UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Fundamentals of Convolutional Neural Networks (CNN)

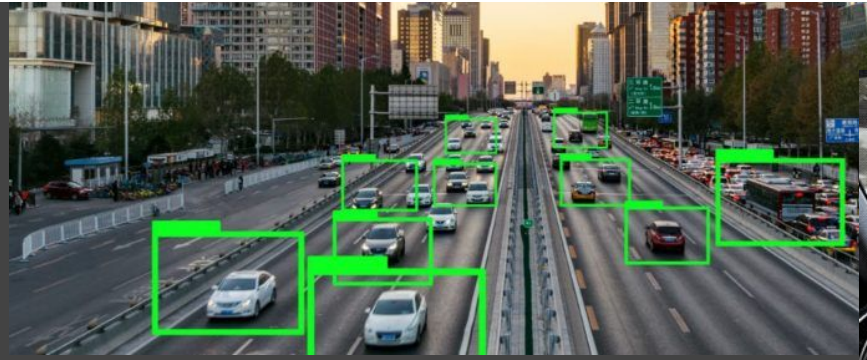
Fundamentals of CNN

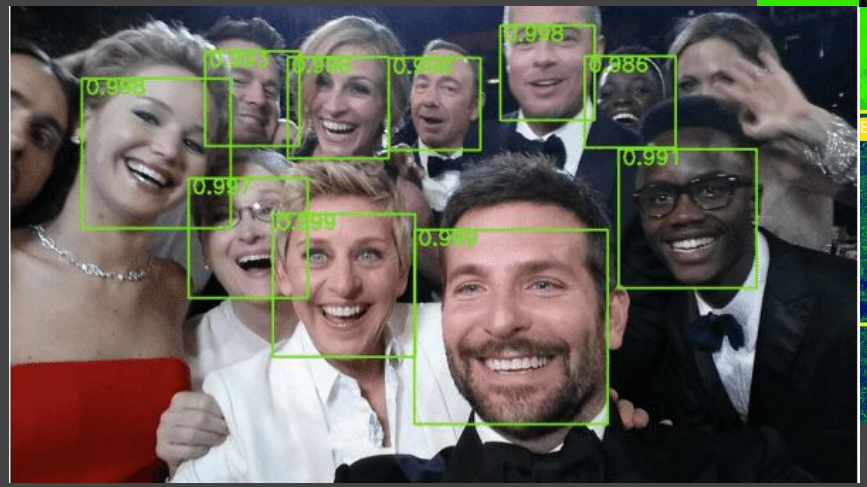
01
Introduction
MLP vs CNN

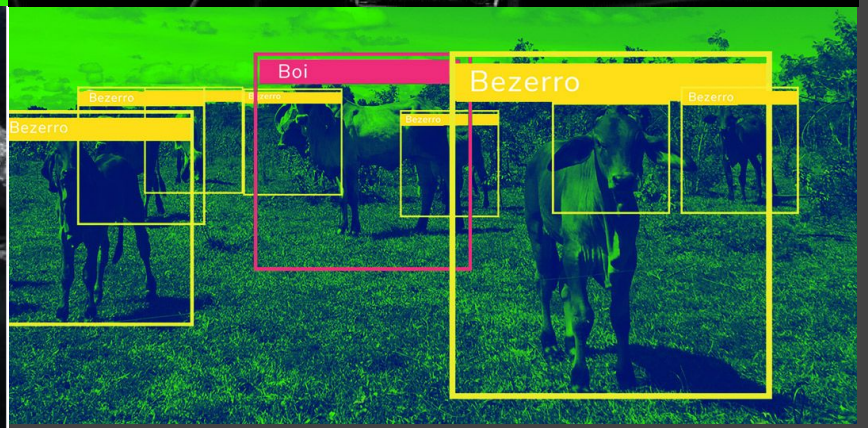
02
CNN Layers
Convolution Layer, Pooling Layer and Fully Connected Layer

03
Case Study: signs dataset
Baseline based on MLP vs CNN
Architecture and extentions

Computer vision is the automated extraction of information from images







J. Physiol. (1959) 147, 226–238

SINGLE UNIT ACTIVITY IN STRIATE CORTEX OF UNRESTRAINED CATS

By D. H. HUBEL*

From the Department of Neurophysiology, Walter Reed Army Institute of Research, Walter Reed Army Medical Center, Washington 12, D.C., U.S.A.

(*Received* 15 *December* 1958)

A beginning has recently been made in recording single neurone activity from animals with chronically implanted electrodes (Hubel, 1957*a*; Gusel'nikov, 1957; Ricci, Doane & Jasper, 1957; Strumwasser, 1958). These methods eliminate anaesthetics, paralysing drugs, brain-stem lesions, and other acute experimental procedures. They make it possible to record electrical events in the higher central nervous system with the animal in a normal state, and to correlate these electrical events with such variables as waking state, attention, learning, and motor activity.

The present paper describes a method for unit recording from the cortex of unanaesthetized, unrestrained cats, and presents some observations from the striate cortex. The objectives have been (1) to observe maintained unit activity under various conditions such as sleep and wakefulness, and (2) to find for each unit the natural stimuli which most effectively influence firing. Of 400 units observed, some 200 are presented here because of their common characteristics. Since there is reason to believe that the remaining 200 units were afferent fibres from the lateral geniculate nucleus, these will be described in a separate paper. A preliminary account of some of this work has been given elsewhere (Hubel, 1958).

METHODS

Unit recordings in unrestrained animals were made with micro-electrodes held in a positioner which was anchored rigidly to the skull during recordings, and removed between recordings

J. Physiol. (1968), **195**, *pp.* 215–243
With 3 *plates and* 14 *text-figures*
Printed in Great Britain

RECEPTIVE FIELDS AND FUNCTIONAL ARCHITECTURE OF MONKEY STRIATE CORTEX

By D. H. HUBEL and T. N. WIESEL

From the Department of Physiology, Harvard Medical School, Boston, Mass., U.S.A.

(*Received* 6 *October* 1967)

SUMMARY

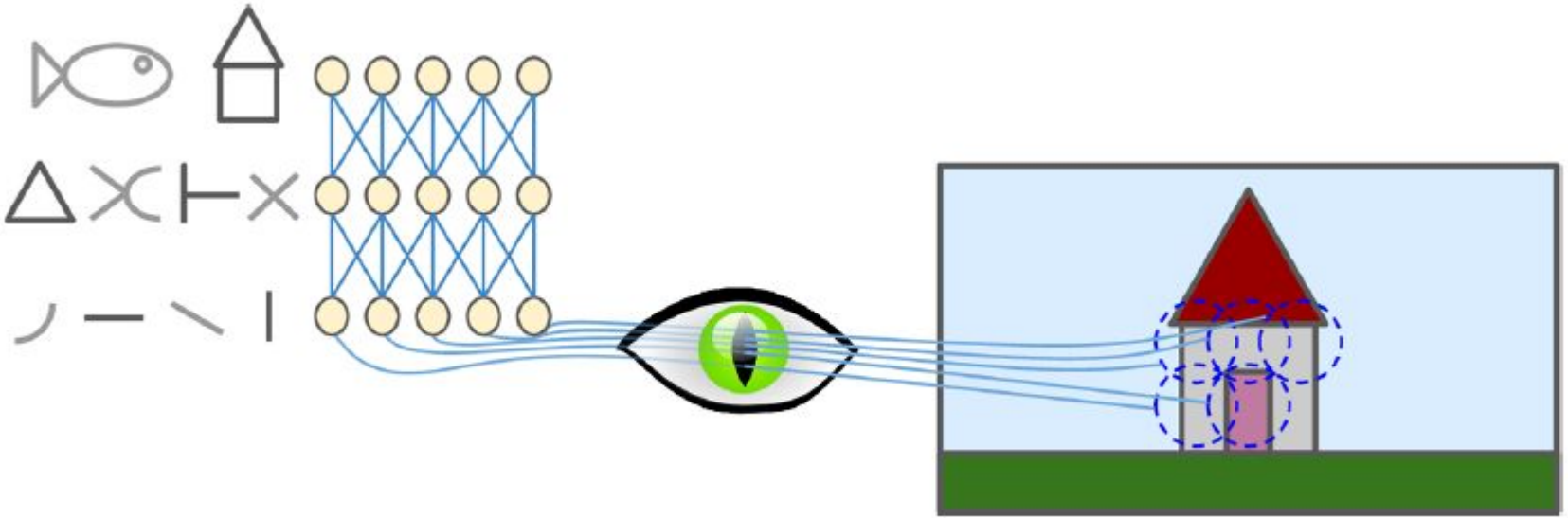
1. The striate cortex was studied in lightly anaesthetized macaque and spider monkeys by recording extracellularly from single units and stimulating the retinas with spots or patterns of light. Most cells can be categorized as simple, complex, or hypercomplex, with response properties very similar to those previously described in the cat. On the average, however, receptive fields are smaller, and there is a greater sensitivity to changes in stimulus orientation. A small proportion of the cells are colour coded.

2. Evidence is presented for at least two independent systems of columns extending vertically from surface to white matter. Columns of the first type contain cells with common receptive-field orientations. They are similar to the orientation columns described in the cat, but are probably smaller in cross-sectional area. In the second system cells are aggregated into columns according to eye preference. The ocular dominance columns are larger than the orientation columns, and the two sets of boundaries seem to be independent.

