Convolutional neural networks

CE/CZ4042 – Tutorial 7

1. The first hidden layer of a convolution neural network (CNN) has a convolution layer consisting of two feature maps with filters $\boldsymbol{w_1}$ and $\boldsymbol{w_2}$ and biases = 0.1, and neurons having sigmoid activation functions, and a pooling layer with a pooling windows of size 2x2:

$$\boldsymbol{w_1} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \text{ and } \boldsymbol{w_2} = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

The input layer is of 6x6 size and receives an input image $\boldsymbol{I}$:

$$\boldsymbol{I} = \begin{pmatrix} 0.7 & 0.1 & 0.2 & 0.3 & 0.3 & 05 \\ 0.8 & 0.1 & 0.3 & 0.5 & 0.1 & 0.0 \\ 1.0 & 0.2 & 0.0 & 0.3 & 0.2 & 0.7 \\ 0.8 & 0.1 & 0.5 & 0.6 & 0.3 & 0.4 \\ 0.1 & 0.0 & 0.9 & 0.3 & 0.3 & 0.2 \\ 1.0 & 0.1 & 0.4 & 0.5 & 0.2 & 0.8 \end{pmatrix}$$

   a. Find the outputs at the first convolution layer if
      i. padding = VALID and strides = $(1, 1)$.
      ii. padding = SAME and strides = $(2, 2)$.
   b. Find the outputs at the first pooling layer for Part (a), assuming strides of $(2, 2)$ and pooling is
      i. max pooling
      ii. mean pooling

$$\boldsymbol{w}_1 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \boldsymbol{w}_2 = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \text{ and } b = 0.1.$$

$$\boldsymbol{I} = \begin{pmatrix} 0.7 & 0.1 & 0.2 & 0.3 & 0.3 & 05 \\ 0.8 & 0.1 & 0.3 & 0.5 & 0.1 & 0.0 \\ 1.0 & 0.2 & 0.0 & 0.3 & 0.2 & 0.7 \\ 0.8 & 0.1 & 0.5 & 0.6 & 0.3 & 0.4 \\ 0.1 & 0.0 & 0.9 & 0.3 & 0.3 & 0.2 \\ 1.0 & 0.1 & 0.4 & 0.5 & 0.2 & 0.8 \end{pmatrix}$$

Synaptic inputs to the feature map with filter $\boldsymbol{w}_1$:
$\boldsymbol{u}_1 = Conv(\boldsymbol{I}, \boldsymbol{w}_1) + b_1$

VALID padding:
$\boldsymbol{u}_1(1,1) = 0.7\times0 + 0.1\times1 + 0.2\times1 + 0.8\times1 + 0.1\times0 + 0.3\times1 + 1.0\times1 + 0.2\times1 + 0.0\times0 + 0.1 = 2.7$
$\boldsymbol{u}_1(1,2) = 0.1\times0 + 0.2\times1 + 0.3\times1 + 0.1\times1 + 0.3\times0 + 0.5\times1 + 0.2\times1 + 0.0\times1 + 0.3\times0 + 0.1 = 1.4$
$\boldsymbol{u}_1(2,1) = 0.8\times0 + 0.1\times1 + 0.3\times1 + 1.0\times1 + 0.2\times0 + 0.0\times1 + 0.8\times1 + 0.1\times1 + 0.5\times0 + 0.1 = 2.4$

$$\boldsymbol{u}_1 = \begin{pmatrix} 2.7 & 1.4 & 1.4 & 1.9 \\ 2.4 & 2.0 & 2.0 & 2.1 \\ 1.7 & 2.0 & 2.6 & 2.6 \\ 2.8 & 2.0 & 3.1 & 2.0 \end{pmatrix}$$

Similarly, $\boldsymbol{u}_2 = Conv(\boldsymbol{I}, \boldsymbol{w}_2) + b_2 = \begin{pmatrix} 0.3 & 0.0 & -0.2 & 2.0 \\ 0.3 & 0.4 & 0.6 & 0.8 \\ -0.1 & 0.8 & 1.1 & -0.3 \\ 0.2 & -0.1 & -0.2 & 0.3 \end{pmatrix}$

