# Convolutional Neural Networks

Carlos Zanini

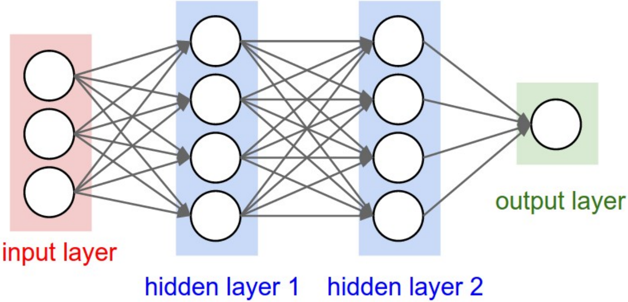
November 13, 2017

# Outline

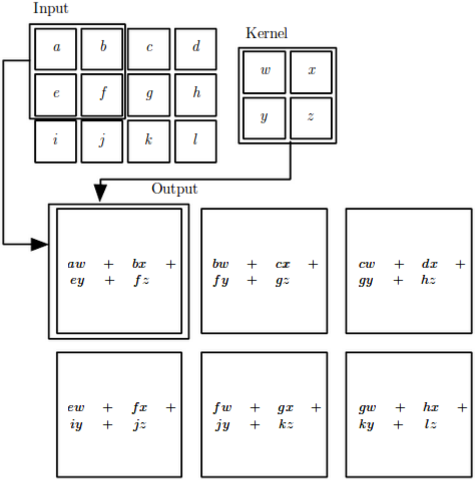
- Quick recap of Multilayer Perceptron (MLP)
- Basics of Convolutional Neural Networks (CNN)
- Relation between MLP and CNN
- Illustration with Lenet 5
- Implementation on MNIST in tensorflow.

# Multilayer perceptron: graphical representation

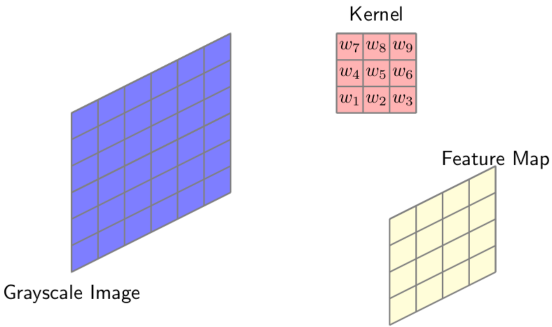
$$\boldsymbol{y}_i = \sigma(\boldsymbol{W}_3\boldsymbol{h}_2 + \boldsymbol{b}_3),$$
$$\boldsymbol{h}_2 = \sigma(\boldsymbol{W}_2\boldsymbol{h}_1 + \boldsymbol{b}_2),$$
$$\boldsymbol{h}_1 = \sigma(\boldsymbol{W}_1\boldsymbol{x}_i + \boldsymbol{b}_1).$$



# Convolution operation Kernel

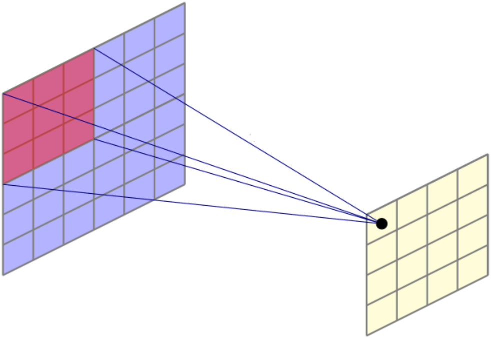


# Convolution operation Kernel

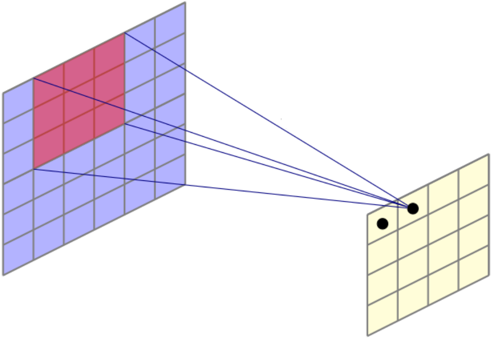



- Convolve image with kernel having weights $\mathbf{w}$ (learned by backpropagation)