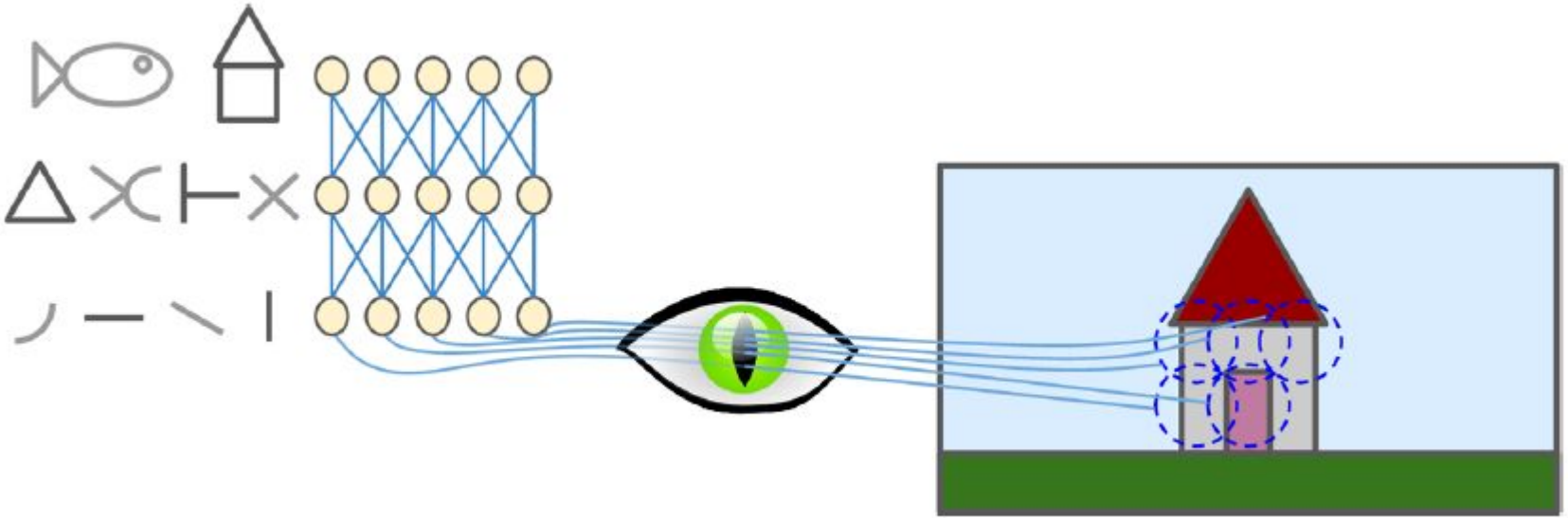
3. There is a tendency for cells to be grouped according to symmetry of responses to movement; in some regions the cells respond equally well to the two opposite directions of movement of a line, but other regions contain a mixture of cells favouring one direction and cells favouring the other.

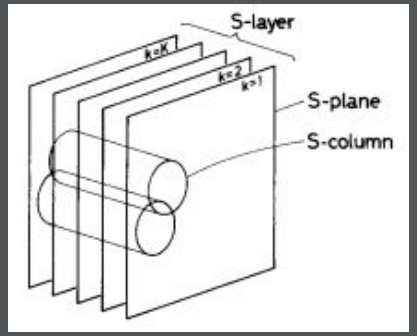
4. A horizontal organization corresponding to the cortical layering can also be discerned. The upper layers (II and the upper two-thirds of III) contain complex and hypercomplex cells, but simple cells are virtually absent. The cells are mostly binocularly driven. Simple cells are found deep in layer III, and in IV A and IV B. In layer IV B they form a large proportion of the population, whereas complex cells are rare. In layers IV A and IV B one finds units lacking orientation specificity; it is not

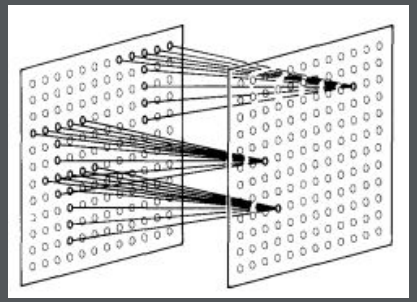
1. Hubel & Wiesel showed that many neurons in the visual cortex have a small local receptive field, meaning they react only to visual stimuli located in a limited region
2. The receptive fields of different neurons may overlap

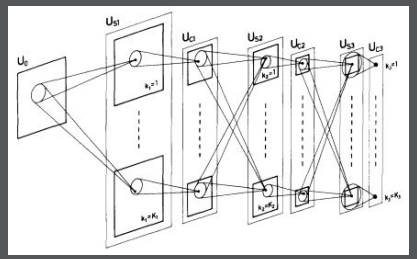


3. Some neurons react only to images of horizontal lines, while others react only to lines with different orientations
4. Some neurons have larger receptive fields, and they react to more complex patterns that are combinations of the lower-level patterns.
5. Higher-level neurons are based on the outputs of neighboring lower-level neurons









Biol. Cybernetics 36, 193–202 (1980)

Biological Cybernetics

Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position

Kunihiko Fukushima

NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Japan

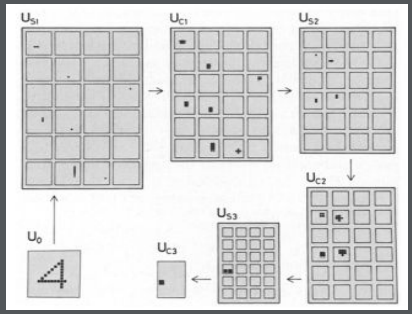
Abstract. A neural network model for a mechanism of visual pattern recognition is proposed in this paper. The network is self-organized by "learning without a teacher", and acquires an ability to recognize stimulus patterns based on the geometrical similarity (Gestalt) of their shapes without affected by their positions. This network is given a nickname "neocognitron". After completion of self-organization, the network has a structure similar to the hierarchy model of the visual nervous system proposed by Hubel and Wiesel. The network consists of an input layer (photoreceptor array) followed by a cascade connection of a number of modular structures, each of which is composed of two layers of cells connected in a cascade. The first layer of each module consists of "S-cells", which show characteristics similar to simple cells or lower order hypercomplex cells, and the second layer consists of "C-cells" similar to complex cells or higher order hypercomplex cells. The afferent synapses to each S-cell have plasticity and are modifiable. The network has an ability of unsupervised learning: We do not need any "teacher" during the process of self-organization, and it is only needed to present a set of stimulus patterns repeatedly to the input layer of the network. The network has been simulated on a digital computer. After repetitive presentation of a set of stimulus patterns, each stimulus pattern has become to elicit an output only from one of the C-cells of the last layer, and conversely, this C-cell has become selectively responsive only to that stimulus pattern. That is, none of the C-cells of the last layer responds to more than one stimulus pattern. The response of the C-cells of the last layer is not affected by the pattern's position at all. Neither is it affected by a small change in shape nor in size of the stimulus pattern.

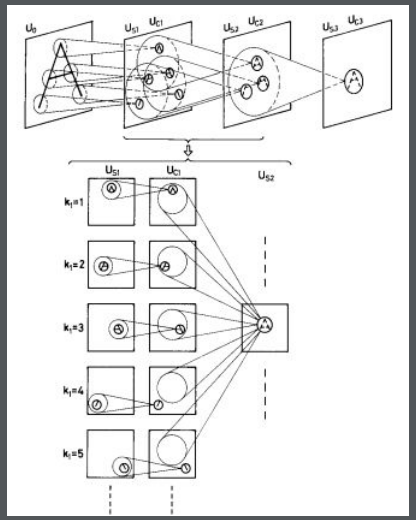
reveal it only by conventional physiological experiments. So, we take a slightly different approach to this problem. If we could make a neural network model which has the same capability for pattern recognition as a human being, it would give us a powerful clue to the understanding of the neural mechanism in the brain. In this paper, we discuss how to synthesize a neural network model in order to endow it an ability of pattern recognition like a human being.

Several models were proposed with this intention (Rosenblatt, 1962; Kabrisky, 1966; Giebel, 1971; Fukushima, 1975). The response of most of these models, however, was severely affected by the shift in position and/or by the distortion in shape of the input patterns. Hence, their ability for pattern recognition was not so high.

In this paper, we propose an improved neural network model. The structure of this network has been suggested by that of the visual nervous system of the vertebrate. This network is self-organized by "learning without a teacher", and acquires an ability to recognize stimulus patterns based on the geometrical similarity (Gestalt) of their shapes without affected by their position nor by small distortion of their shapes.

This network is given a nickname "neocognitron"[1], because it is a further extention of the "cognitron", which also is a self-organizing multilayered neural network model proposed by the author before (Fukushima, 1975). Incidentally, the conventional cognitron also had an ability to recognize patterns, but its response was dependent upon the position of the stimulus patterns. That is, the same patterns which were presented at different positions were taken as different patterns by the conventional cognitron. In the neocognitron proposed here, however, the response of the network is little affected by the position of the





Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Abstract—

Multilayer Neural Networks trained with the backpropagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Given an appropriate network architecture, Gradient-Based Learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns such as handwritten characters, with minimal preprocessing. This paper reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. Convolutional Neural Networks, that are specifically designed to deal with the variability of 2D shapes, are shown to outperform all other techniques.

Real-life document recognition systems are composed of multiple modules including field extraction, segmentation, recognition, and language modeling. A new learning paradigm, called Graph Transformer Networks (GTN), allows such multi-module systems to be trained globally using Gradient-Based methods so as to minimize an overall performance measure.