Feature maps at the convolutional layer

$$\boldsymbol{y}_1 = f(\boldsymbol{u}_1) = \frac{1}{1 + e^{-\boldsymbol{u}_1}} = \begin{pmatrix} 0.94 & 0.8 & 0.8 & 0.87 \\ 0.92 & 0.88 & 0.88 & 0.89 \\ 0.85 & 0.88 & 0.93 & 0.93 \\ 0.94 & 0.88 & 0.96 & 0.88 \end{pmatrix}$$

$$\boldsymbol{y}_2 = f(\boldsymbol{u}_2) = \frac{1}{1 + e^{-\boldsymbol{u}_2}} = \begin{pmatrix} 0.57 & 0.5 & 0.45 & 0.55 \\ 0.57 & 0.60 & 0.65 & 0.69 \\ 0.48 & 0.69 & 0.75 & 0.43 \\ 0.55 & 0.48 & 0.45 & 0.57 \end{pmatrix}$$

Feature maps at the convolutional layer:

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 0.94 & 0.8 & 0.8 & 0.87 \\ 0.92 & 0.88 & 0.88 & 0.89 \\ 0.85 & 0.88 & 0.93 & 0.93 \\ 0.94 & 0.88 & 0.96 & 0.88 \end{pmatrix} \\ \begin{pmatrix} 0.57 & 0.5 & 0.45 & 0.55 \\ 0.57 & 0.60 & 0.65 & 0.69 \\ 0.48 & 0.69 & 0.75 & 0.43 \\ 0.55 & 0.48 & 0.45 & 0.57 \end{pmatrix} \end{pmatrix}$$

Pooling VALID 2x2 strides = 2:

Max-pooling:

$$o = \begin{pmatrix} \begin{pmatrix} 0.94 & 0.89 \\ 0.94 & 0.96 \end{pmatrix} \\ \begin{pmatrix} 0.60 & 0.69 \\ 0.69 & 0.75 \end{pmatrix} \end{pmatrix}$$

Mean-pooling:

$$p_{ave} = \begin{pmatrix} \begin{pmatrix} 0.88 & 0.86 \\ 0.89 & 0.92 \end{pmatrix} \\ \begin{pmatrix} 0.56 & 0.58 \\ 0.55 & 0.55 \end{pmatrix} \end{pmatrix}$$

$$\boldsymbol{w}_1 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \boldsymbol{w}_2 = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \text{ and } b = 0.1.$$

$$\boldsymbol{I} = \begin{pmatrix} 0.7 & 0.1 & 0.2 & 0.3 & 0.3 & 05 \\ 0.8 & 0.1 & 0.3 & 0.5 & 0.1 & 0.0 \\ 1.0 & 0.2 & 0.0 & 0.3 & 0.2 & 0.7 \\ 0.8 & 0.1 & 0.5 & 0.6 & 0.3 & 0.4 \\ 0.1 & 0.0 & 0.9 & 0.3 & 0.3 & 0.2 \\ 1.0 & 0.1 & 0.4 & 0.5 & 0.2 & 0.8 \end{pmatrix}$$

Synaptic inputs to the feature map with filter $\boldsymbol{w}_1$:
$\boldsymbol{u}_1 = Conv(\boldsymbol{I}, \boldsymbol{w}_1) + b_1$

SAME padding, strides = (2, 2):
$\boldsymbol{u}_1(1,1) = 0.7\times0 + 0.1\times1 + 0.2\times1 + 0.8\times1 + 0.1\times0 + 0.3\times1 + 1.0\times1 + 0.2\times1 + 0.0\times0 + 0.1 = 2.7$
$\boldsymbol{u}_1(1,2) = 0.2\times0 + 0.3\times1 + 0.3\times1 + 0.3\times1 + 0.5\times0 + 0.1\times1 + 0.0\times1 + 0.3\times1 + 0.2\times0 + 0.1 = 1.4$
$\boldsymbol{u}_1(1,3) = 0.3\times0 + 0.5\times1 + 0.0\times1 + 0.1\times1 + 0.0\times0 + 0.0\times1 + 0.2\times1 + 0.7\times1 + 0.0\times0 + 0.1 = 1.6$
$\boldsymbol{u}_1(2,1) = 1.0\times0 + 0.2\times1 + 0.0\times1 + 0.8\times1 + 0.1\times0 + 0.5\times1 + 0.1\times1 + 0.0\times1 + 0.9\times0 + 0.1 = 1.7$

$$\boldsymbol{u}_1 = \begin{pmatrix} 2.7 & 1.4 & 1.6 \\ 1.7 & 2.6 & 1.6 \\ 2.4 & 1.3 & 0.5 \end{pmatrix}$$