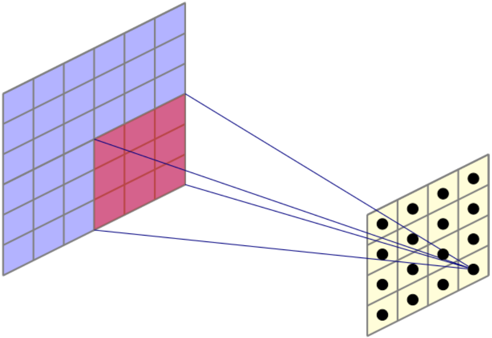
# Convolution operation Kernel



# Convolution operation Kernel



# Convolution operation Kernel



# Convolution operation Kernel

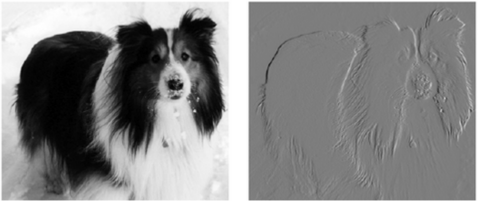


Figure: Example of edge detection: Kernel $= (-1, 1)$. The pixel on the right is subtracted by the one on the left.

# Convolution operation Kernel

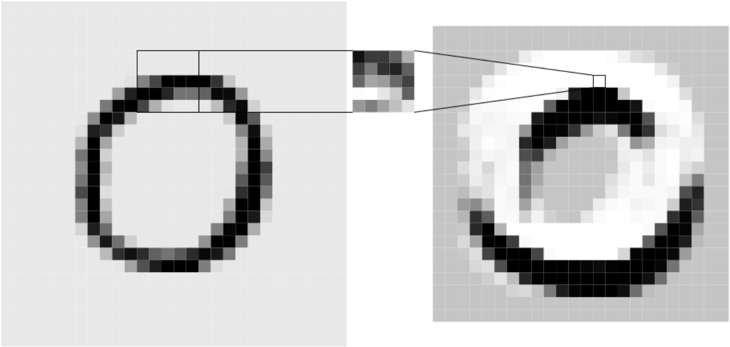


**Figure 3:** Input image (left), weight vector (center), and resulting feature map (right). The feature map is obtained by scanning the input image with a single neuron that has a local receptive field, as indicated. White represents -1, black represents +1.

# Zero padding

- Add borders of zeroes around the image
- Avoid convolutional stages to reduce image dimension

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | | | | | | | 0 |
| 0 | | | | | | | 0 |
| 0 | | | | | | | 0 |
| 0 | | | | | | | 0 |
| 0 | | | | | | | 0 |
| 0 | | | | | | | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Connection between CNN and neural networks

CNN is a specific type of MLP, with the following restrictions:

- Sparse interactions
- Parameter sharing
- Equivariant representations

# Sparse connections

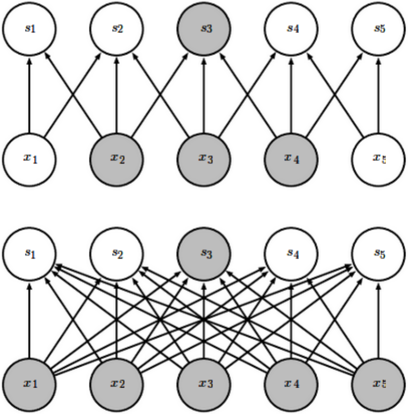



Figure 9.3: *Sparse connectivity, viewed from above:* We highlight one output unit, $s_3$, and also highlight the input units in $\boldsymbol{x}$ that affect this unit. These units are known as the **receptive field** of $s_3$. *(Top)*When $\boldsymbol{s}$ is formed by convolution with a kernel of width 3, only three inputs affect $s_3$. *(Bottom)*When $\boldsymbol{s}$ is formed by matrix multiplication, connectivity is no longer sparse, so all of the inputs affect $s_3$.

# Parameter sharing

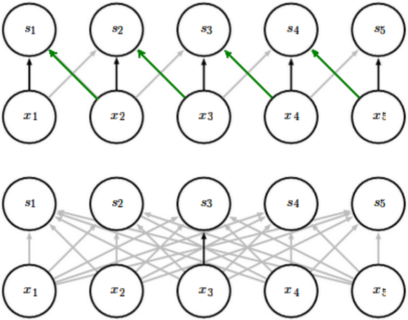
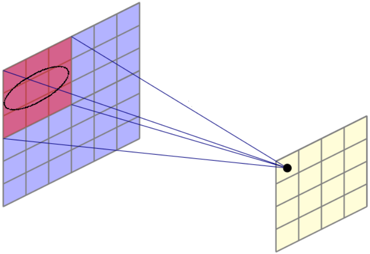



Figure 9.5: Parameter sharing: Black arrows indicate the connections that use a particular parameter in two different models. *(Top)*The black arrows indicate uses of the central element of a 3-element kernel in a convolutional model. Due to parameter sharing, this single parameter is used at all input locations. *(Bottom)*The single black arrow indicates the use of the central element of the weight matrix in a fully connected model. This model has no parameter sharing so the parameter is used only once.