Two systems for on-line handwriting recognition are described. Experiments demonstrate the advantage of global training, and the flexibility of Graph Transformer Networks.

A Graph Transformer Network for reading bank check is also described. It uses Convolutional Neural Network character recognizers combined with global training techniques to provides record accuracy on business and personal checks. It is deployed commercially and reads several million checks per day.

Keywords— **Neural Networks, OCR, Document Recognition, Machine Learning, Gradient-Based Learning, Convolutional Neural Networks, Graph Transformer Networks, Finite State Transducers.**

Nomenclature

- GT Graph transformer.

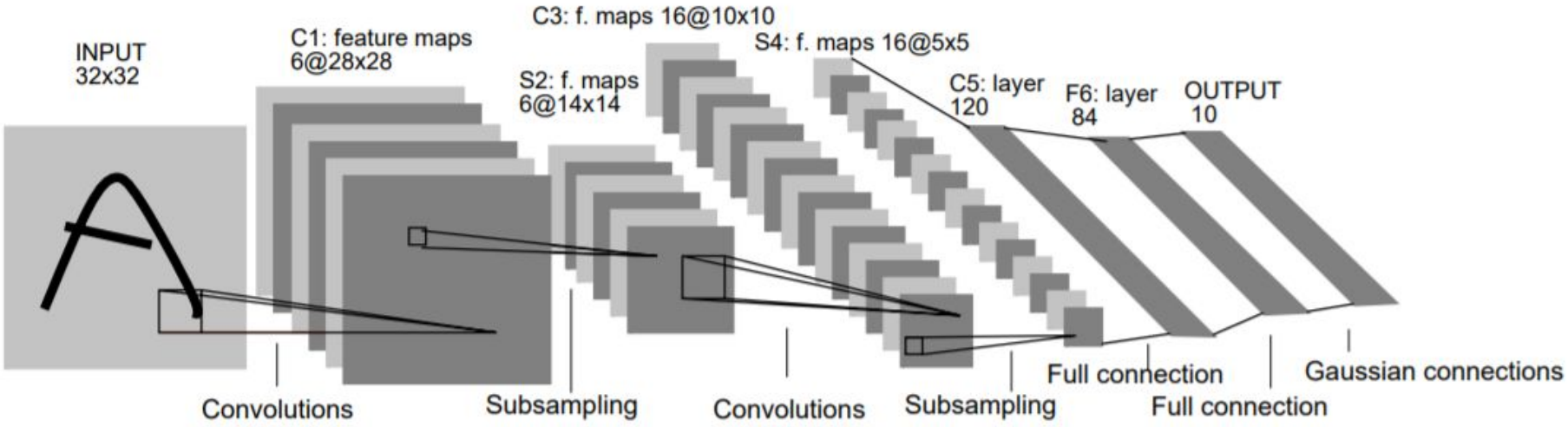
I. Introduction

Over the last several years, machine learning techniques, particularly when applied to neural networks, have played an increasingly important role in the design of pattern recognition systems. In fact, it could be argued that the availability of learning techniques has been a crucial factor in the recent success of pattern recognition applications such as continuous speech recognition and handwriting recognition.

The main message of this paper is that better pattern recognition systems can be built by relying more on automatic learning, and less on hand-designed heuristics. This is made possible by recent progress in machine learning and computer technology. Using character recognition as a case study, we show that hand-crafted feature extraction can be advantageously replaced by carefully designed learning machines that operate directly on pixel images. Using document understanding as a case study, we show that the traditional way of building recognition systems by manually integrating individually designed modules can be replaced by a unified and well-principled design paradigm, called *Graph Transformer Networks*, that allows training all the modules to optimize a global performance criterion.

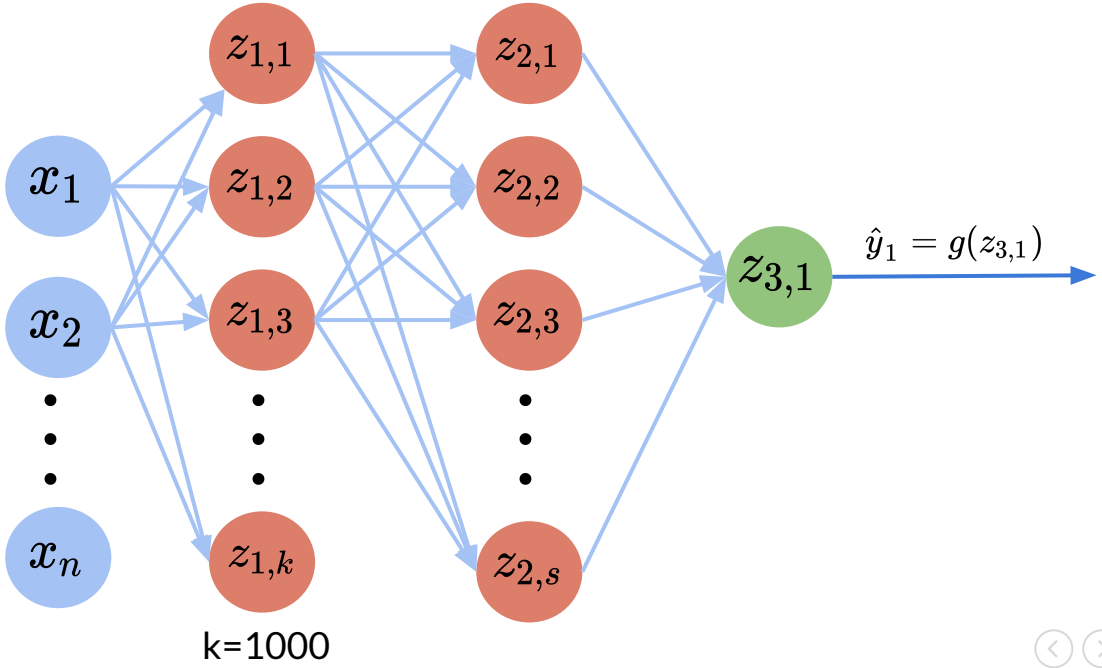
Since the early days of pattern recognition it has been known that the variability and richness of natural data, be it speech, glyphs, or other types of patterns, make it almost impossible to build an accurate recognition system entirely by hand. Consequently, most pattern recognition systems are built using a combination of automatic learn-

LeNet-5



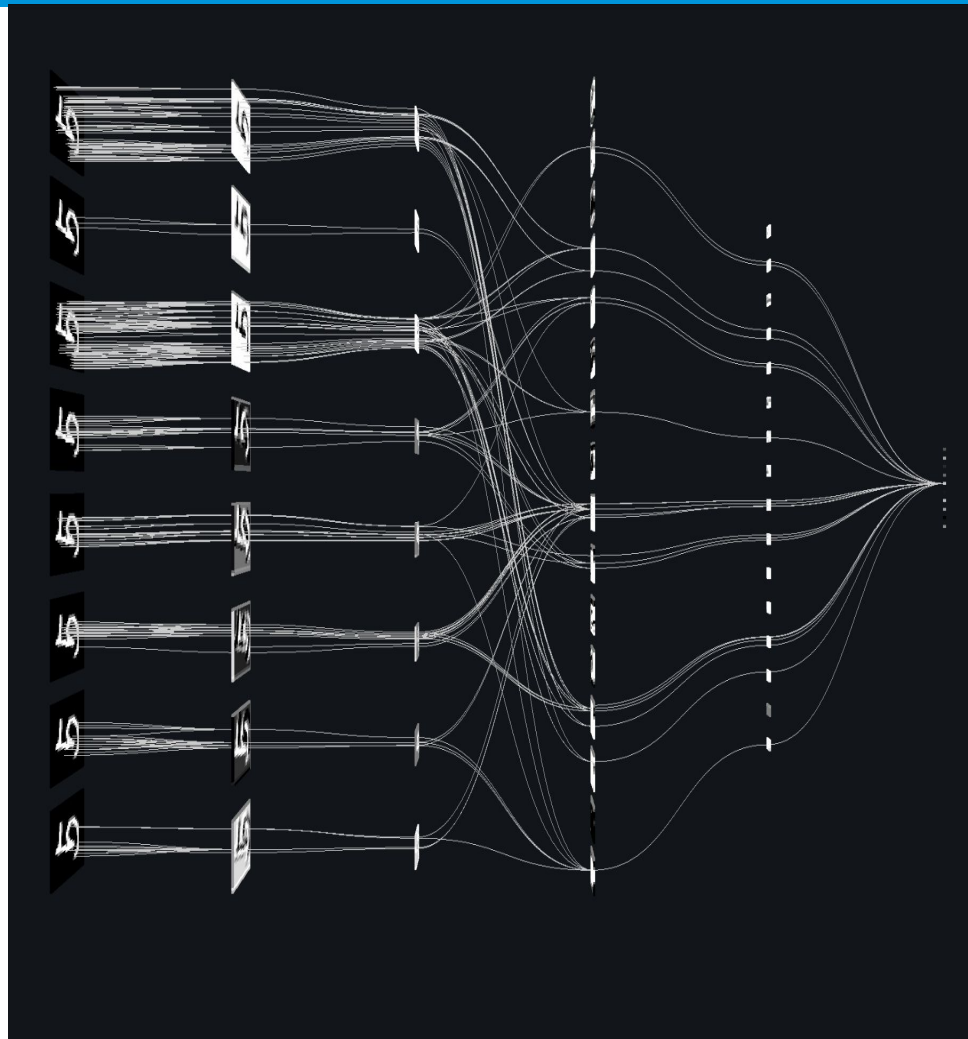
This architecture has some building blocks that you already know, such as **fully connected** layers and **sigmoid activation** functions, but it also introduces two new building blocks: **convolutional layers** and **pooling layers**

Why not simply use a deep neural network with fully connected layers for image recognition task (large images)?