Similarly, $\boldsymbol{u}_2 = Conv(\boldsymbol{I}, \boldsymbol{w}_2) + b_2 = \begin{pmatrix} 0.3 & -0.2 & 0.2 \\ -0.1 & 1.1 & -0.3 \\ -0.9 & -1.4 & -0.4 \end{pmatrix}$

Feature maps at the convolutional layer

$$\boldsymbol{y}_1 = f(\boldsymbol{u}_1) = \frac{1}{1 + e^{-\boldsymbol{u}_1}} = \begin{pmatrix} 0.94 & 0.8 & 0.83 \\ 0.85 & 0.93 & 0.83 \\ 0.92 & 0.79 & 0.63 \end{pmatrix}$$

$$\boldsymbol{y}_2 = f(\boldsymbol{u}_2) = \frac{1}{1 + e^{-\boldsymbol{u}_2}} = \begin{pmatrix} 0.57 & 0.45 & 0.55 \\ 0.48 & 0.75 & 0.43 \\ 0.29 & 0.20 & 0.40 \end{pmatrix}$$

Feature maps at the convolutional layer:

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 0.94 & 0.8 & 0.83 \\ 0.85 & 0.93 & 0.83 \\ 0.92 & 0.79 & 0.63 \end{pmatrix} \\ \begin{pmatrix} 0.57 & 0.45 & 0.55 \\ 0.48 & 0.75 & 0.43 \\ 0.29 & 0.20 & 0.40 \end{pmatrix} \end{pmatrix}$$

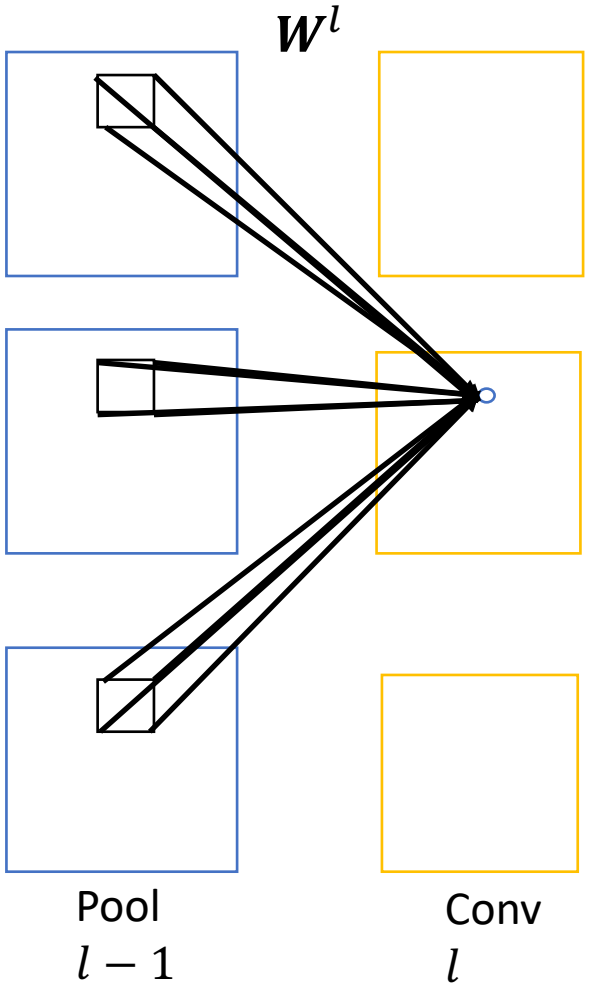
Pooling VALID 2x2 strides = 2:

Max-pooling:

$$o = \begin{pmatrix} (0.94) \\ (0.75) \end{pmatrix}$$

Mean-pooling:

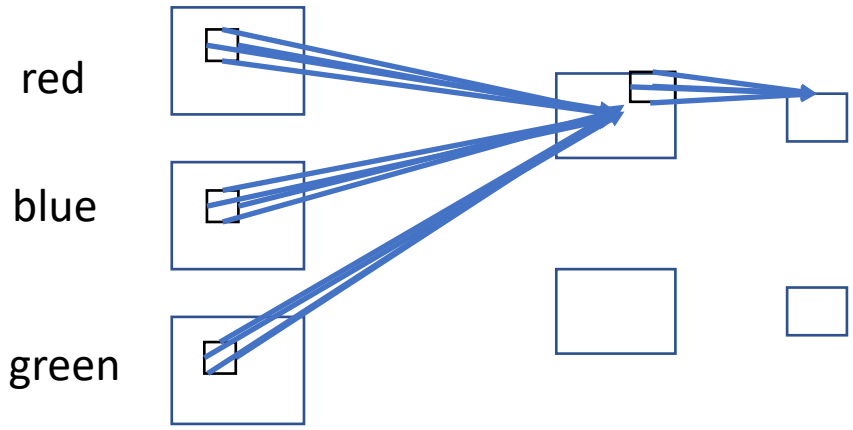
$$p_{ave} = \begin{pmatrix} (0.88) \\ (0.56) \end{pmatrix}$$



Weight vector in CNN is
a 4-dimensional tensor

```
u1 = tf.nn.conv2d(x, w, strides = [1, 1, 1, 1], padding = 'VALID') + b
u2 = tf.nn.conv2d(x, w, strides = [1, 2, 2, 1], padding = 'SAME') + b

x = [no_patterns, input_width, input_height, no_input_maps]
w = [ window_width, window_height, input_maps, output_maps]



y1 = tf.nn.sigmoid(u1)

u1_, y1_ = sess.run([u1, y1], {x: I.reshape([1, 6, 6, 1])})



o1 = tf.nn.max_pool(y1, ksize=[1, 2, 2, 1], strides = [1, 2, 2, 1], padding = 'VALID')
```

2. Given '3wolfmoon.jpg' color image of size 639×516.
    a. Initialize weights and biases of a convolutional layer with two kernels of size 9×9. Note that the input image is in color and has three channels.
    b. Display the feature maps at the convolution layer, assuming sigmoid activation functions. Use VALID padding and strides = 1.
    c. Display the outputs of a mean pooling layer with a pooling window size 5×5 and strides = 5.



3 input channels
3x639x516

Convolution Layer
Two filters
9x9x3x2
VALID
Stride = 1
631x508x2

Pooling Layer
Window size = (5, 5)
Strides = 5
VALID
126x101x2

Original image

Red

Blue

Green

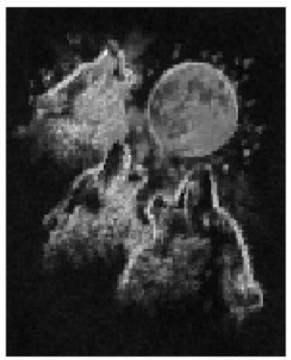
Original image

Convolved feature map 1

Convolved feature map 2

Original image

Pooled feature map 1



Pooled feature map 2

3. Design a CNN with one hidden layer to recognize digit images in MNIST database:

   http://yann.lecun.com/exdb/mnist/

   The convolution layer consists of 25 filters of dimensions 9x9 and the pooling layer has a pooling window size 4x4. Assume VALID padding and default strides for both convolution and pooling layer. Train the network with mini batch gradient decent learning with learning factor $\alpha = 10^{-3}$ and batch size = 128.

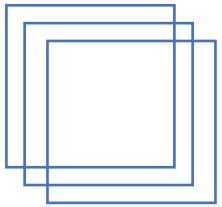
   Plot

   a. The training and test errors against learning epochs.
   b. Final filter weights
   c. Feature maps at the convolution and pooling layers for a representative test pattern
   d. Repeat training by introducing decay parameter $\beta = 10^{-6}$ and momentum term with $\gamma = 0.5$, and compare the learning curves

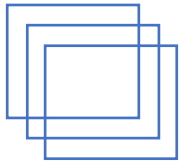
MNIST database: 28x28 =784 inputs
Training set = 12000 images
Testing set = 2000 images

