Maior Dúvida da Aula

1. Convolução só funciona para imagens?
2. A principal função das redes CNNs é tratar de imagens? Podemos usar elas para quais problemas além de classificação?

Keras

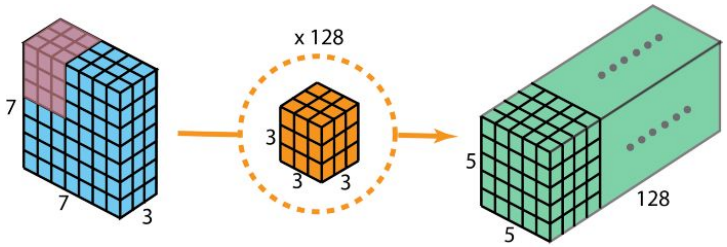
Search Keras documentation...

» Keras API reference / Layers API / Convolution layers

- About Keras
- Getting started
- Developer guides
- Keras API reference
  - Models API
  - Layers API
  - Callbacks API
  - Data preprocessing
  - Optimizers
  - Metrics

Convolution layers

- Conv1D layer
- Conv2D layer
- Conv3D layer
- SeparableConv1D layer
- SeparableConv2D layer
- DepthwiseConv2D layer
- Conv2DTranspose layer
- Conv3DTranspose layer



3. Gostaria de saber se existe alguma aplicação para um filtro que não tenha dois eixos de simetria, por exemplo retangular ou 'em estrela'. Todos os filtros mostrados eram quadrados e achei isso curioso.

Keras

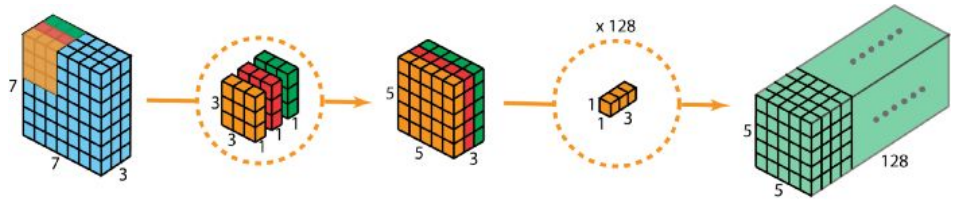
Search Keras documentation...

» Keras API reference / Layers API / Convolution layers

- About Keras
- Getting started
- Developer guides
- Keras API reference
  - Models API
  - Layers API
  - Callbacks API
  - Data preprocessing
  - Optimizers
  - Metrics

Convolution layers

- Conv1D layer
- Conv2D layer
- Conv3D layer
- SeparableConv1D layer
- SeparableConv2D layer
- DepthwiseConv2D layer
- Conv2DTranspose layer
- Conv3DTranspose layer



4. Qual a principal razão pela aplicação de múltiplos filtros em cada camada convolucional?

5. Uma maior quantidade de filtros é proporcional a uma maior diversidade quanto ao aprendizado da rede?

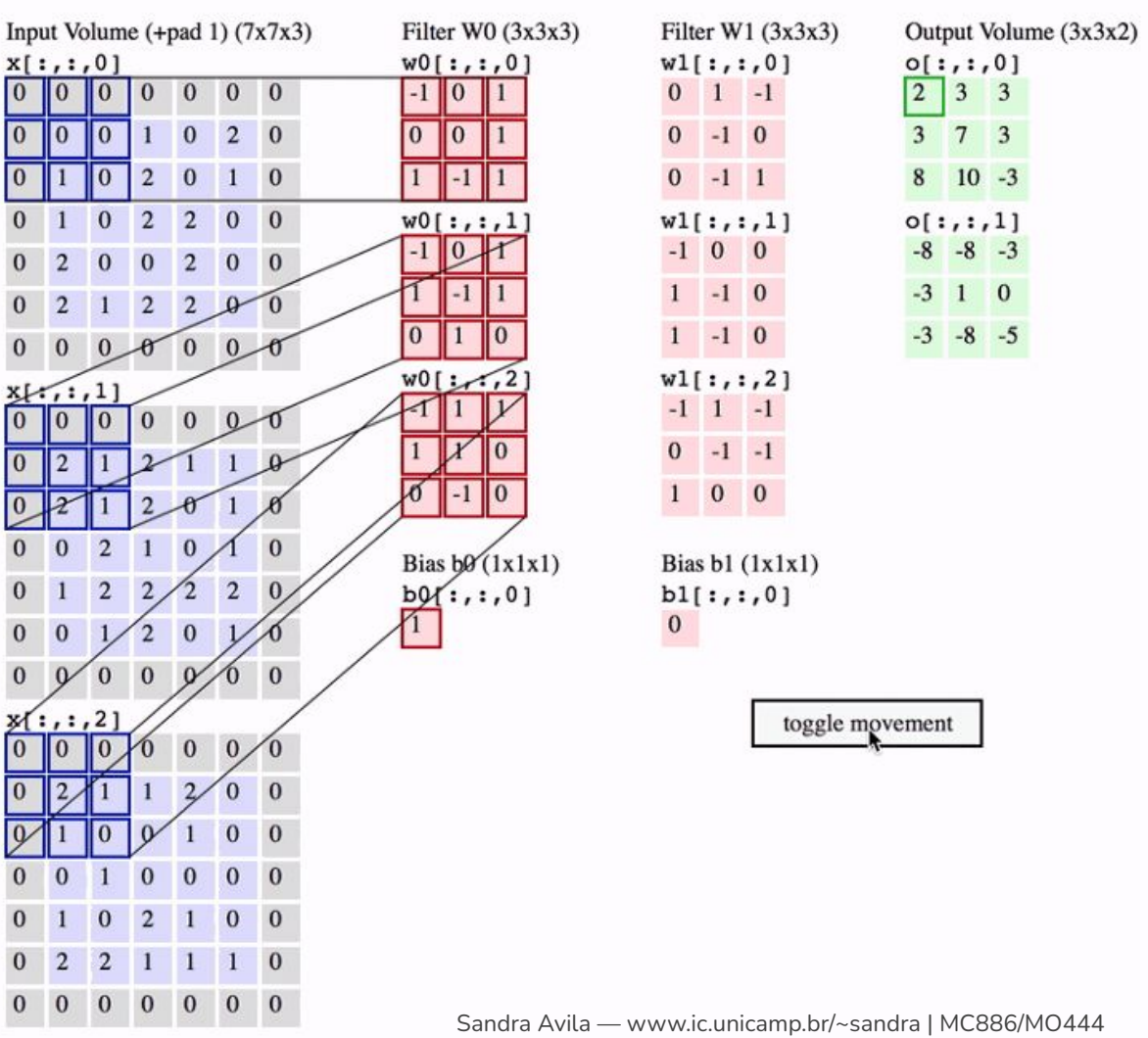
6. Como é determinado o tamanho do filtro?

7. Seria possível a rede aprender o tamanho do filtro e stride ideal?

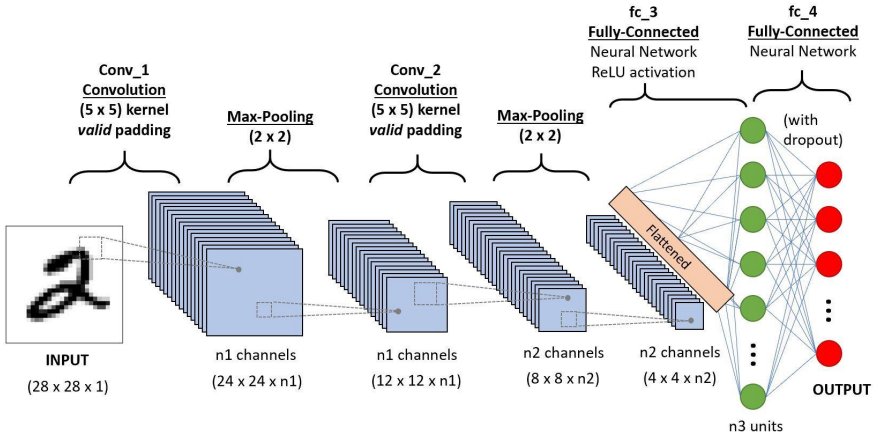
8. Existe algum tipo de rede que aceite imagens de variados tamanhos? Até agora vimos apenas redes para um determinado tamanho, ex 28x28.

9. Qual é o target(y) usado pela rede neural que cria um filtro na rede convolucional para poder fazer o aprendizado?

http://cs231n.github.io/convolutional-networks



Convolutional Neural Networks (CNNs)

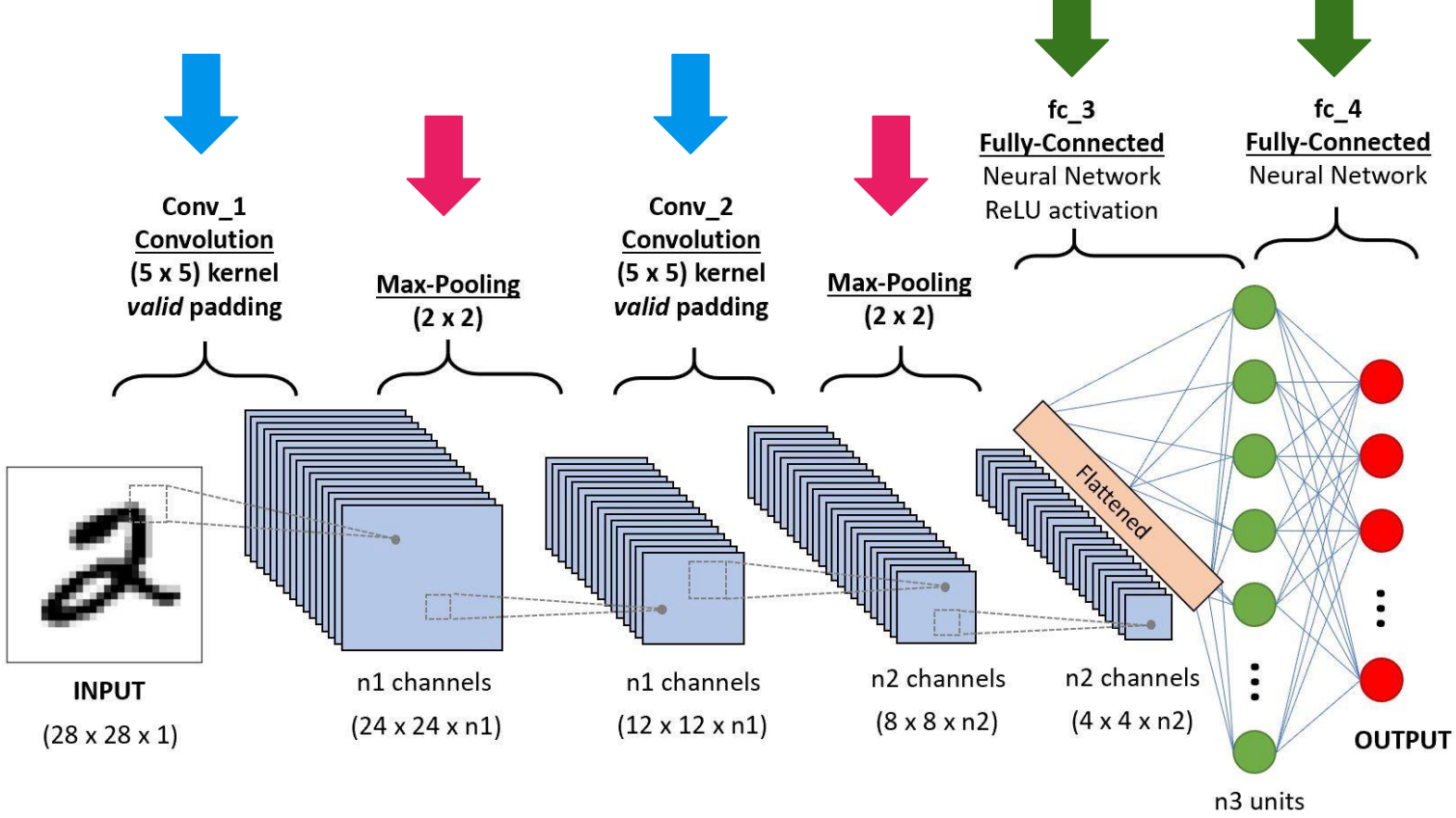


"Gradient-based learning applied to document recognition", 1998 http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Today's Agenda