CNN Layers

Convolutional Layer
Pooling Layer
Fully Connected Layer



Convolution Layer - Vertical Edge Detection

10x1 + 10x0 + 10x(-1) + 10x1 + 10x0 +
10x(-1) + 10x1 + 10x0 + 10x(-1) = 0

6 x 6 input image

10^{1}	10^{0}	10^{1}	0	0	0
10^{1}	10^{0}	10^{1}	0	0	0
10^{1}	10^{0}	10^{1}	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

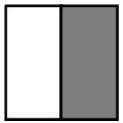
*

3 x 3 filter

1	0	-1
1	0	-1
1	0	-1

=

0			



Convolution Layer - Vertical Edge Detection

10x1 + 10x0 + 0x(-1) + 10x1 + 10x0 +
0x(-1) + 10x1 + 10x0 + 0x(-1) = 30

6 x 6 input image

10	10 1	10 0	0 -1	0	0
10	10 1	10 0	0 -1	0	0
10	10 1	10 0	0 -1	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

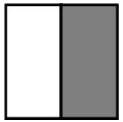
*

3 x 3 filter

1	0	-1
1	0	-1
1	0	-1

=

0	30		



Convolution Layer - Vertical Edge Detection

10x1 + 0x0 + 0x(-1) + 10x1 + 0x0 +
0x(-1) + 10x1 + 0x0 + 0x(-1) = 30

6 x 6 input image

10	10	10 (1)	0 (0)	0 (-1)	0
10	10	10 (1)	0 (0)	0 (-1)	0
10	10	10 (1)	0 (0)	0 (-1)	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

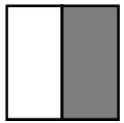
*

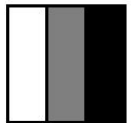
3 x 3 filter

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	





Convolution Layer - Vertical Edge Detection

0x1 + 0x0 + 0x(-1) + 0x1 + 0x0 + 0x(-1) +
0x1 + 0x0 + 0x(-1) = 0

6 x 6 input image

10	10	10	0 (1)	0 (0)	0 (-1)
10	10	10	0 (1)	0 (0)	0 (-1)
10	10	10	0 (1)	0 (0)	0 (-1)
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

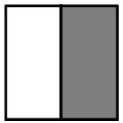
*

3 x 3 filter

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0



Convolution Layer - Vertical Edge Detection

10x1 + 10x0 + 10x(-1) + 10x1 + 10x0 + 10x(-1) + 10x1 + 10x0 + 10x(-1) = 0

6 x 6 input image

10	10	10	0	0	0
10^{1}	10^{0}	10^{-1}	0	0	0
10^{1}	10^{0}	10^{-1}	0	0	0
10^{1}	10^{0}	10^{-1}	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

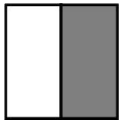
*

3 x 3 filter

1	0	-1
1	0	-1
1	0	-1

=

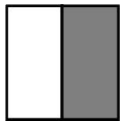
0	30	30	0
0			



Convolution Layer - Vertical Edge Detection

6 x 6 input image

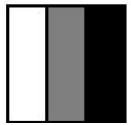
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0 1	0 0	0 -1
10	10	10	0 1	0 0	0 -1
10	10	10	0 1	0 0	0 -1



*

3 x 3 filter

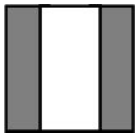
1	0	-1
1	0	-1
1	0	-1

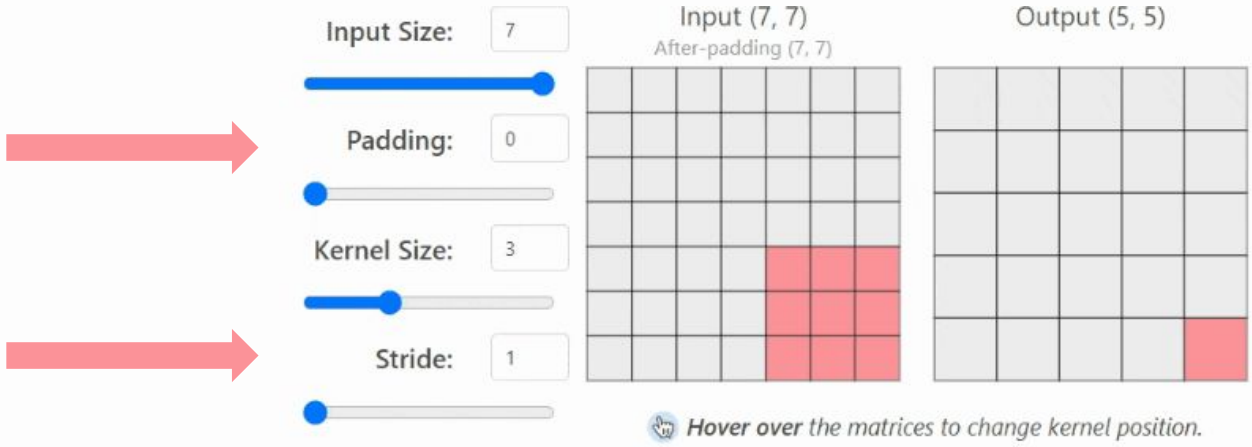


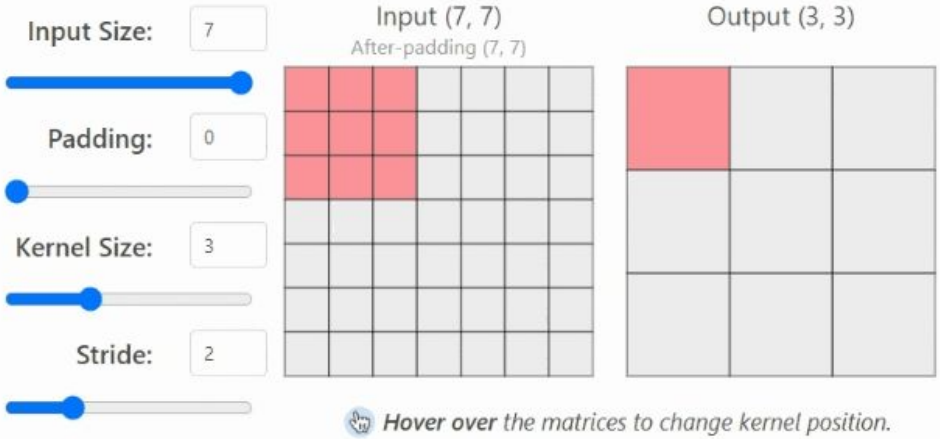
=

Feature Map

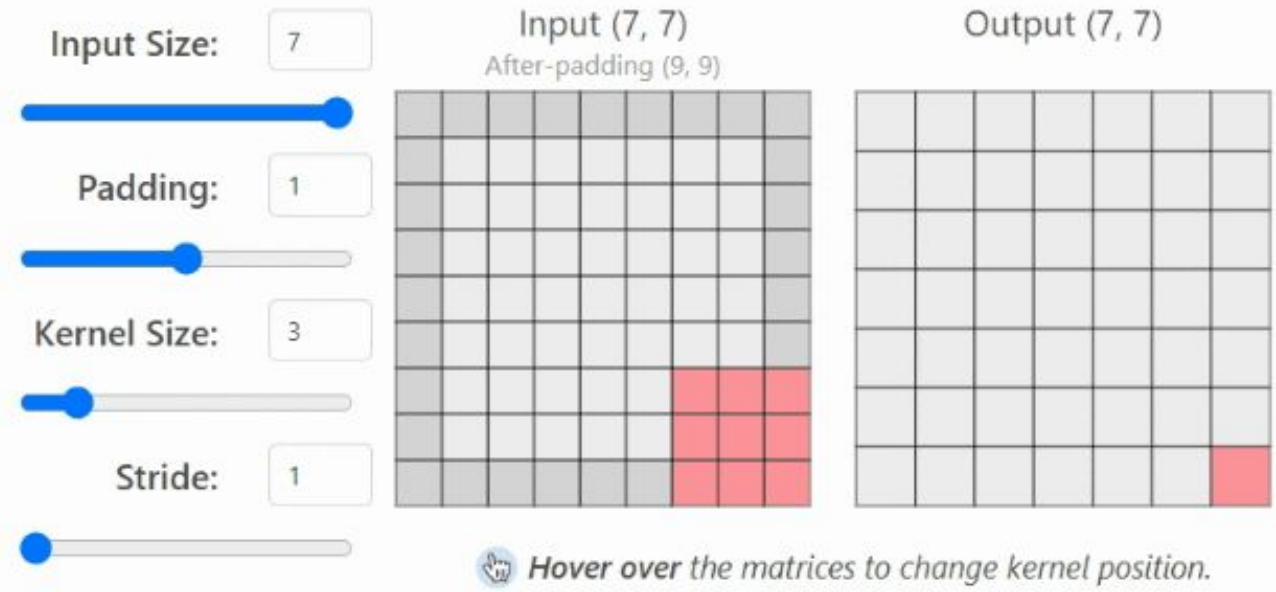
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0







https://poloclub.github.io/cnn-explainer/



Padding

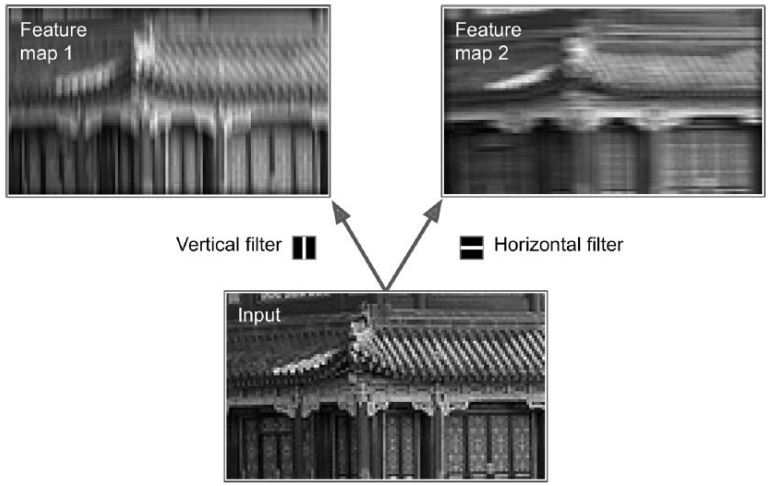
- Valid (0)
- Same (1)