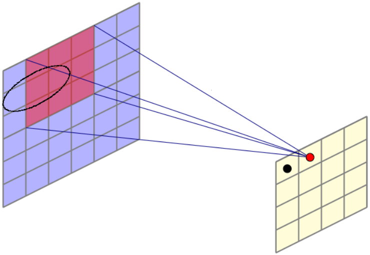
# Equivariance

Translated versions of the image are mapped to translated versions of the same feature:



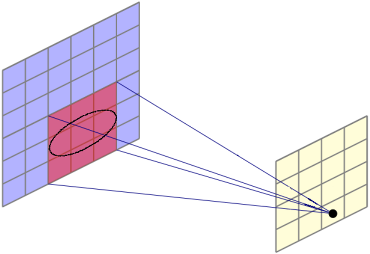
# Equivariance

Translated versions of the image are mapped to translated versions of the same feature:



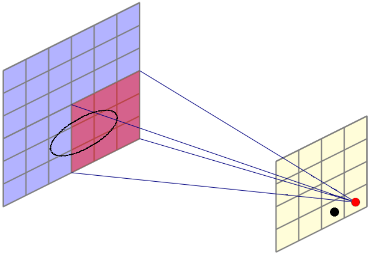
# Equivariance

Translated versions of the image are mapped to translated versions of the same feature:



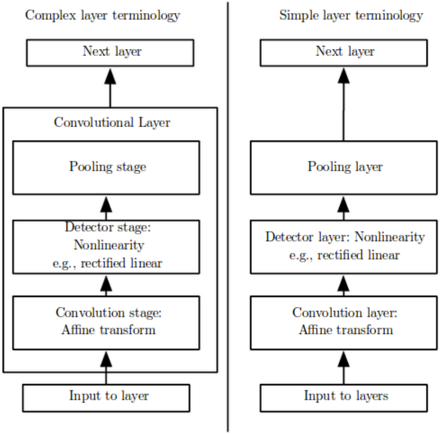
# Equivariance

Translated versions of the image are mapped to translated versions of the same feature:



# The 3 stages of convolutional layers

1. Convolution stage
2. Detection stage: Pointwise non-linear function (e.g. RELU, sigmoid, tanh)
3. Pooling stage: Summary statistic of nearby pixels (e.g, max, avg, L2-norm)



# Pooling layer

- Progressively reduce the spatial size of the representation
- Reduce the amount of parameters and computation in the network
- Control overfitting.

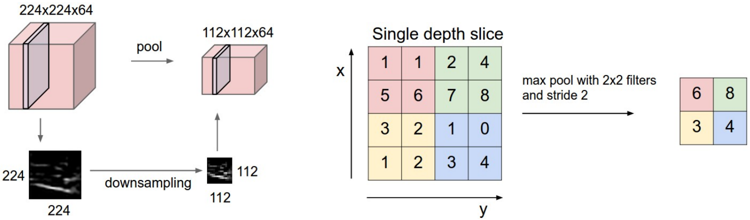
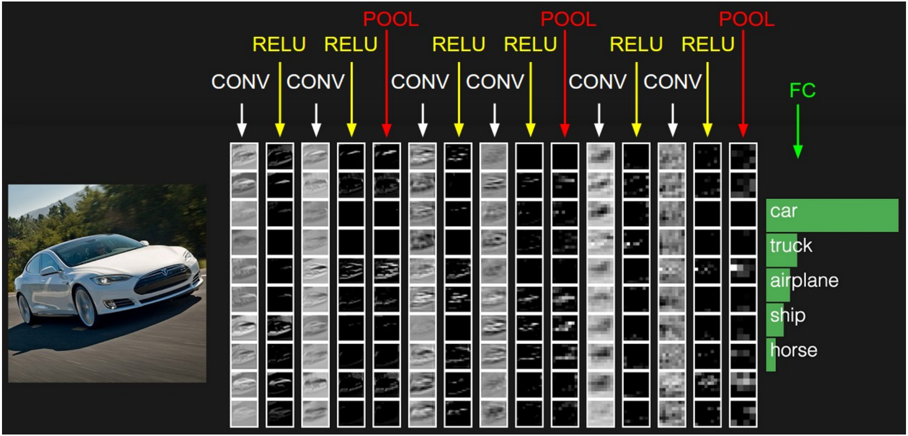
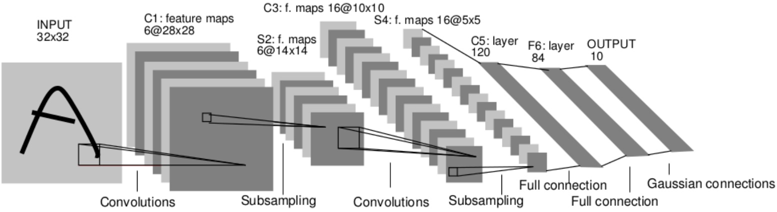