- ~~What is Deep Learning?~~
- ~~Deep Learning & Applications~~
- ~~Neural Networks vs. Convolutional Networks~~
- ~~What is a convolution?~~
- Convolutional Neural Networks
  - ~~Convolution Layer~~
  - Pooling Layer
  - Fully-connected Layer



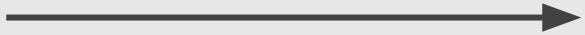
There are a few distinct types of layers (e.g., CONV/POOL/FC are by far the most popular).

Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently

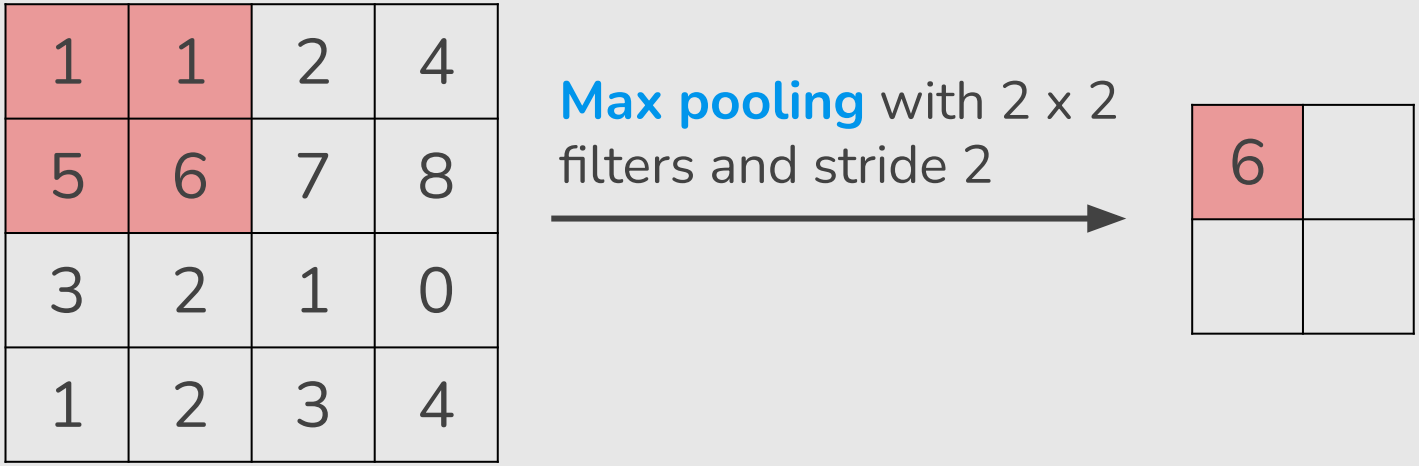
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Max pooling with 2 x 2 filters and stride 2



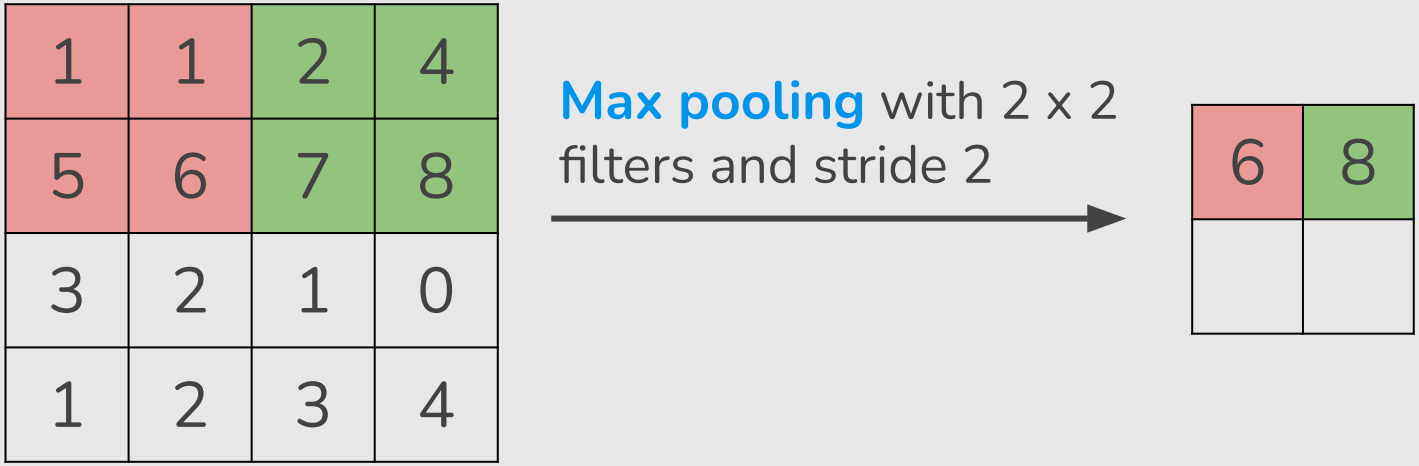
Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently



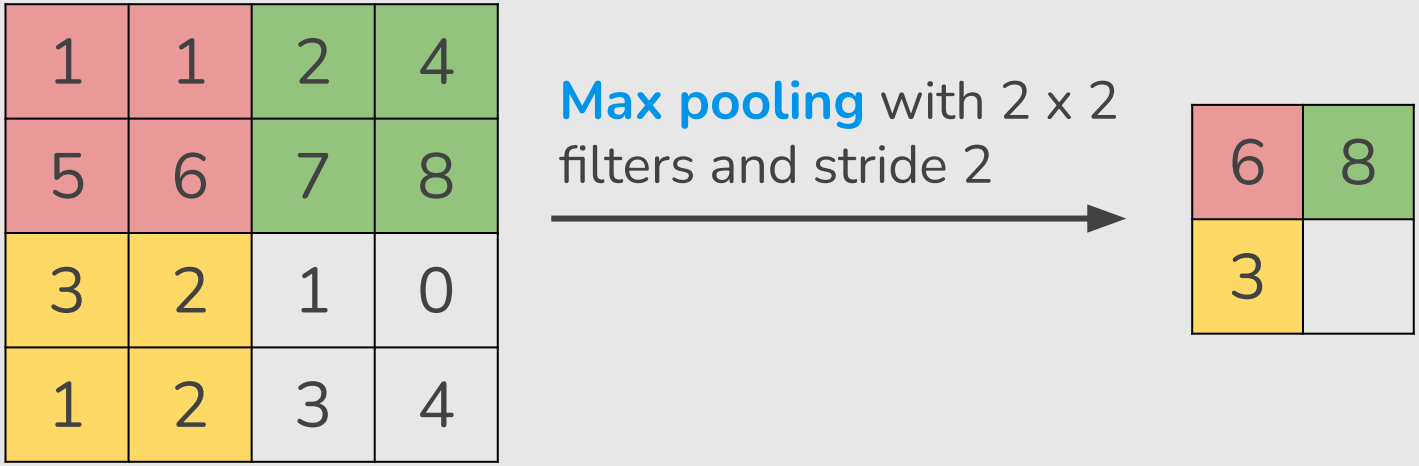
Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently



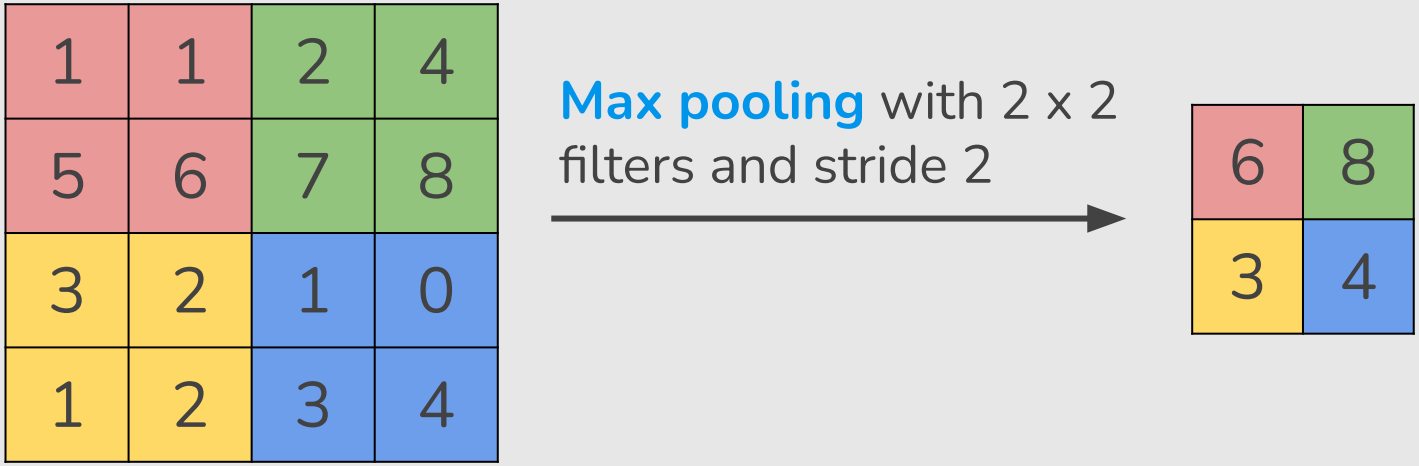
Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently



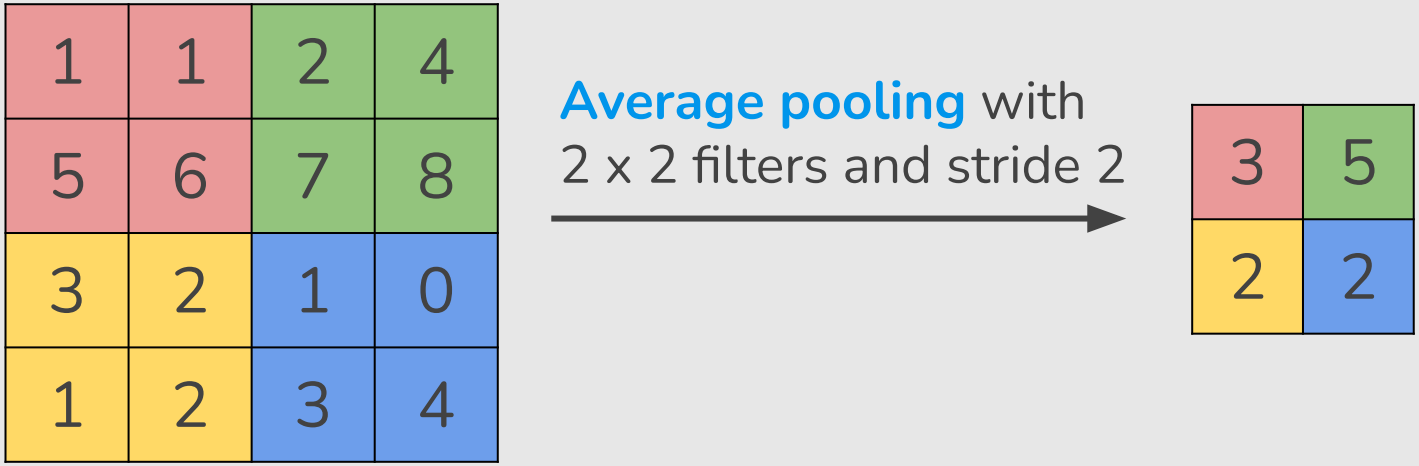
Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently



Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently



```
from tensorflow.keras import layers

model = tf.keras.Sequential()
#Camada convolucional com 10 filtros de tamanho 3x3 e ativação ReLU
model.add(layers.Conv2D(10, 3, padding='valid', activation='relu', input_shape=(28,28,1)))
#Max pooling de tamanho 2x2
model.add(layers.MaxPooling2D(pool_size=(2,2)))




model.summary()
```

MNIST 28 x 28

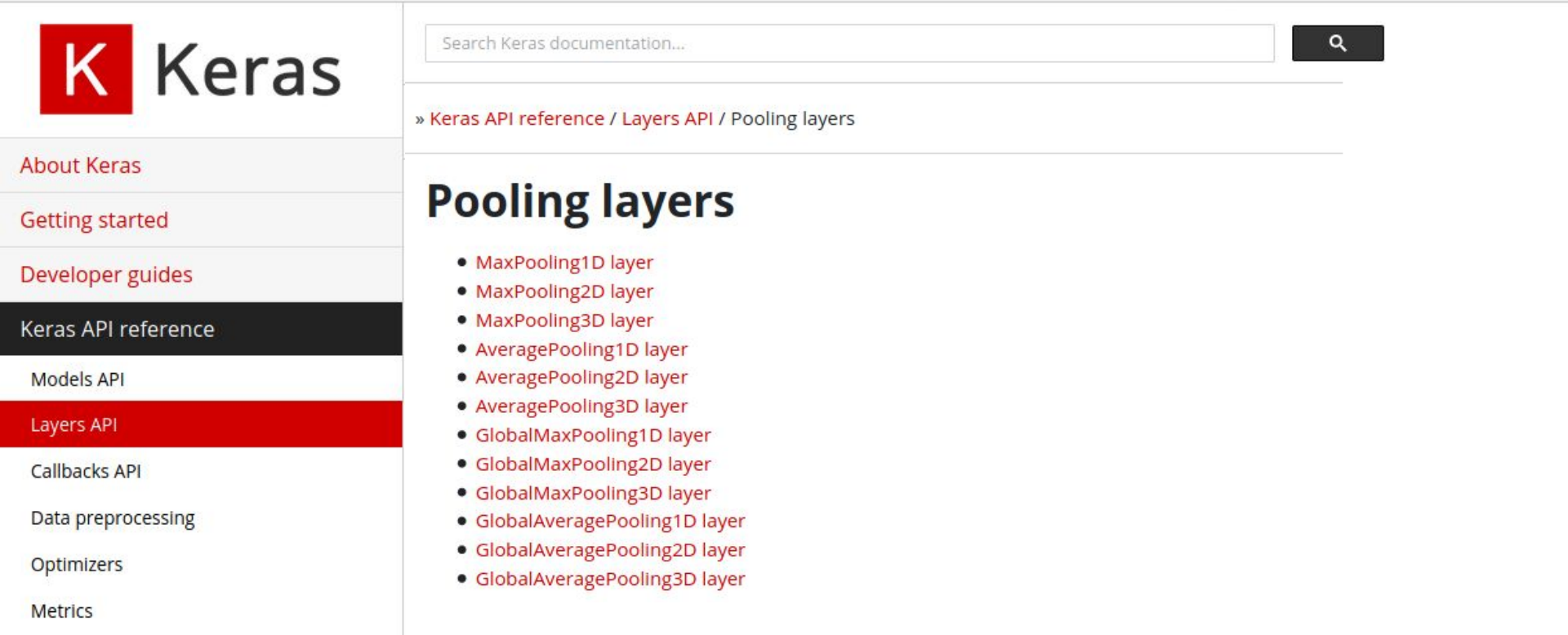
```
from tensorflow.keras import layers

model = tf.keras.Sequential()
#Camada convolucional com 10 filtros de tamanho 3x3 e ativação ReLU
model.add(layers.Conv2D(10, 3, padding='valid', activation='relu', input_shape=(28,28,1)))
#Max pooling de tamanho 2x2
model.add(layers.MaxPooling2D(pool_size=(2,2)))



model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 10)        100
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 10)        0
```

MNIST 28 x 28

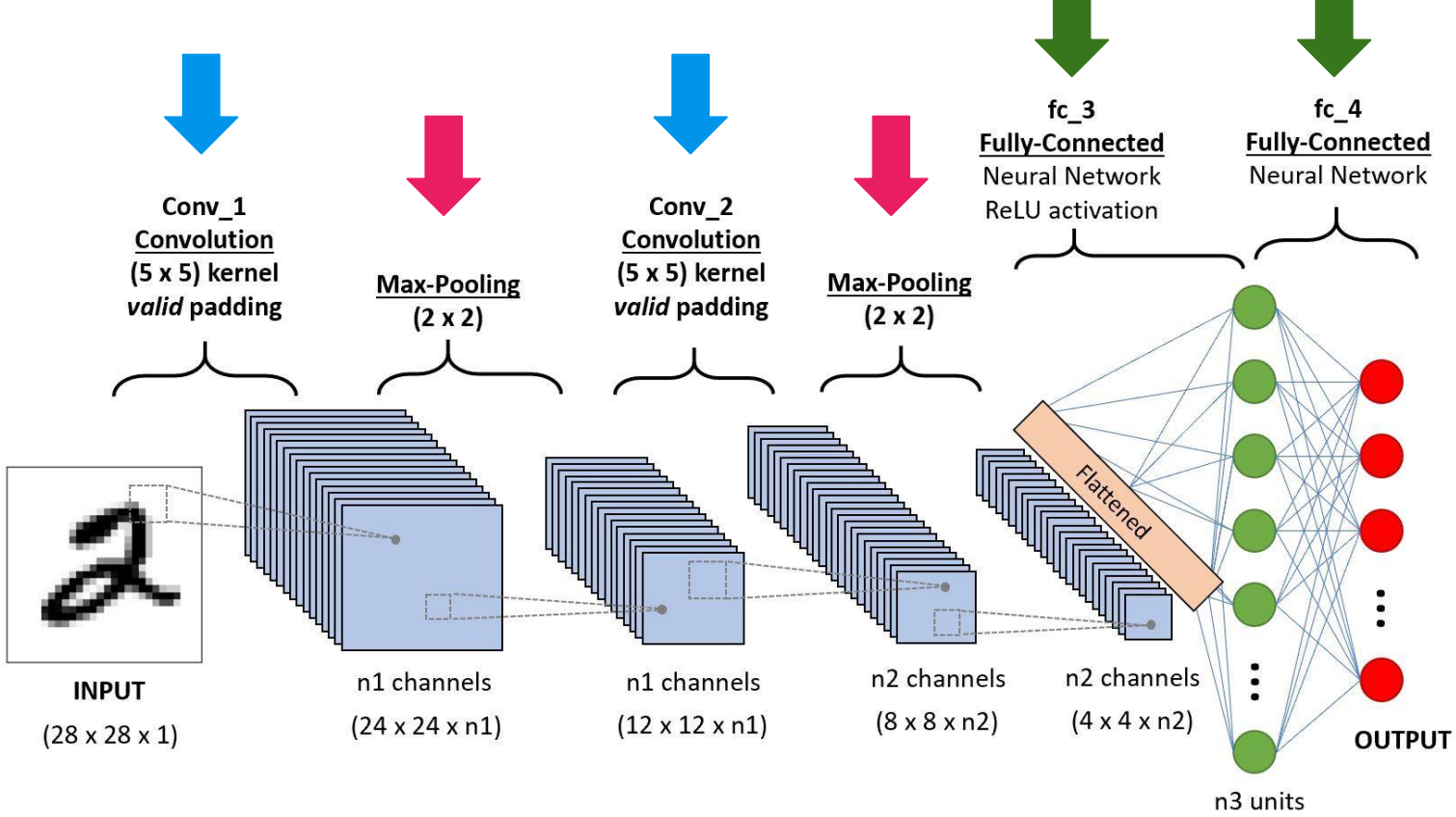
Pooling Layers

https://keras.io/api/layers/pooling_layers



Today's Agenda

- ~~What is Deep Learning?~~
- ~~Deep Learning & Applications~~
- ~~Neural Networks vs. Convolutional Networks~~
- ~~What is a convolution?~~
- Convolutional Neural Networks
  - ~~Convolution Layer~~
  - ~~Pooling Layer~~
  - Fully-connected Layer



There are a few distinct types of layers (e.g., **CONV**/**POOL**/**FC** are by far the most popular).