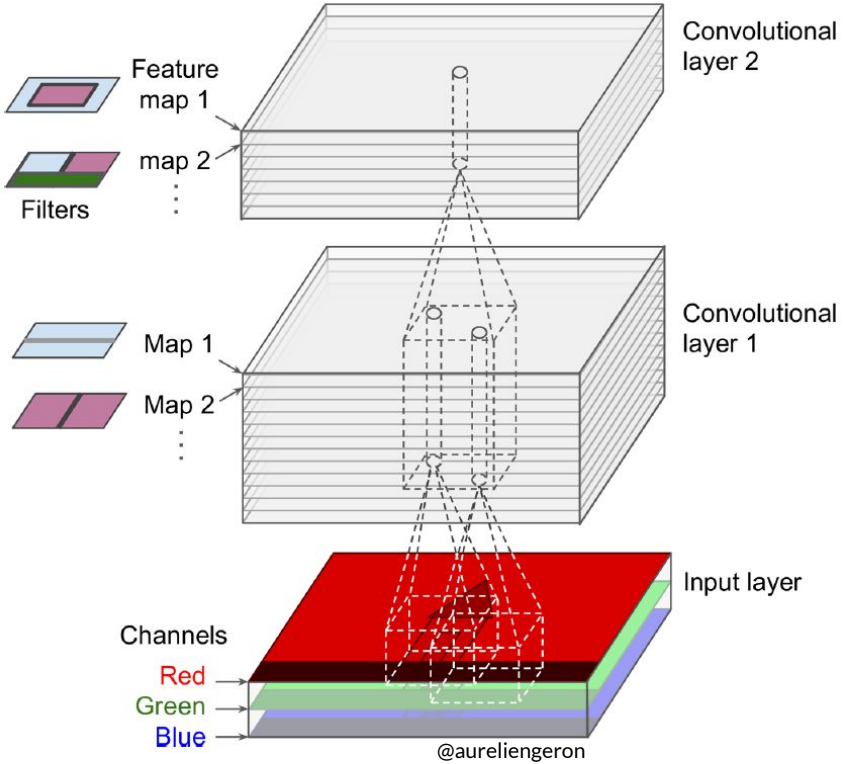
Summary of convolutions

- n x n image
- f x f filter
- padding p
- stride s

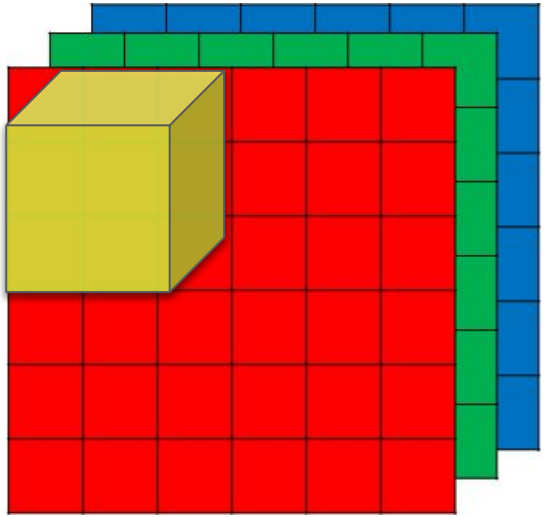
output size $\rightarrow \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$

Power of Learned Filters & Convolutions over volumes



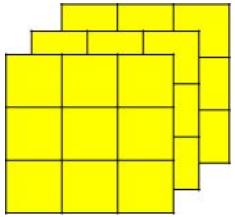


Convolution on RGB image



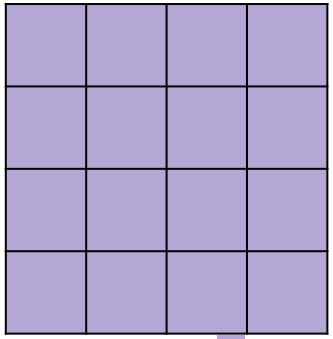
6x6x3
input image

*



3x3x3
filter

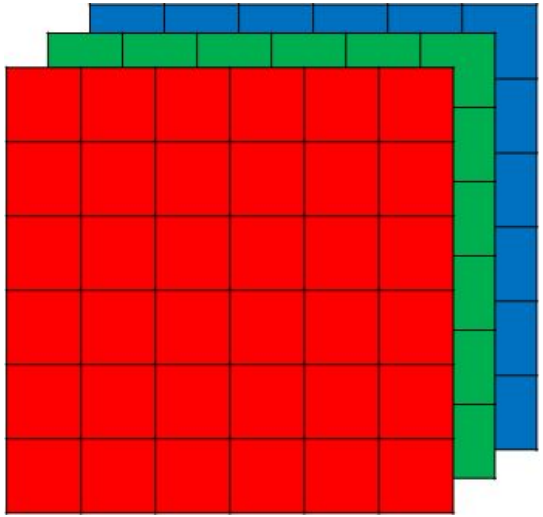
=



4 x 4 x 1
feature map

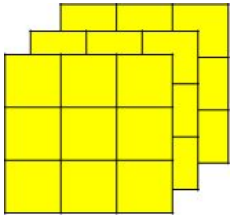
$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

Convolution on RGB image



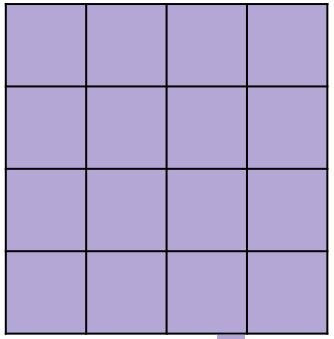
6x6x3
input image

*



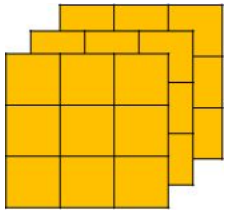
3x3x3
filter

=



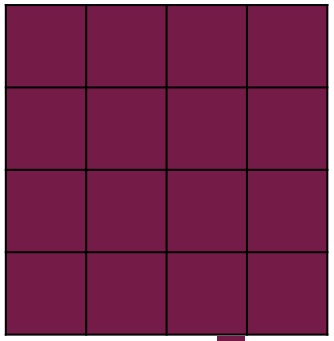
4 x 4 x 1
feature map

*



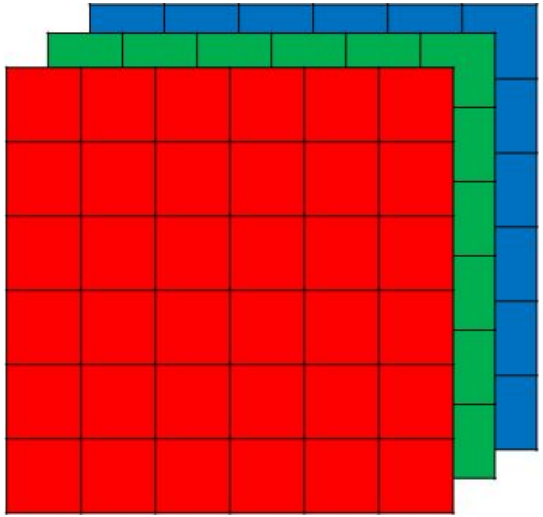
3x3x3
filter

=



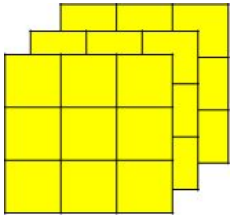
4 x 4 x 1
feature map

Convolution on RGB image



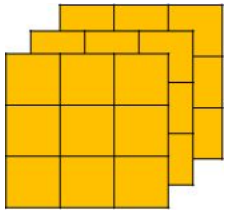
6x6x3
input image

*



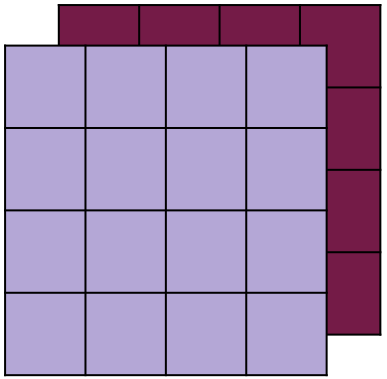
3x3x3
filter

*



3x3x3
filter

=



4 x 4 x 2
feature maps

Parameter sharing: a feature detector (such as a filter) that is useful in one part of the image is probably useful in another part of the image.

