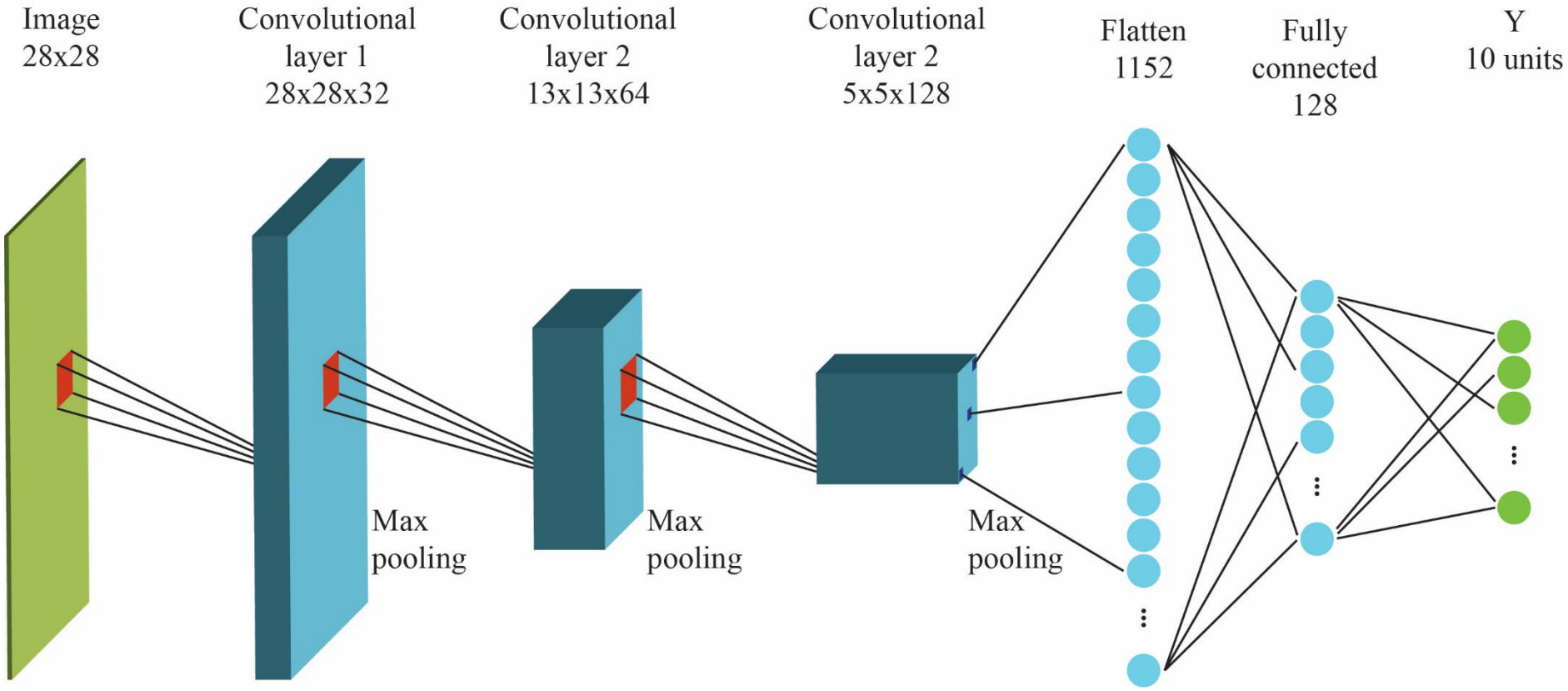
MNIST Fashion: 2-layer network + Dropout

- Dropout layer is applied to the convolutional layer
- Dropout prevents the overfitting and increases the test accuracy