Fully Connected Layer

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

```
from tensorflow.keras import layers

model = tf.keras.Sequential()
#Camada convolucional com 10 filtros de tamanho 3x3 e ativação ReLU
model.add(layers.Conv2D(10, 3, padding='valid', activation='relu', input_shape=(28,28,1)))
#Max pooling de tamanho 2x2
model.add(layers.MaxPooling2D(pool_size=(2,2)))
#Operação de vetorização dos dados
model.add(layers.Flatten())
#Densa com 10 nós de saída
model.add(layers.Dense(10))

model.summary()
```

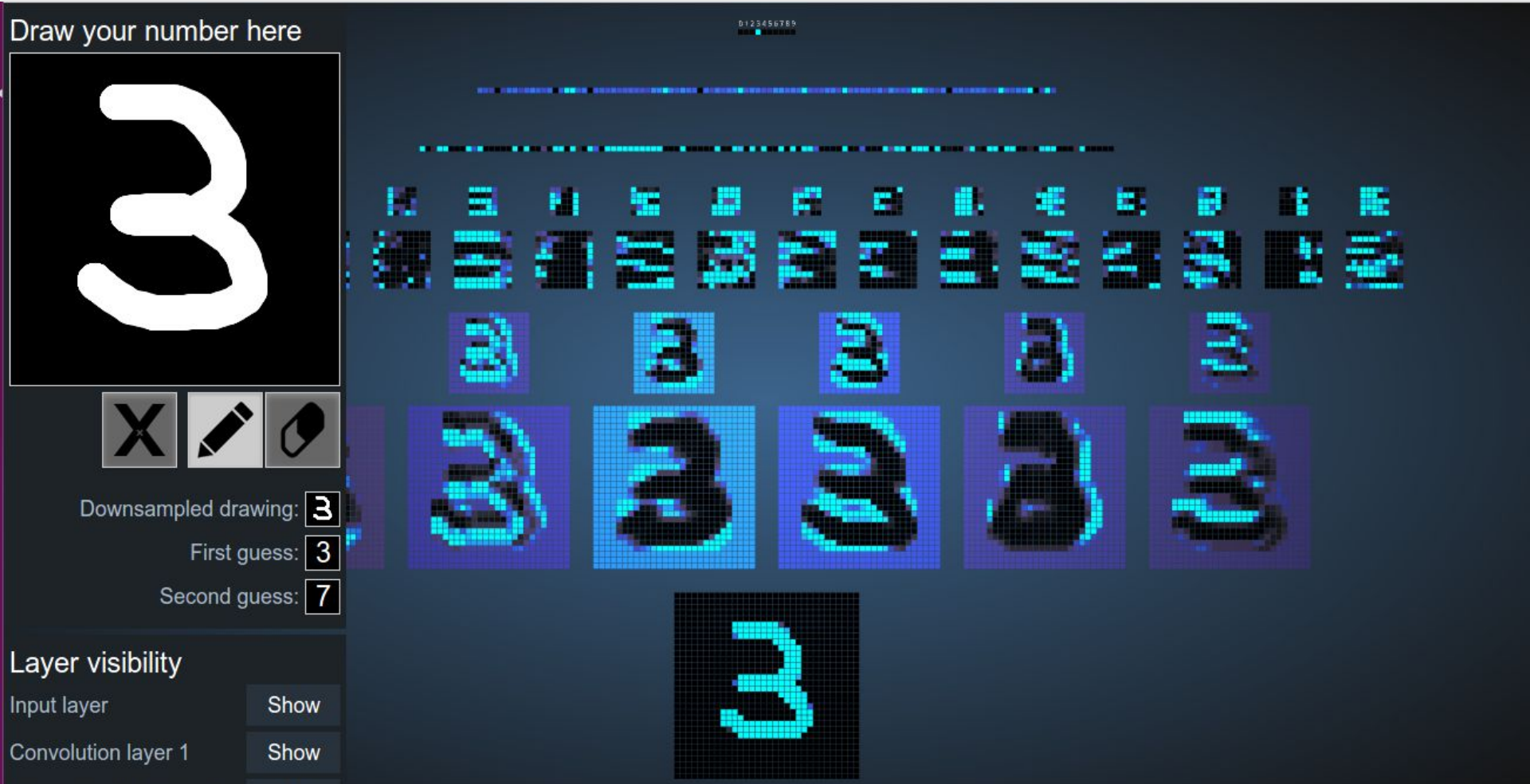
MNIST 28 x 28

```python
from tensorflow.keras import layers

model = tf.keras.Sequential()
#Camada convolucional com 10 filtros de tamanho 3x3 e ativação ReLU
model.add(layers.Conv2D(10, 3, padding='valid', activation='relu', input_shape=(28,28,1)))
#Max pooling de tamanho 2x2
model.add(layers.MaxPooling2D(pool_size=(2,2)))
#Operação de vetorização dos dados
model.add(layers.Flatten())
#Densa com 10 nós de saída
model.add(layers.Dense(10))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 10)        100
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 10)        0
_________________________________________________________________
flatten (Flatten)            (None, 1690)              0
_________________________________________________________________
dense (Dense)                (None, 10)                16910
=================================================================
Total params: 17,010
Trainable params: 17,010
Non-trainable params: 0
```

MNIST 28 x 28

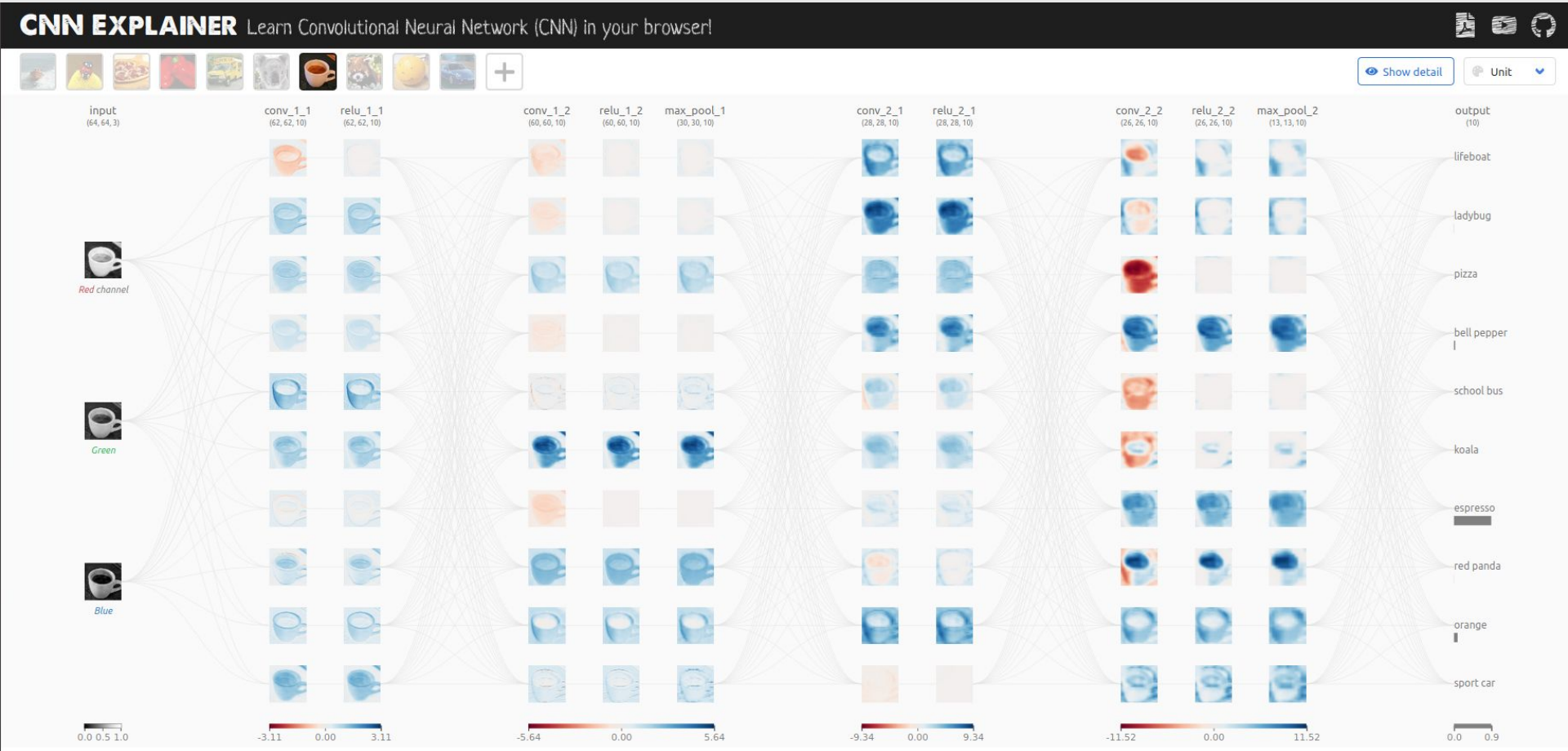
https://adamharley.com/nn_vis/cnn/2d.html



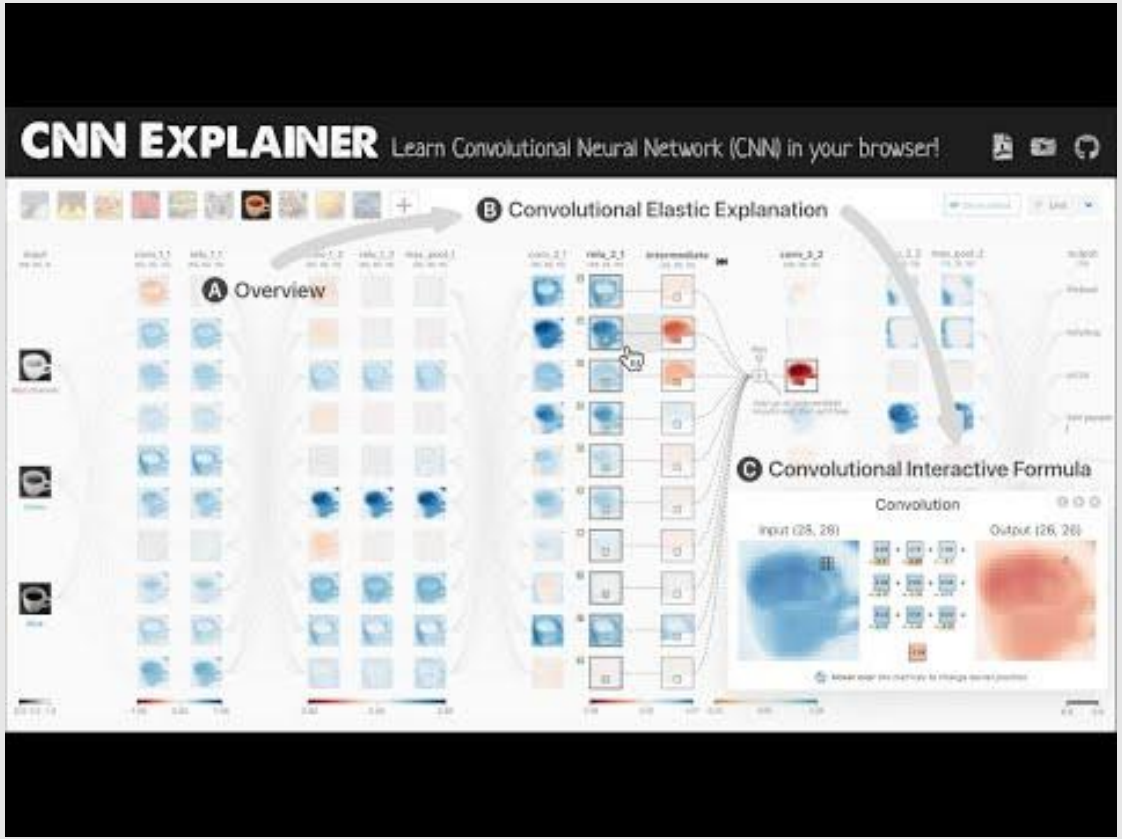
Visualizing a CNN trained on Handwritten Digits