Input pixel values were normalized to
mean = 0.0, s.d. = 1.0

input image
28x28



Convolution 1
15 filters
9x9 size
Size = 25x20x20



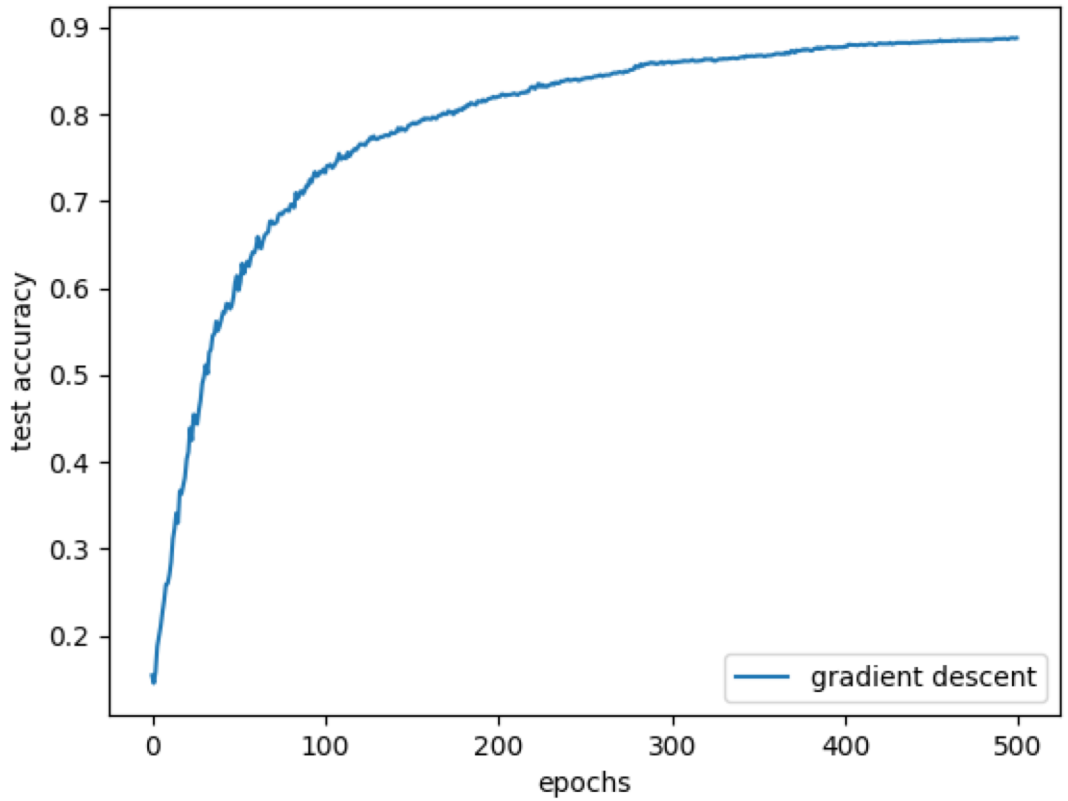
Pooling 1
4x4 pooling
25x5x5

Output
Softmax layer
10 neurons

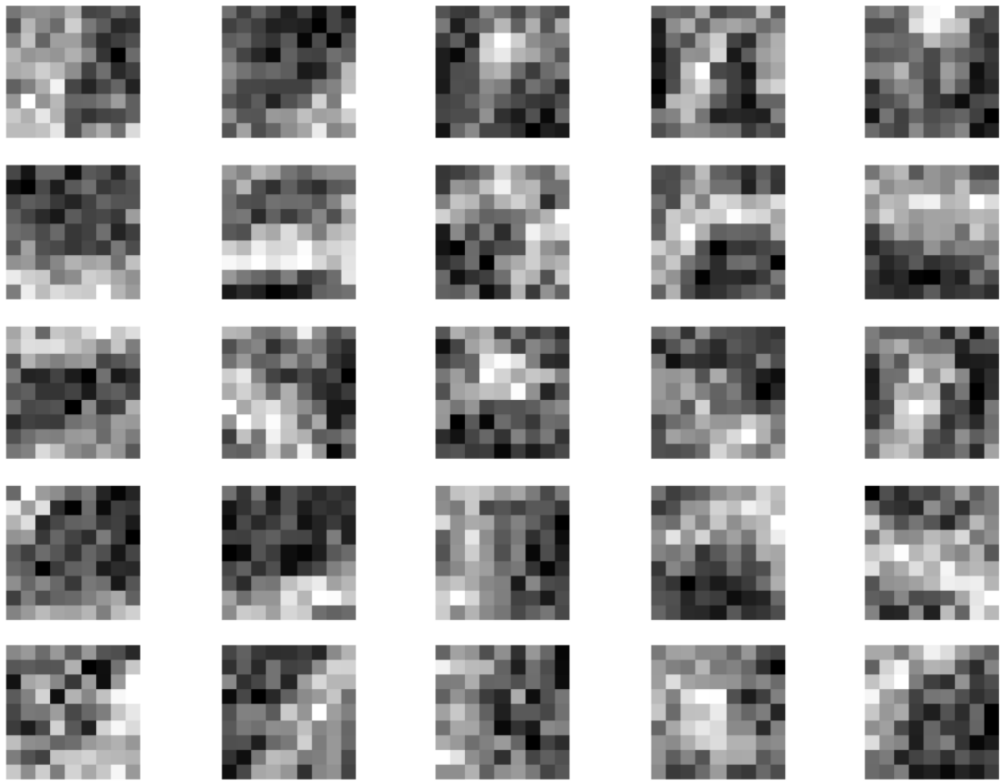
Mini-batch gradient descent
Learning rate = 10^{-3}

200 epochs
VALID padding
Default strides

Learning curves



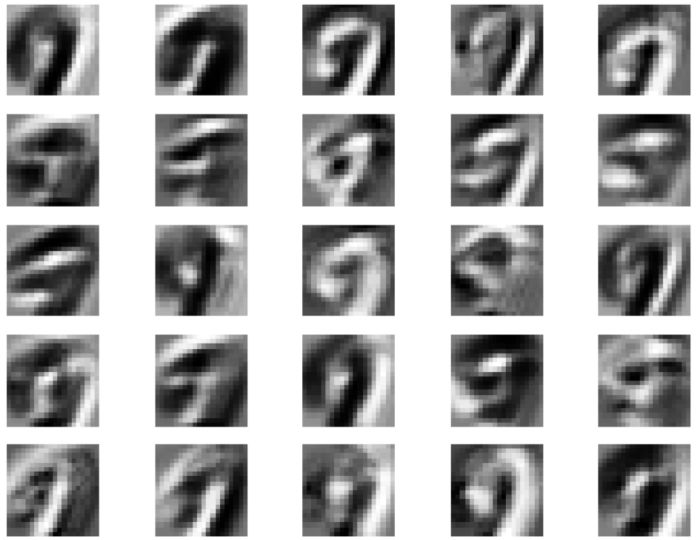
Features learned by the first convolution layer



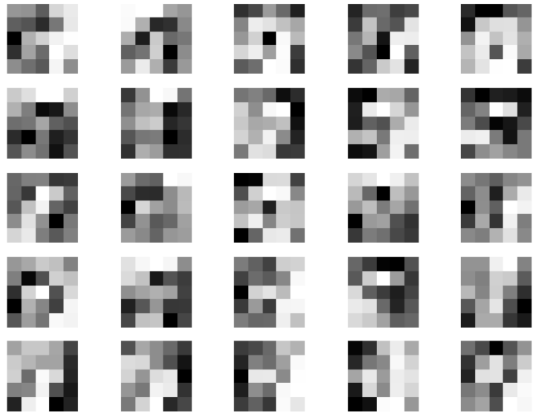
25 x 9 x 9

Input image

Feature maps at first convolution layer 25x20x20



Feature maps at first pooling layer 25x5x5



Gradient descent:

$$\boldsymbol{W} = \boldsymbol{W} - \alpha \nabla_{\boldsymbol{W}} J$$

Weight decay (regularization):

$$J_1(\boldsymbol{W}, \boldsymbol{b}) = J(\boldsymbol{W}, \boldsymbol{b}) + \beta_2 \sum_{ij} \left(w_{ij}\right)^2$$

Weight decay:

$$\boldsymbol{W} = \boldsymbol{W} - \alpha(\nabla_{\boldsymbol{W}} J + \beta \boldsymbol{W})$$

Sgd with momentum:

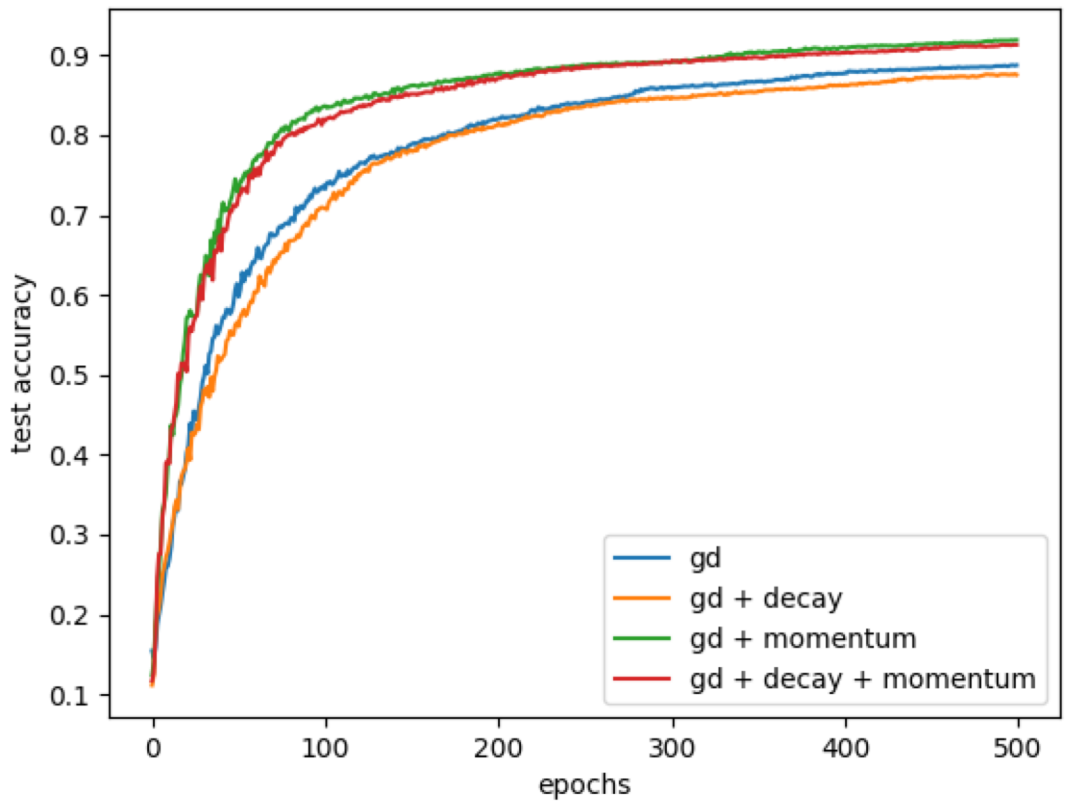
$$\boldsymbol{V} = \gamma \boldsymbol{V} - \alpha \nabla_{\boldsymbol{W}} J$$
$$\boldsymbol{W} = \boldsymbol{W} + \boldsymbol{V}$$

Sgd with decay and momentum:

$$\boldsymbol{V} = \gamma \boldsymbol{V} - \alpha(\nabla_{\boldsymbol{W}} J + \beta \boldsymbol{W})$$
$$\boldsymbol{W} = \boldsymbol{W} + \boldsymbol{V}$$

```
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(labels=y_,logits=y_conv)
cross_entropy = tf.reduce_mean(cross_entropy)

regularization = tf.nn.l2_loss(W_conv) + tf.nn.l2_loss(W_fc)

train_step1 = tf.train.GradientDescentOptimizer(alpha).minimize(cross_entropy)
train_step2 = tf.train.GradientDescentOptimizer(alpha).minimize(cross_entropy + beta*regularization)
train_step3 = tf.train.MomentumOptimizer(alpha, gamma).minimize(cross_entropy)
train_step4 = tf.train.MomentumOptimizer(alpha, gamma).minimize(cross_entropy + beta*regularization)
```

Learning parameter $\alpha = 10^{-3}$
Decay parameter $\beta = 10^{-6}$
Momentum $parame ter\ \gamma = 0.5$


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
test accuracy
0
100
200
300
400
500
epochs
gd
gd + decay
gd + momentum
gd + decay + momentum