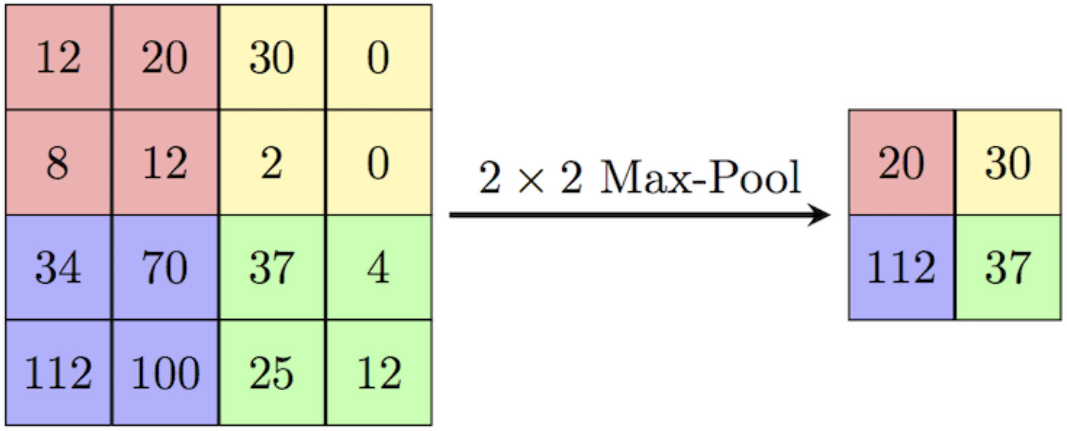


Vgg
(Oxford Visual Geometry Group)



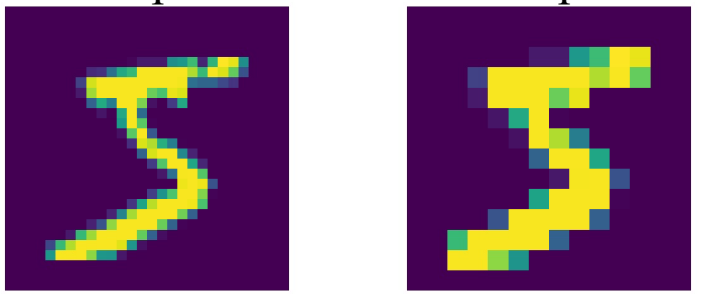
The receptive fields of the convolutional layers are trained simultaneously for all positions