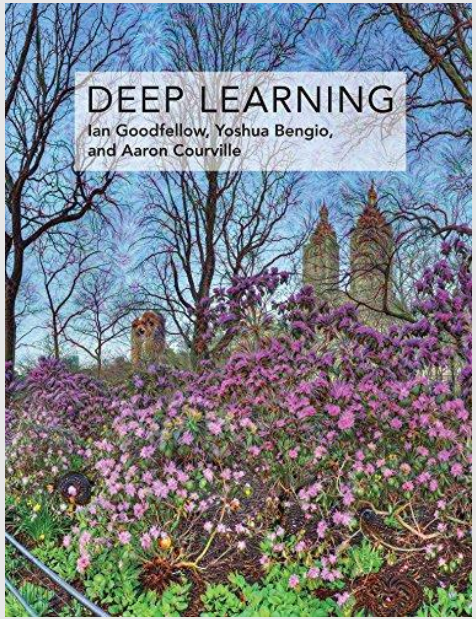
- Input image: 1024 pixels (32 x 32 image)
- CONV 1 (+ RELU): 6 5 × 5 (stride 1) filters
- POOL 1: 2 × 2 max pooling (with stride 2)
- CONV 2 (+ RELU): 16 5 × 5 (stride 1) filters
- POOL 2: 2 × 2 max pooling (with stride 2)
- 2 FC layers:
  - 120 neurons in the first FC layer
  - 100 neurons in the second FC layer
- Output layer: 10 neurons in the third FC

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



https://youtu.be/HnWIHWFbuUQ (3 min)





“Deep Learning”, Goodfellow & Bengio & Courville, 2016.

http://www.deeplearningbook.org/contents/convnets.html

Chapter 9

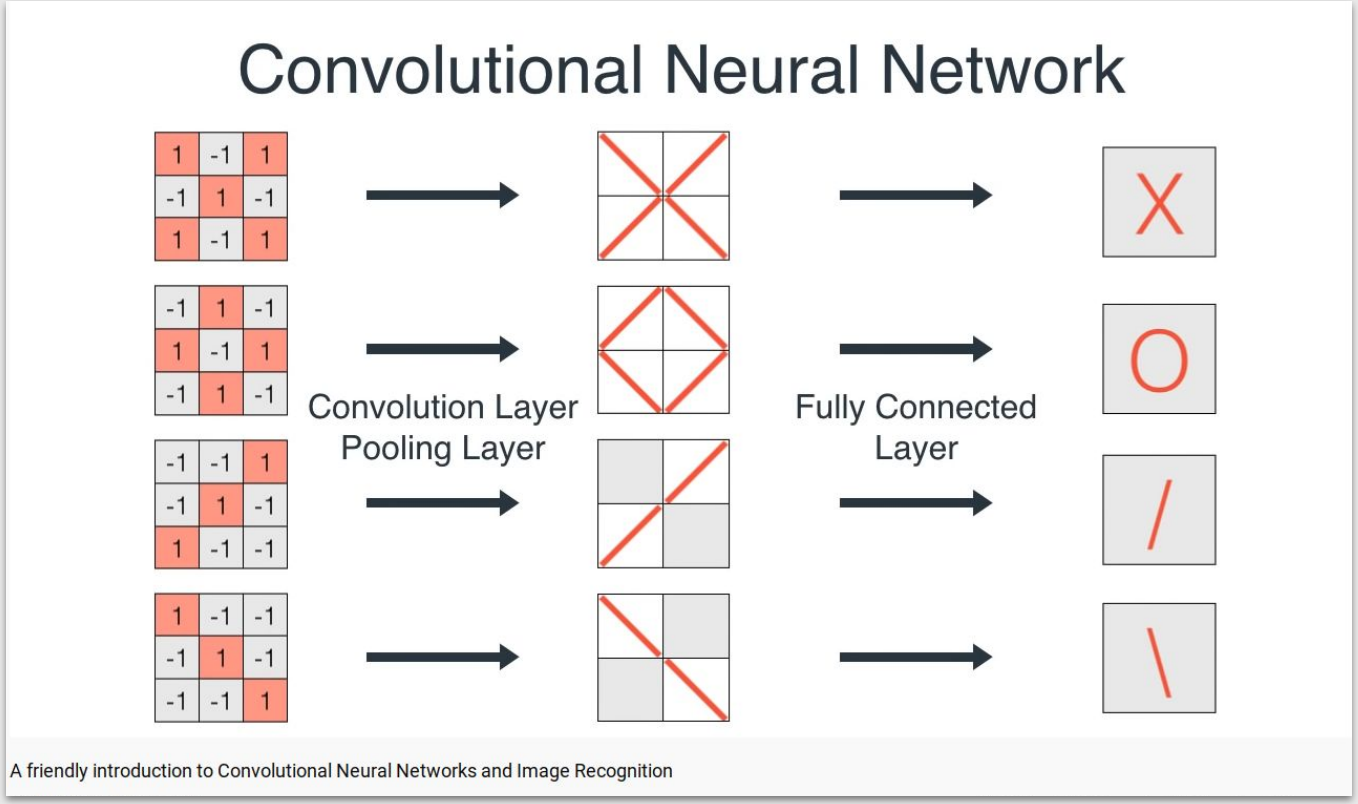
Convolutional Networks

Convolutional networks (LeCun, 1989), also known as **convolutional neural networks**, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name “convolutional neural network” indicates that the network employs a mathematical operation called **convolution**. Convolution is a specialized kind of linear operation. *Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.*

In this chapter, we first describe what convolution is. Next, we explain the motivation behind using convolution in a neural network. We then describe an operation called **pooling**, which almost all convolutional networks employ. Usually, the operation used in a convolutional neural network does not correspond precisely to the definition of convolution as used in other fields, such as engineering or pure mathematics. We describe several variants on the convolution function that are widely used in practice for neural networks. We also show how convolution may be applied to many kinds of data, with different numbers of dimensions. We then discuss means of making convolution more efficient. Convolutional networks stand out as an example of neuroscientific principles influencing deep learning. We discuss these neuroscientific principles, then conclude with comments about the role convolutional networks have played in the history of deep learning. One topic this chapter does not address is how to choose the architecture of your convolutional network. The goal of this chapter is to describe the kinds of tools that convolutional networks provide, while chapter 11 describes general guidelines

326

“A friendly introduction to Convolutional Neural Networks and Image Recognition” https://youtu.be/2-Ol7ZB0MmU



recod.ai
reasoning for complex data

CNN Architectures

Machine Learning

Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

MC886/MO444, October 27, 2022

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - GoogLeNet (2014)
  - ResNet (2015)

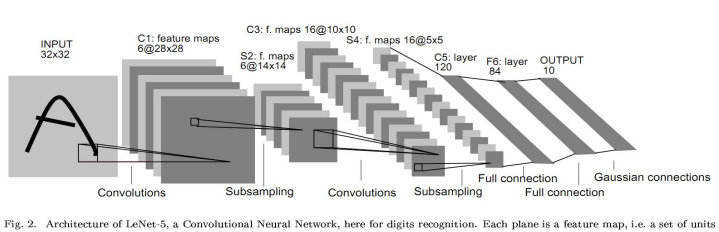
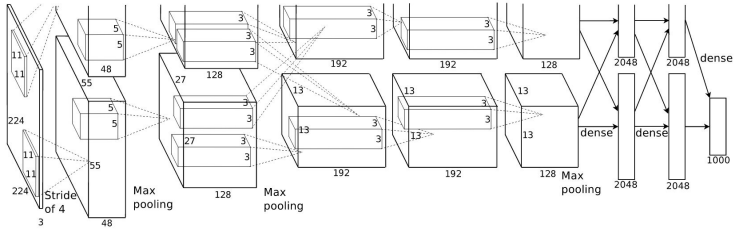
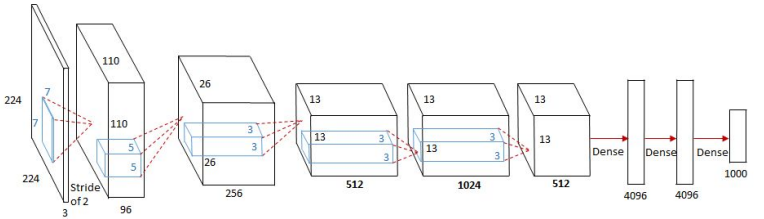
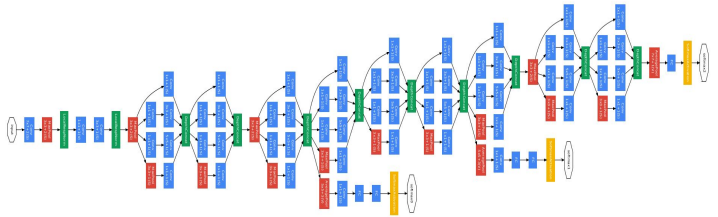
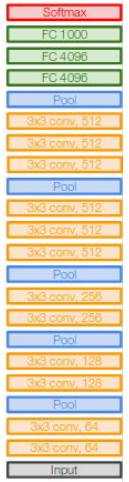


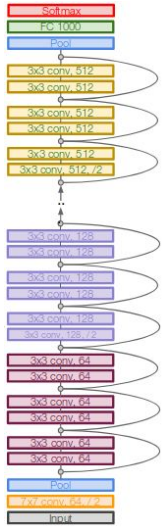
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.











Today's Agenda

- CNN Architectures
  - **LeNet (1998)**
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - GoogLeNet (2014)
  - ResNet (2015)

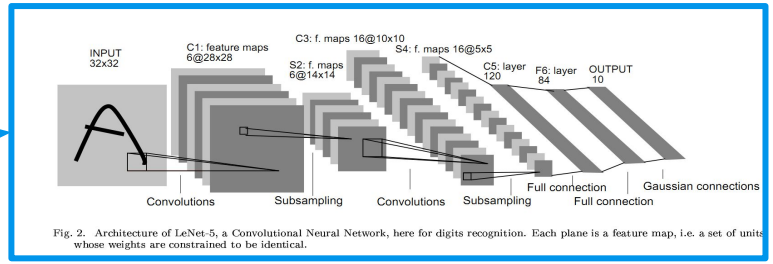
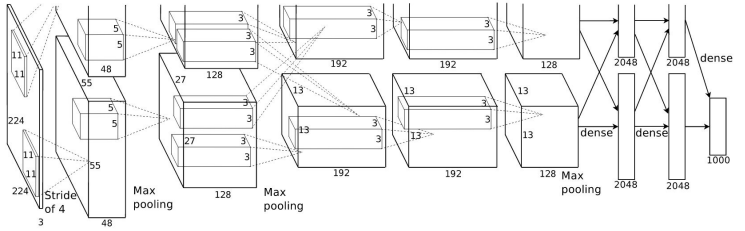
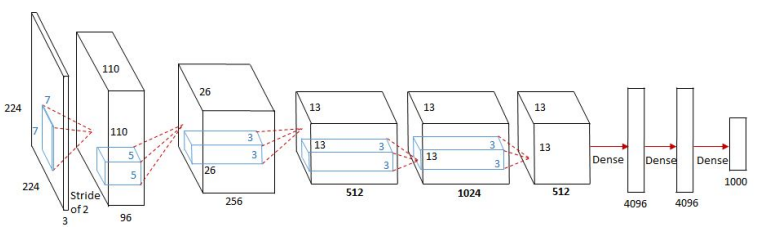
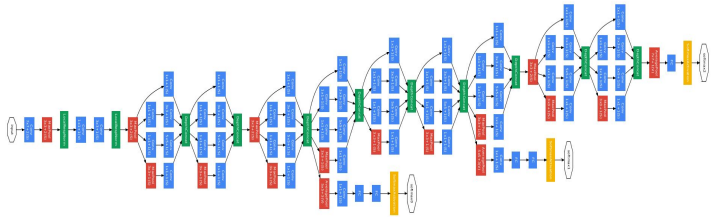
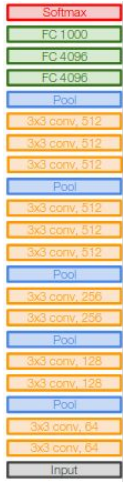


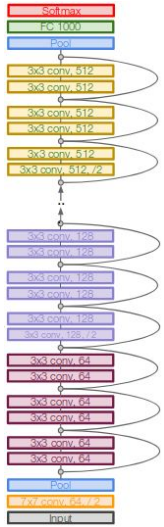
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.