Figure: Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. Filter size 2, stride 2 into output. Max pooling, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

# Lenet 5 - Architecture




1. Convolutional kernels: 5x5, stride 1
2. Average pooling kernels: 2x2, stride 2

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | X | | | | X | X | X | | | X | X | X | X | | X | X |
| 1 | X | X | | | | X | X | X | | | X | X | X | X | | X |
| 2 | X | X | X | | | | X | X | X | | | X | | X | X | X |
| 3 | | X | X | X | | | X | X | X | X | | | X | | X | X |
| 4 | | | X | X | X | | | X | X | X | X | | X | X | | X |
| 5 | | | | X | X | X | | | X | X | X | X | | X | X | X |

TABLE I
EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

# Lenet-5 - Filtered images (after training)

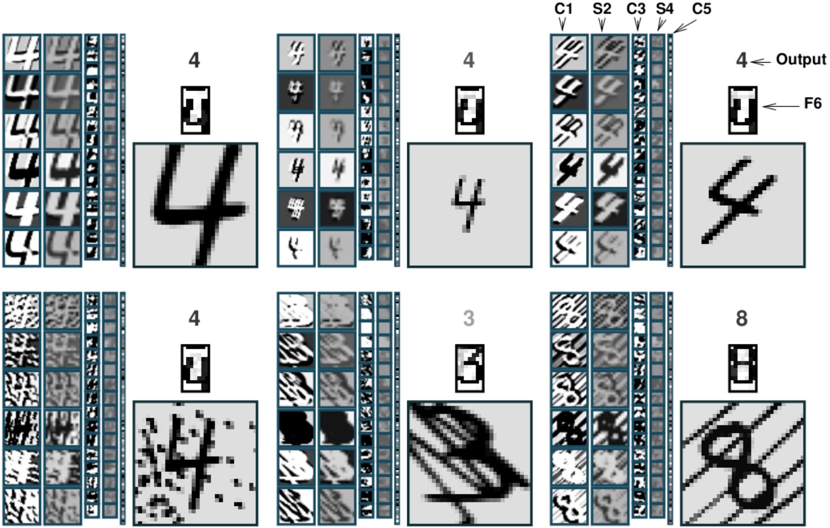



Fig. 8. The 82 test patterns misclassified by LeNet-5. Below each image is displayed the correct answers (left) and the network answer (right). These errors are mostly caused either by genuinely ambiguous patterns, or by digits written in a style that are under-represented in the training set.

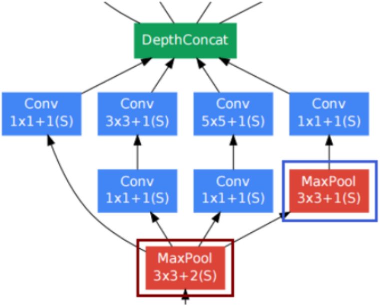
# Implementation of CNN

- 2 Convolutional layers: $2\times($ conv. $+$ activ. $+$ pooling)
    1. Convolutional stage: dimension 5x5, stride 1, zero padding
    2. Activation stage: $max(0, x)$ (RELU)
    3. Max pooling: dimension 2x2, stride 2
- Convolutional layers with 32 and 64 different feature maps, respectively.
- 2 Fully connected layers with 1024 hidden units each.
- Softmax output layer.

# Misclassifications

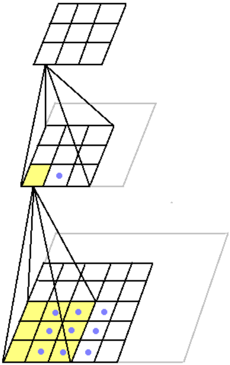
Total 93 misclassifications out of 10000 test examples

# Inception: Going Deeper With Convolutions (Szegedy, et al., 2015)




# Very deep neural networks for large scale image recognition (Simonyan and Zisserman 2015)

- Stack 2 3x3 filters to get a 5x5 receptive field (no pooling between)
- Stack 3 3x3 filters to get a 7x7 receptive field (no pooling between)
- $\vdots$



# Rethinking the inception architecture for computer vision (Szegedy, et al., 2016)

- Combine the ideas in both articles