One layer of a convolutional network explained using CNN EXPLAINER

- Filter size, number of filters
- Feature map size
- Number of parameters
- Understanding nonlinearity

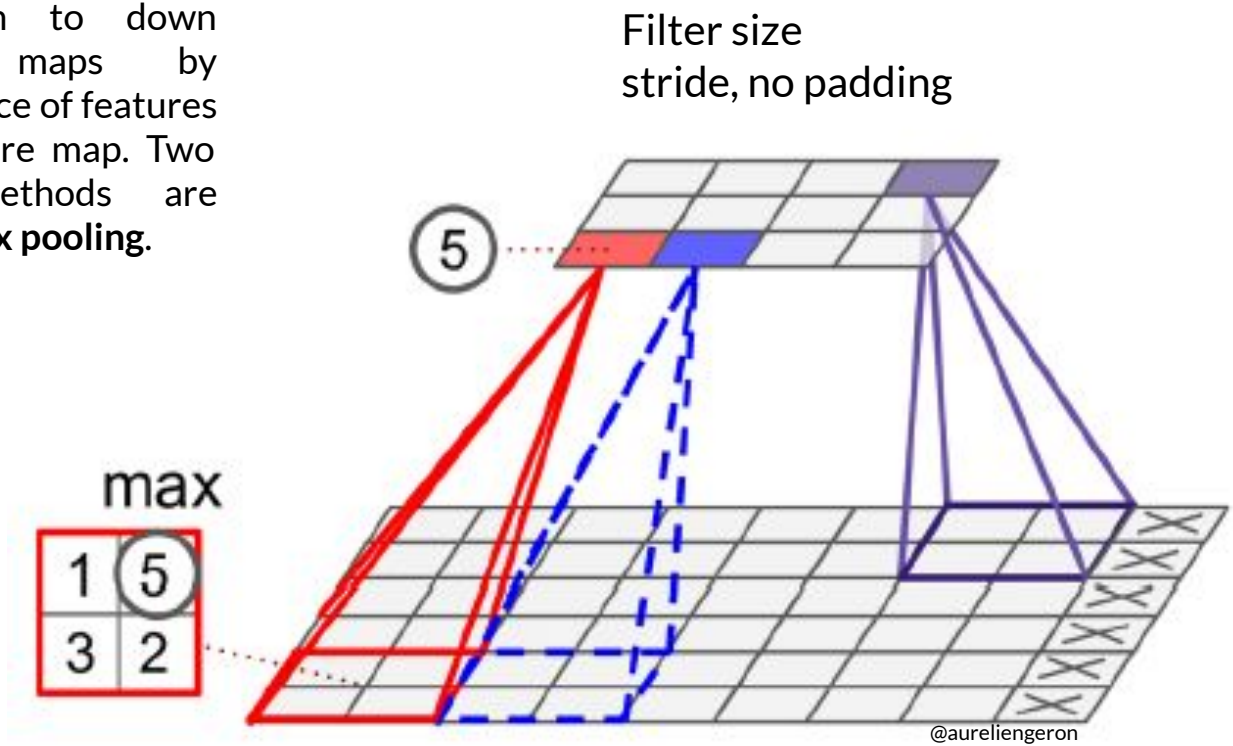
https://poloclub.github.io/cnn-explainer

```python
import tensorflow as tf

model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(10, (3,3), activation=tf.nn.relu),
  tf.keras.layers.Conv2D(10, (3,3), activation=tf.nn.relu)
])

# build the model
model.build(input_shape=(None,64,64,3))

# summarize layers
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 10)        280
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 60, 10)        910
=================================================================
```

Pooling Layer

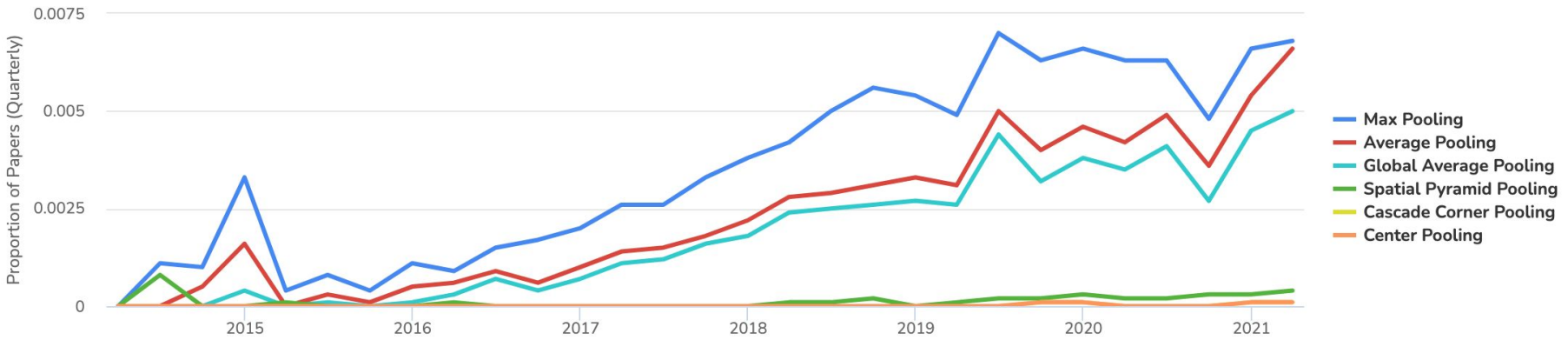
Provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are **average pooling** and **max pooling**.



A pooling layer has no weights (parameters), only hyperparameters

Pooling Operations

Usage Over Time

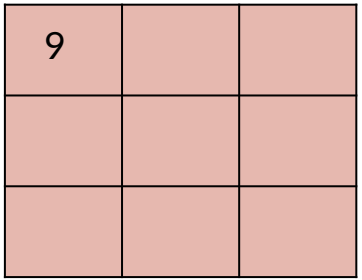


This feature is experimental; we are continuously improving our matching algorithm.
https://paperswithcode.com/methods/category/pooling-operation