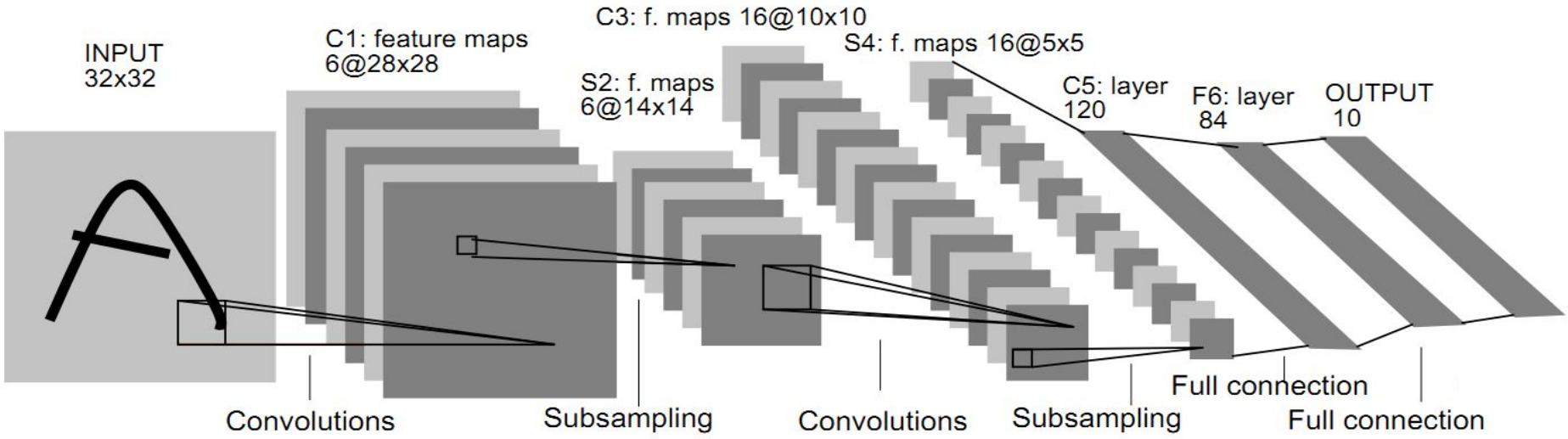








LeNet-5 [LeCun et al., 1998]



Convolution filters: 5x5 with stride 1

Subsampling (Pooling) layers: 2x2 with stride 2

[CONV-POOL-CONV-POOL-FC-FC]

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - **AlexNet (2012)**
  - ZFNet (2013)
  - VGGNet (2014)
  - GoogLeNet (2014)
  - ResNet (2015)

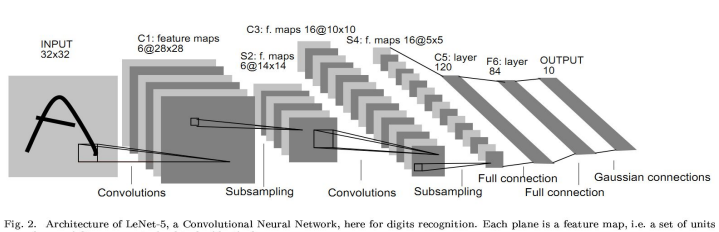
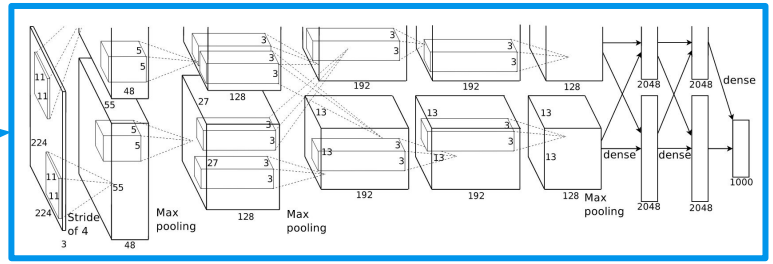
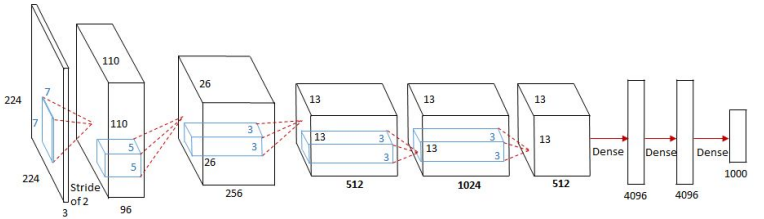
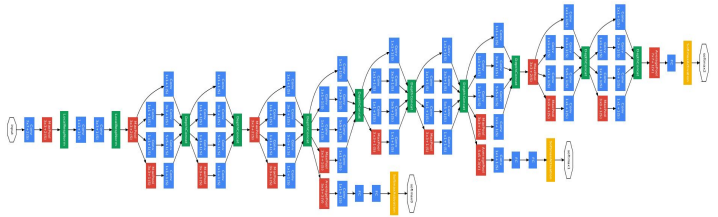
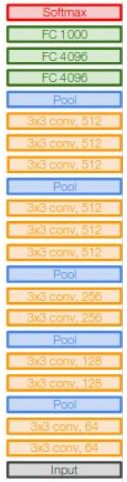


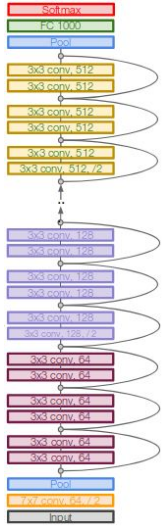
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