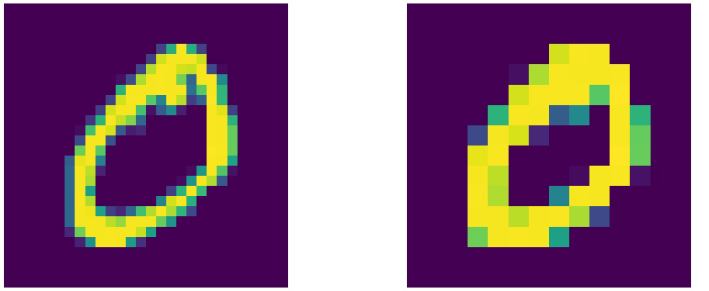
Down sampling through Max(imum) pooling



Input
28x28 pixels

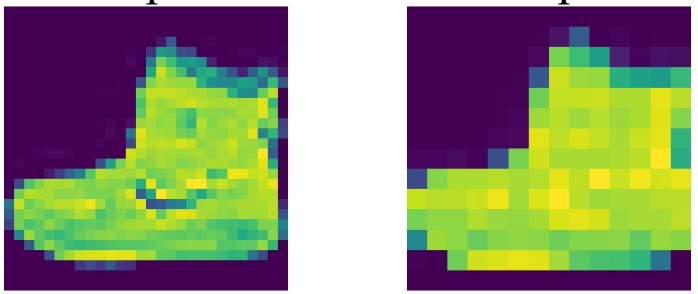
After max-pooling
14x14 pixels

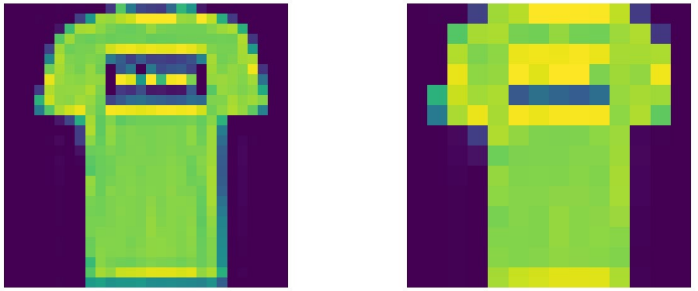




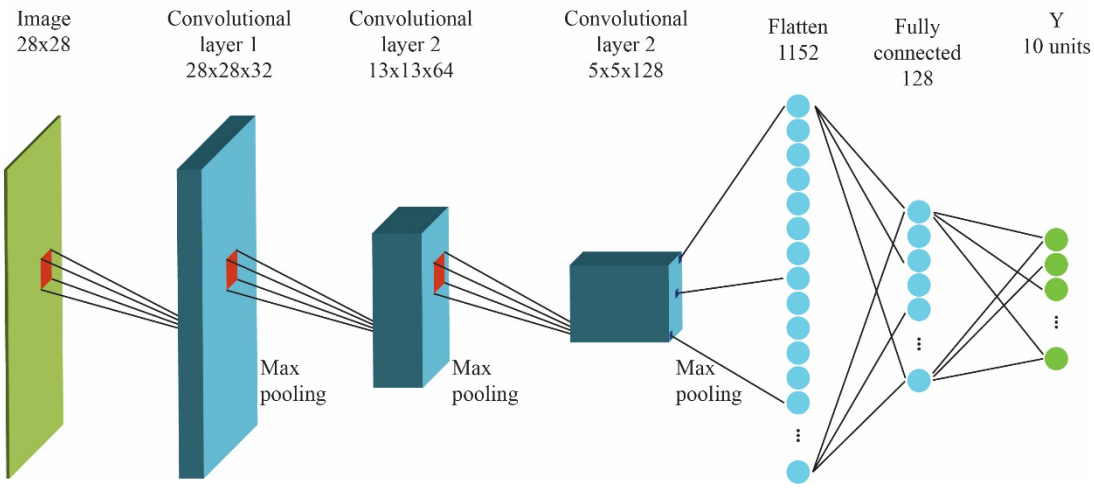
Input
28x28 pixels

After max-pooling
14x14 pixels



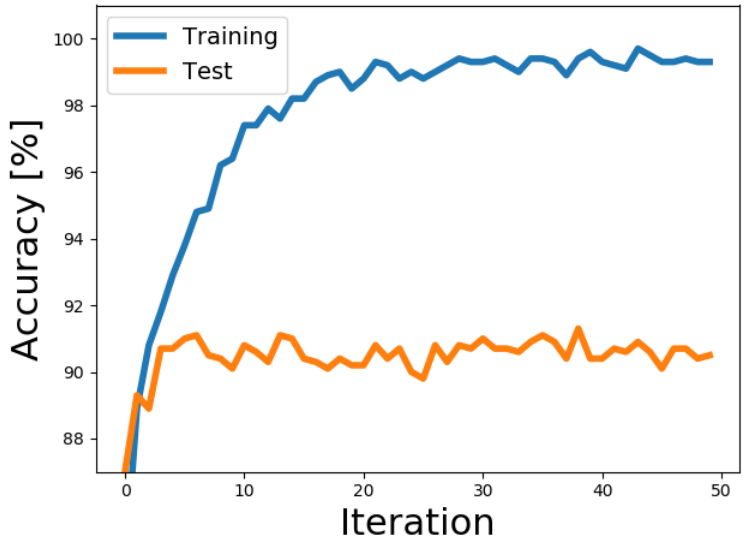


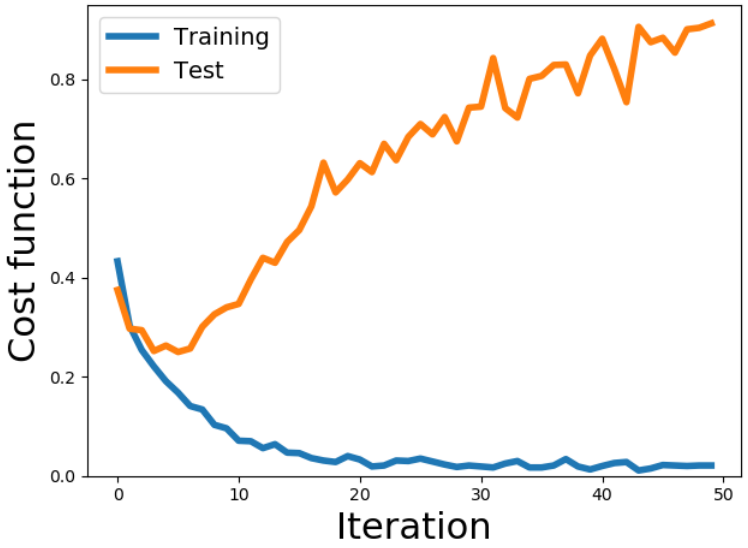
MNIST Fashion: Deep neural networks (Vgg: Visual geometry group, Oxford)



60000 Training samples