Pooling Layer: Max pooling

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9

Feature map 5 x 5



Filter 3 x 3
Stride 1

Pooling Layer: Max pooling

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9

Feature map 5 x 5

9	9	

Filter 3 x 3
Stride 1

Pooling Layer: Max pooling

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9

Feature map 5 x 5

9	9	5

Filter 3 x 3
Stride 1

Pooling Layer: Max pooling

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9

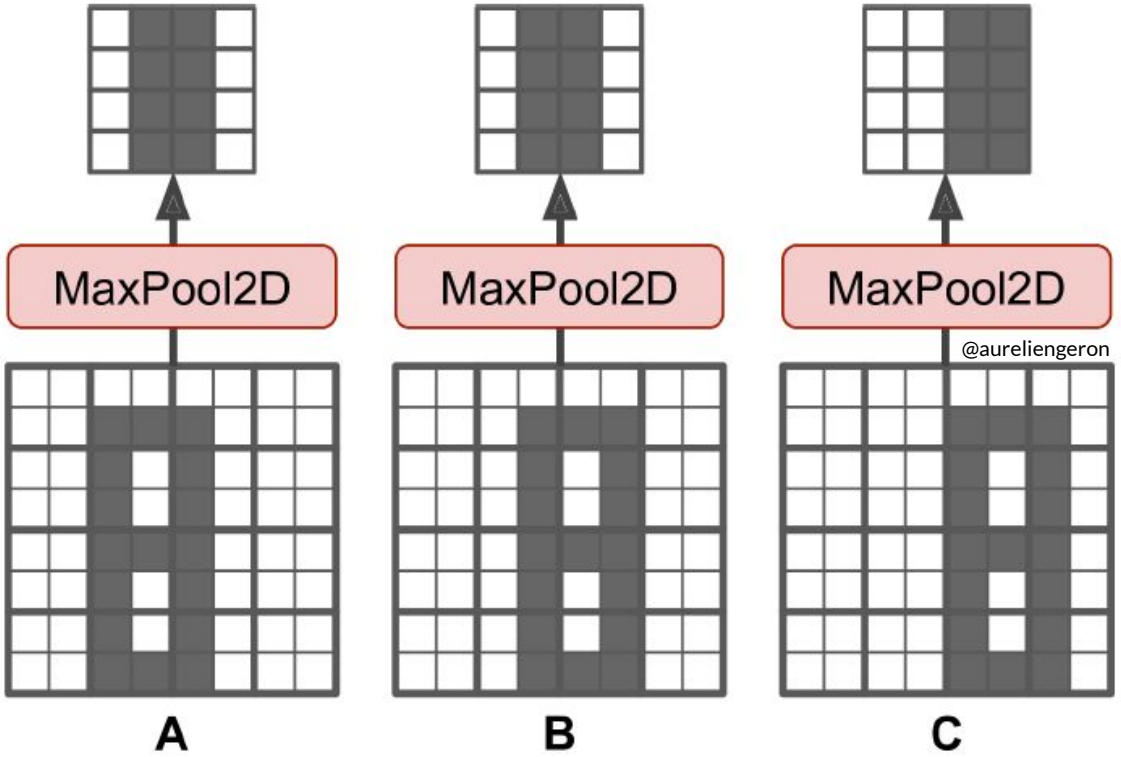
Feature map 5 x 5

9	9	5
9	9	5
8	6	9

Filter 3 x 3
Stride 1

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

Pooling Layer: local translation invariance



Pooling layer of a convolutional network explained using CNN EXPLAINER

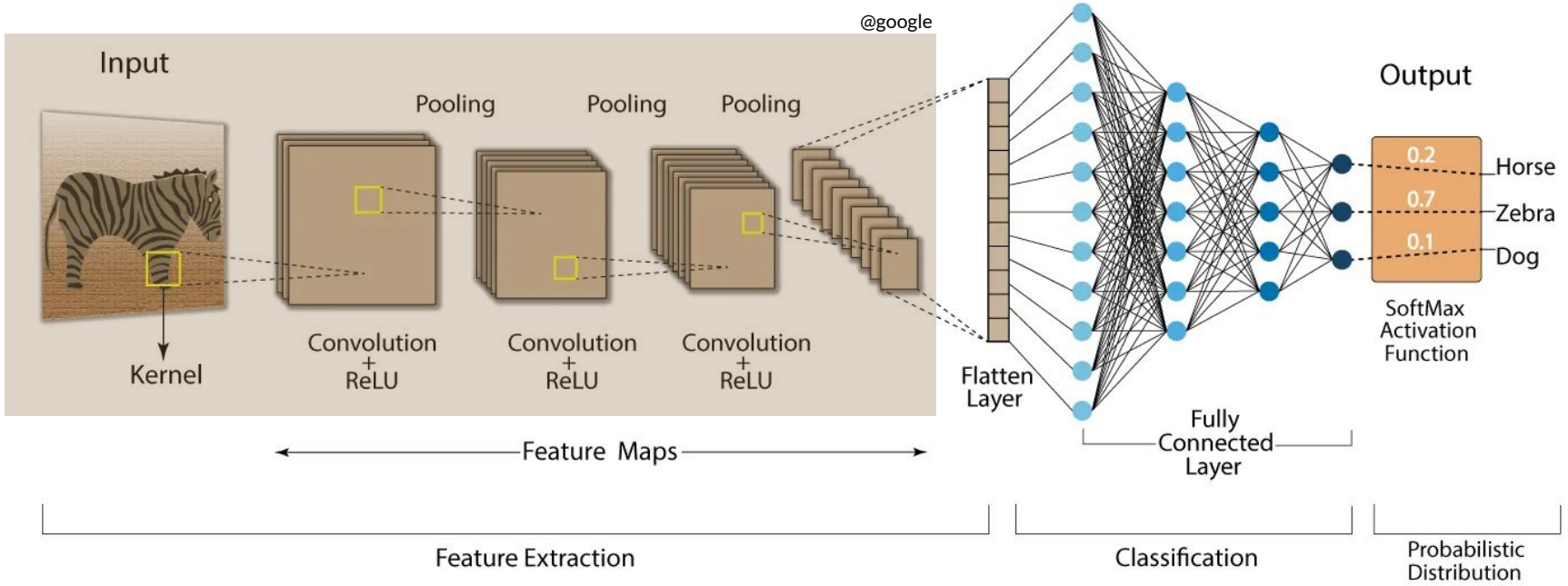
- Filter size, number of filters, stride
- Feature map size
- Number of parameters

https://poloclub.github.io/cnn-explainer

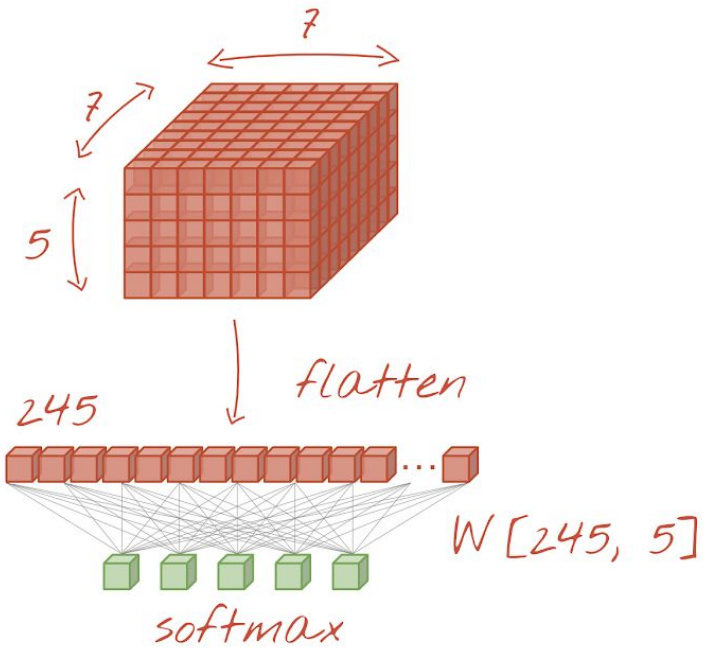
```
import tensorflow as tf

model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(10, (3,3), activation=tf.nn.relu),
  tf.keras.layers.Conv2D(10, (3,3), activation=tf.nn.relu),
  tf.keras.layers.MaxPool2D(pool_size=(2,2),strides=2)
])

# build the model
model.build(input_shape=(None,64,64,3))

# summarize layers
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 62, 62, 10)        280
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 60, 60, 10)        910
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 30, 30, 10)        0
=================================================================
```

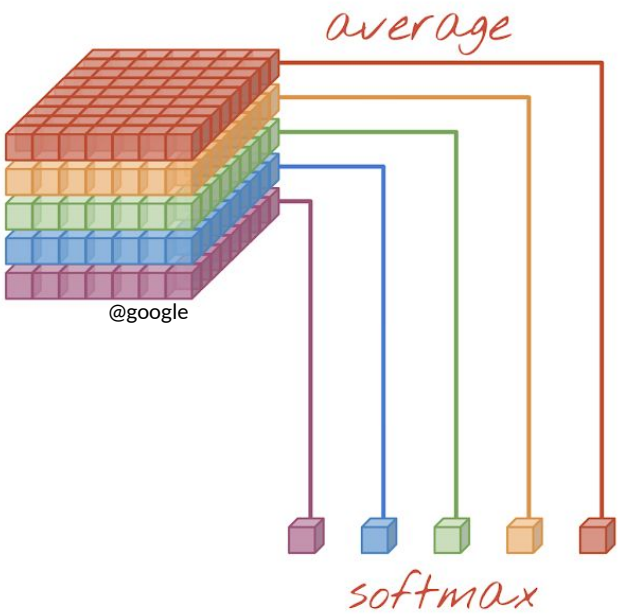
Fully Connected Layer



Fully connected layer



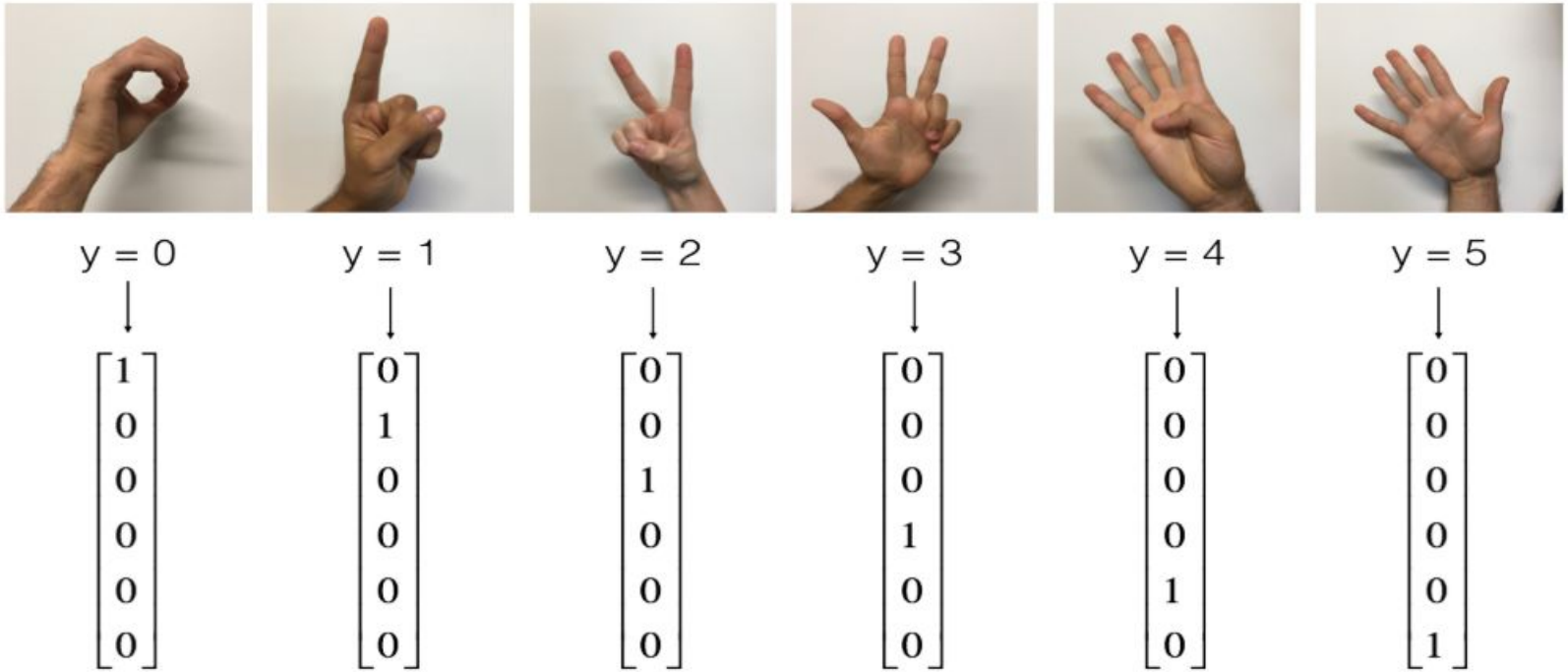
Global average pooling



1225 weights — cheaper → **0** weights

Case Study: signs dataset

deeplearning.ai



Baseline

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 25)	307225
dense_4 (Dense)	(None, 12)	312
dense_5 (Dense)	(None, 6)	78