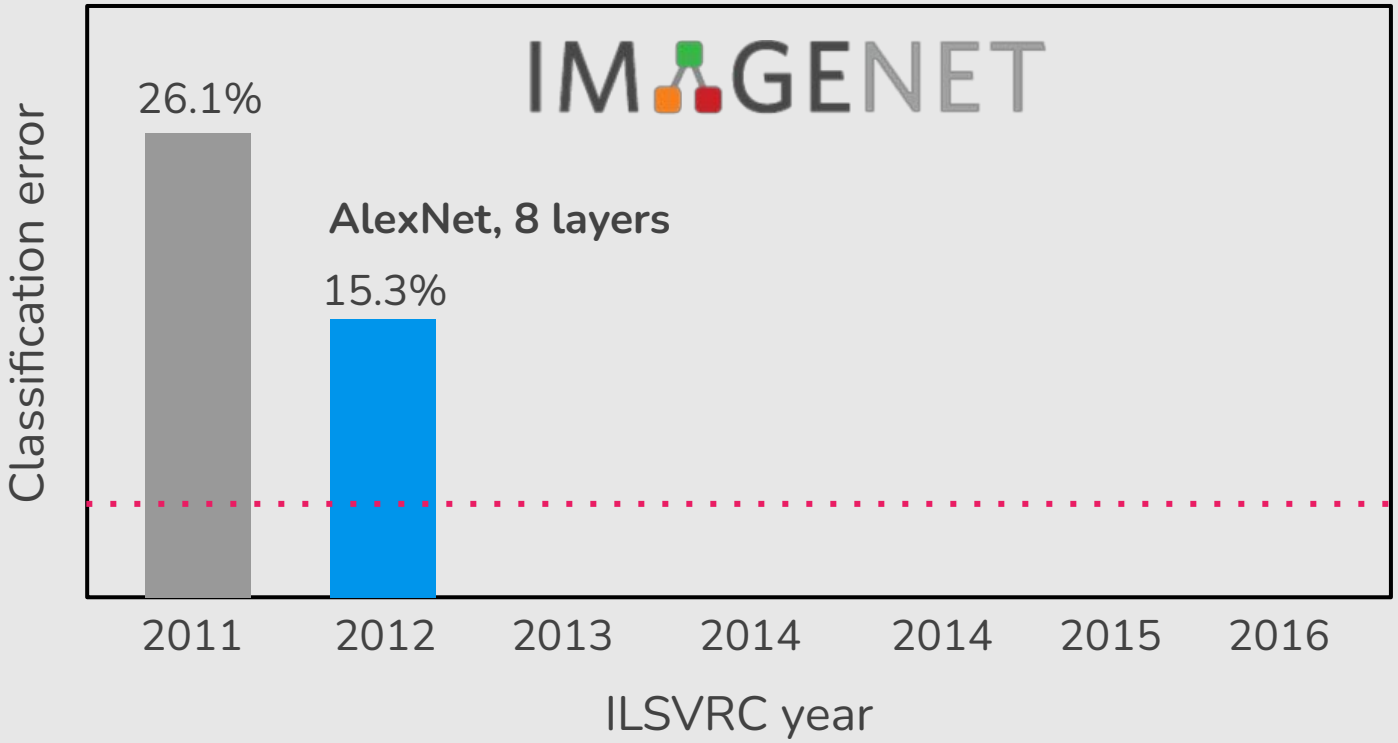






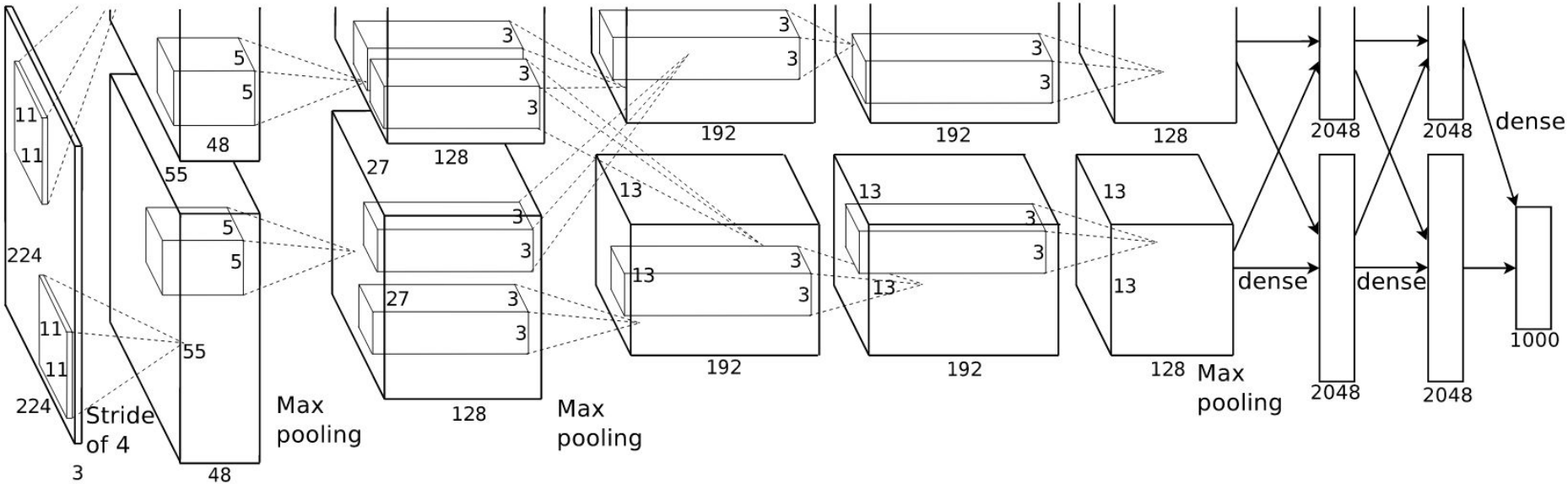






"ImageNet classification with deep convolutional neural networks". NIPS, 2012.

AlexNet [Krizhevsky et al., 2012]

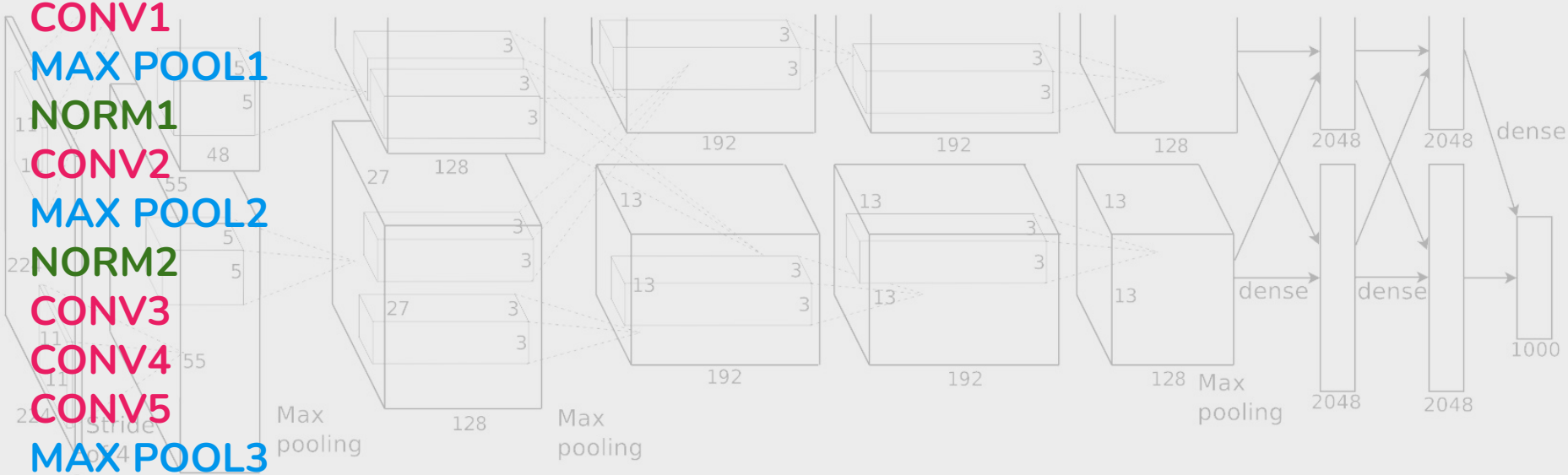


"ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.

AlexNet [Krizhevsky et al., 2012]

Architecture:

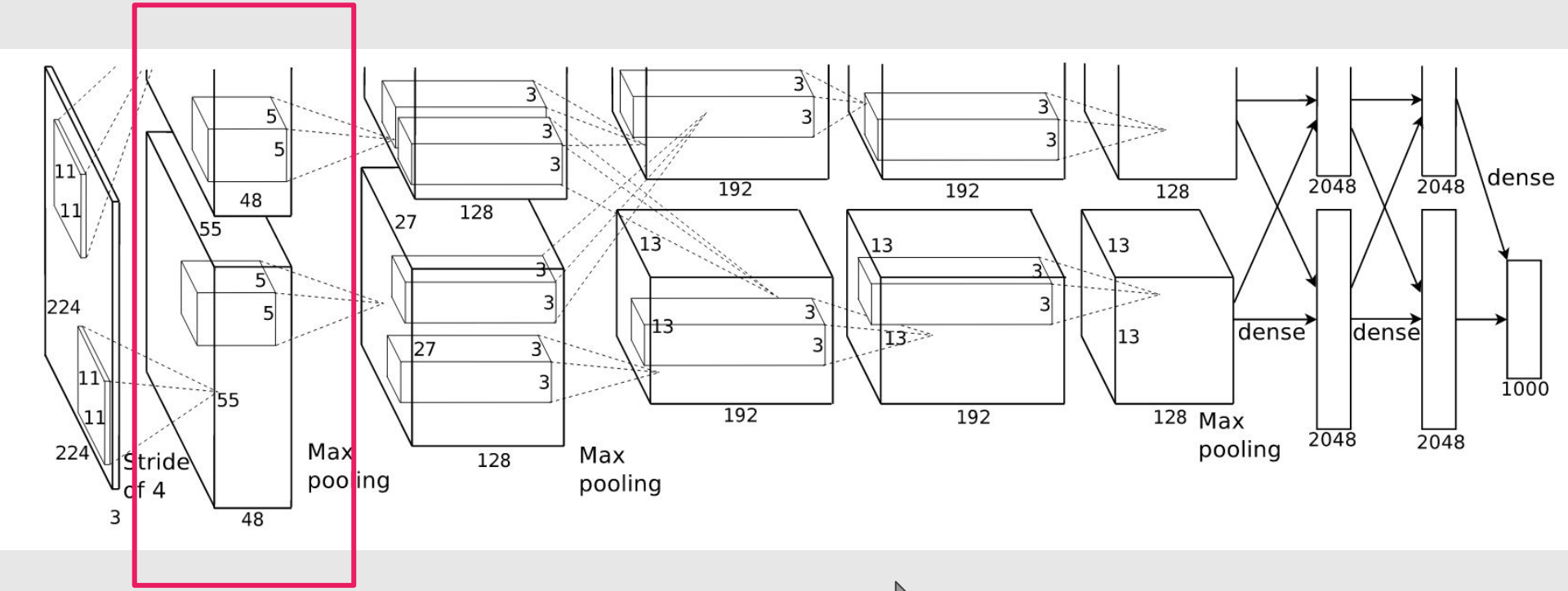
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
MAX POOL3
FC6
FC7
FC8



AlexNet [Krizhevsky et al., 2012]

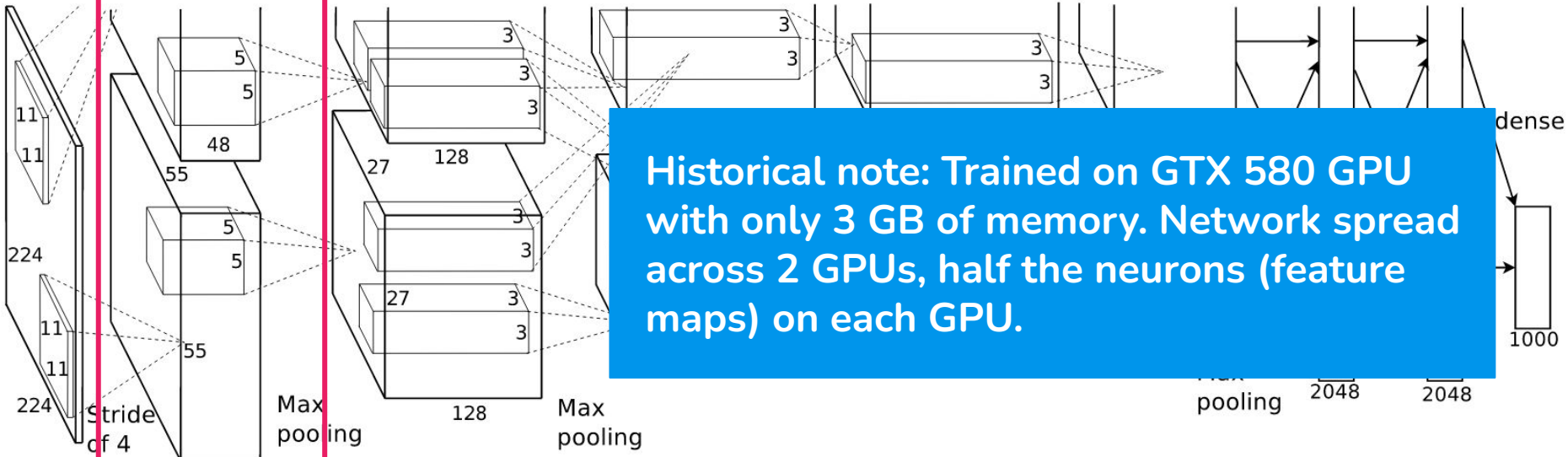
[227x227x3]	INPUT
[55x55x96]	**CONV1**: 96 11x11 filters at stride 4, pad 0
[27x27x96]	**MAX POOL1**: 3x3 filters at stride 2
[27x27x96]	**NORM1**: Normalization layer
[27x27x256]	**CONV2**: 256 5x5 filters at stride 1, pad 2
[13x13x256]	**MAX POOL2**: 3x3 filters at stride 2
[13x13x256]	**NORM2**: Normalization layer
[13x13x384]	**CONV3**: 384 3x3 filters at stride 1, pad 1
[13x13x384]	**CONV4**: 384 3x3 filters at stride 1, pad 1
[13x13x256]	**CONV5**: 256 3x3 filters at stride 1, pad 1
[6x6x256]	**MAX POOL3**: 3x3 filters at stride 2
[4096]	**FC6:** 4096 neurons
[4096]	**FC7:** 4096 neurons
[1000]	**FC8:** 1000 neurons (class scores)

AlexNet [Krizhevsky et al., 2012]



[55x55x96] **CONV1** ➡ [55x55x48] x 2

AlexNet [Krizhevsky et al., 2012]



Historical note: Trained on GTX 580 GPU with only 3 GB of memory. Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