10000 Test samples

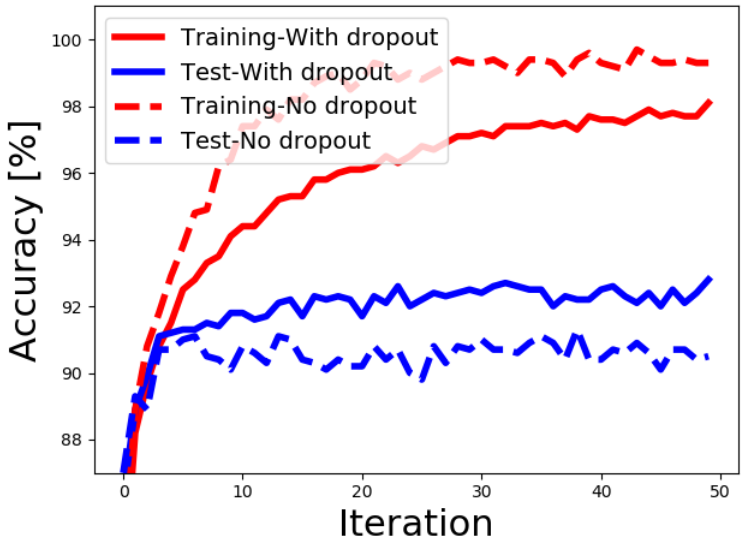
- Optimizer: gradient descent
- Rectified linear activation function in the middle layer
- Softmax activation function in the final layer
- Receptive field : 3x3
- Batch size : 1000

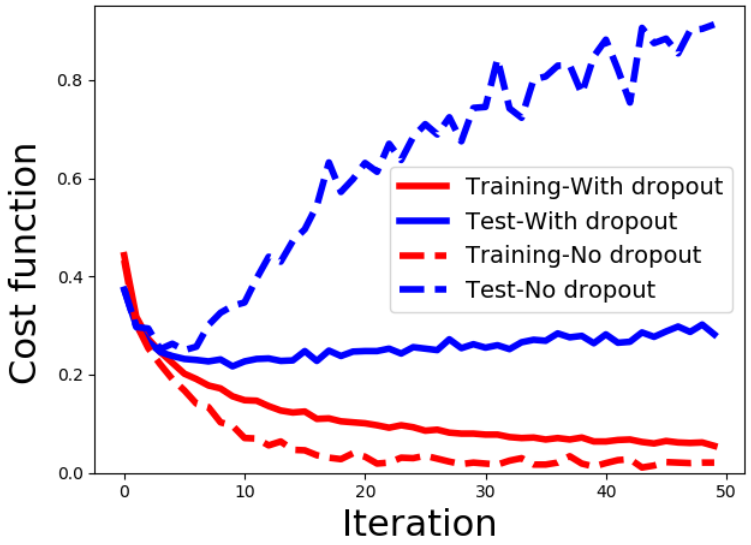




MNIST Fashion: Deep neural networks + Dropout

Drop-out is applied to ALL the hidden units





Network Type	Dropout	Training Accuracy	Test Accuracy
2 layer	No	100	89.7
2 layer	Yes	96.7	90.1
Vgg	No	99.3	90.7
Vgg	Yes	98.1	92.2