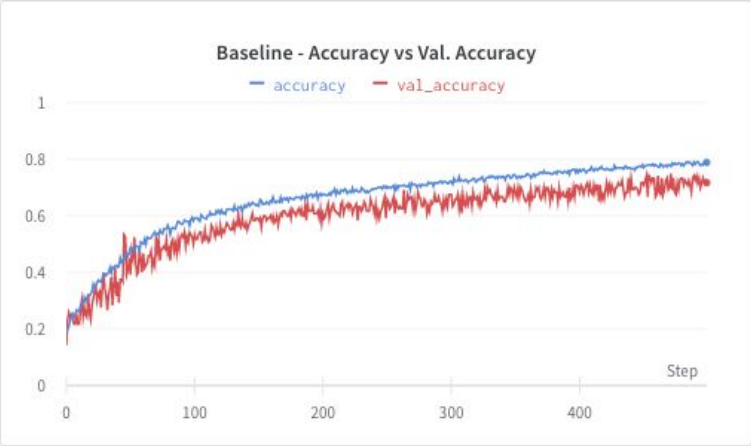
Total params: 307,615
Trainable params: 307,615
Non-trainable params: 0

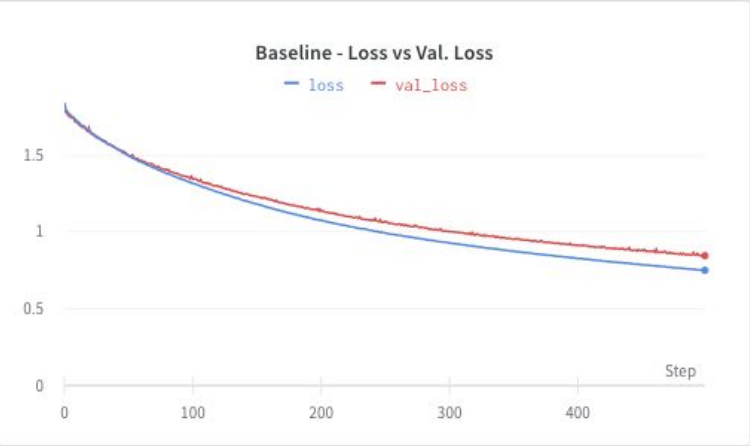
CNN

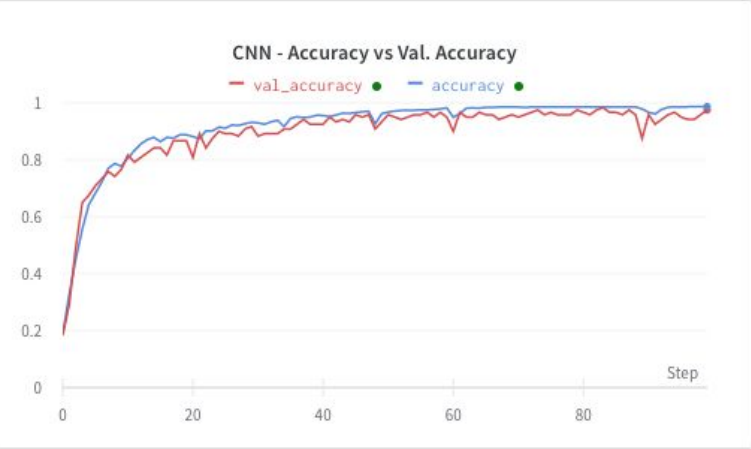
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 60, 60, 16)	1216
max_pooling2d (MaxPooling2D)	(None, 20, 20, 16)	0
conv2d_1 (Conv2D)	(None, 16, 16, 32)	12832
max_pooling2d_1 (MaxPooling2	(None, 5, 5, 32)	0
flatten (Flatten)	(None, 800)	0
dense (Dense)	(None, 64)	51264
dense_1 (Dense)	(None, 6)	390

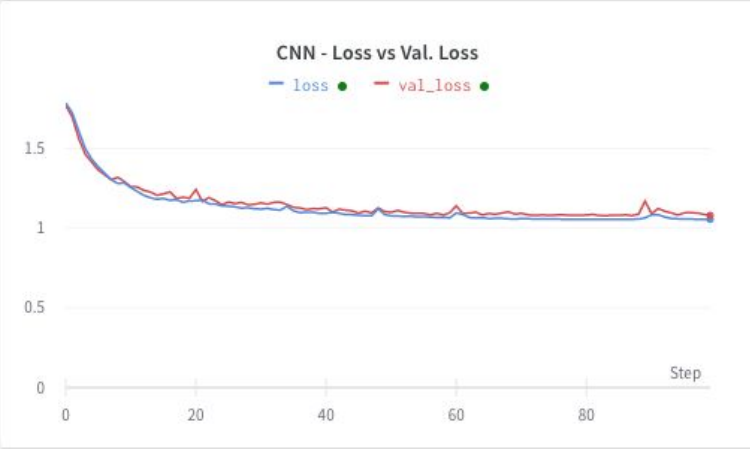
Total params: 65,702
Trainable params: 65,702
Non-trainable params: 0

Baseline vs CNN


Baseline - Accuracy vs Val. Accuracy
accuracy
val_accuracy
1
0.8
0.6
0.4
0.2
0
Step
0
100
200
300
400

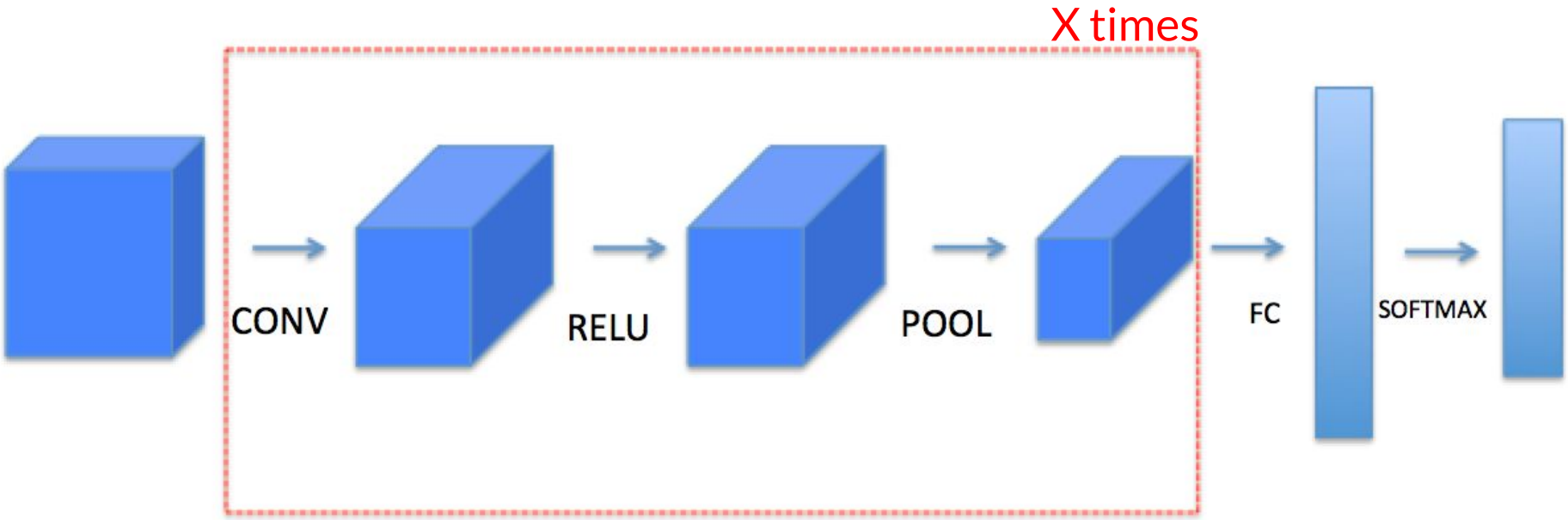

Baseline - Loss vs Val. Loss
loss
val_loss
1.5
1
0.5
0
Step
0
100
200
300
400


CNN - Accuracy vs Val. Accuracy
val_accuracy
accuracy
1
0.8
0.6
0.4
0.2
0
Step
0
20
40
60
80


CNN - Loss vs Val. Loss
loss
val_loss
1.5
1
0.5
0
Step
0
20
40
60
80

Weights & Biases

The building blocks of a convnet



```
model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(16, (5,5), activation=tf.nn.relu, input_shape=(64,64,3)),
  tf.keras.layers.MaxPool2D((3,3)),
  tf.keras.layers.Conv2D(32, (5,5), activation=tf.nn.relu),
  tf.keras.layers.MaxPool2D((3,3)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(64, activation=tf.nn.relu),
  tf.keras.layers.Dense(6, activation=tf.nn.softmax)
])
```

[Extension]

- Increase the number of blocks Conv + Pool
- Evaluate the first Conv layer with a filter (11,11) or (7,7)
- Use batch normalization and dropout
- Investigate a range of filter sizes
- Analyze the use of data augmentation
- Change the final layer for a GlobalAvgPool2D()
- Create a report on Wandb.ai and share it on twitter