[55x55x96] **CONV1** ⇨ [55x55x48] x 2

AlexNet [Krizhevsky et al., 2012]

Details:

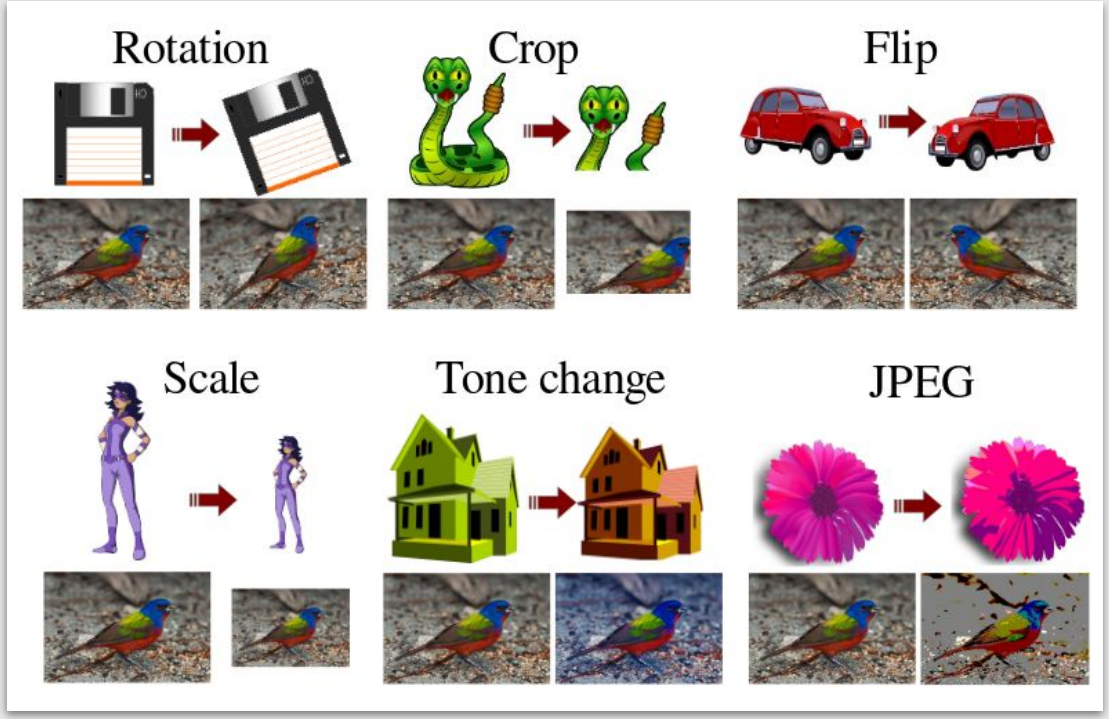
- 60 million learned parameters
- first use of ReLU
- used Norm layers
- heavy data augmentation
- dropout 0.5
- batch size 128
- 7 CNN ensemble: 18.2% -> 15.3%
- 5-6 days to train on 2 GTX 580 3GB GPUs

AlexNet [Krizhevsky et al., 2012]

Details:

- 60 million learned parameters
- first use of ReLU
- used Norm layers
- heavy **data augmentation**
- dropout 0.5
- batch size 128
- 7 CNN ensemble: 18.2% -> 15.3%
- 5-6 days to train on 2 GTX 580 3GB GPUs

Data Augmentation



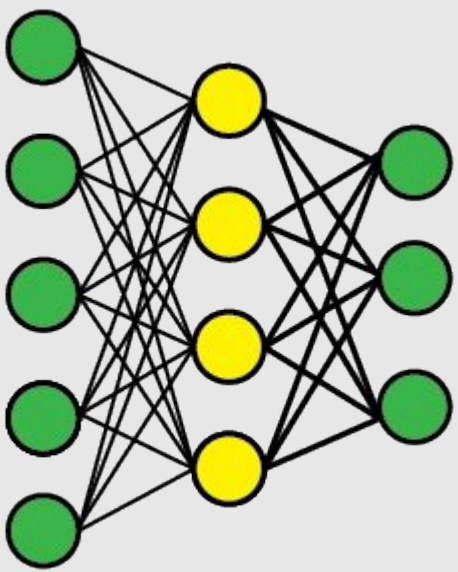
"Transformation Pursuit for Image Classification", CVPR 2014.

AlexNet [Krizhevsky et al., 2012]

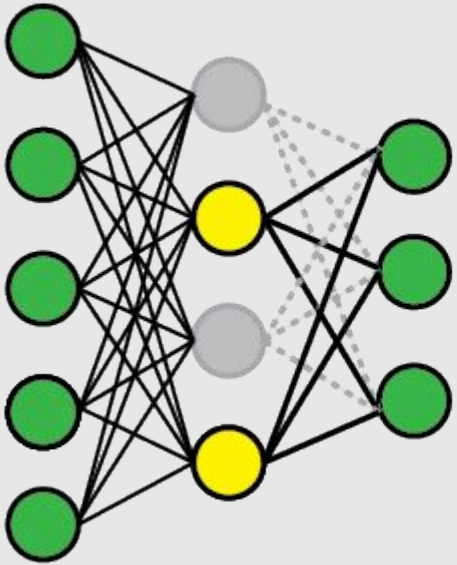
Details:

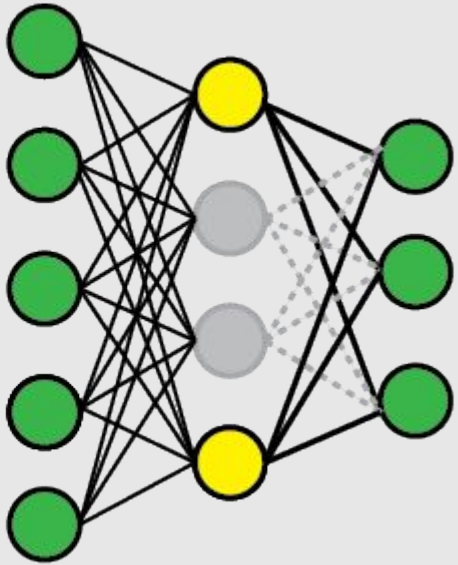
- 60 million learned parameters
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- **dropout** 0.5
- batch size 128
- 7 CNN ensemble: 18.2% -> 15.3%

Dropout [Hinton et al., 2012]



Standard Network





After applying dropout

"Dropout: A Simple Way to Prevent Neural Networks from Overfitting", https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

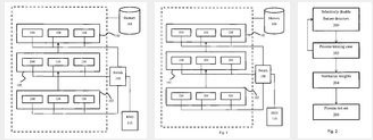
Google Patents

System and method for addressing overfitting in a neural network

Abstract

A system for training a neural network. A switch is linked to feature detectors in at least some of the layers of the neural network. For each training case, the switch randomly selectively disables each of the feature detectors in accordance with a preconfigured probability. The weights from each training case are then normalized for applying the neural network to test data.

Images (3)



Classifications

G06N3/084 Back-propagation

View 4 more classifications

US9406017B2

United States

Download PDF | Find Prior Art | Similar

Inventor: Geoffrey E. Hinton, Alexander Krizhevsky, Ilya Sutskever, Nitish Srivastva

Current Assignee : Google LLC

Worldwide applications

2013 · US WO BR AU 2016 · US

Application US14/015,768 events

2012-12-24 • Priority to US201261745711P

2013-08-30 • Application filed by Google LLC

2014-06-26 • Publication of US20140180986A1

2016-08-02 • Publication of US9406017B2

2016-08-02 • Application granted

2019-10-10 • Application status is Active

2034-09-03 • Adjusted expiration

Show all events

Info: Non-patent citations (24), Cited by (13), Legal events, Similar documents, Priority and Related Applications

External links: USPTO, USPTO Assignment, Espacenet, Global Dossier, Discuss

Description

CROSS-REFERENCE TO RELATED APPLICATION

Claims (24)

Hide Dependent

What is claimed is:

https://patents.google.com/patent/US9406017B2/en

AlexNet [Krizhevsky et al., 2012]

Details:

- 60 million learned parameters
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- 7 CNN **ensemble**: 18.2% -> 15.3%

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - **ZFNet (2013)**
  - VGGNet (2014)
  - GoogLeNet (2014)
  - ResNet (2015)

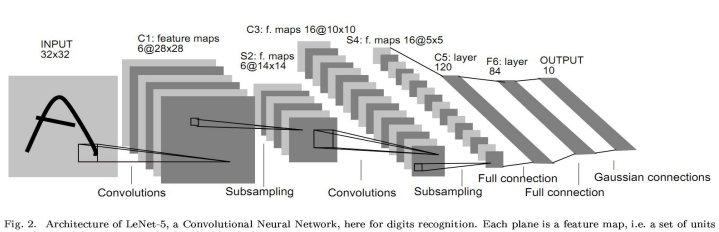
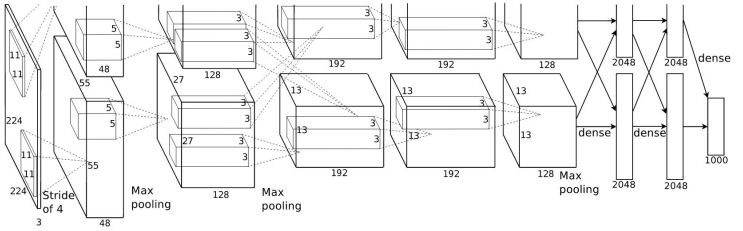
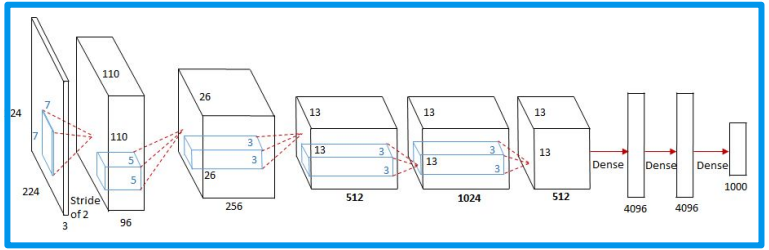
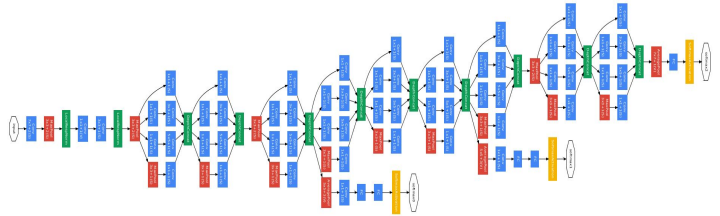
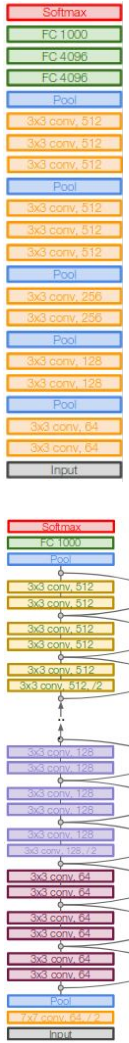


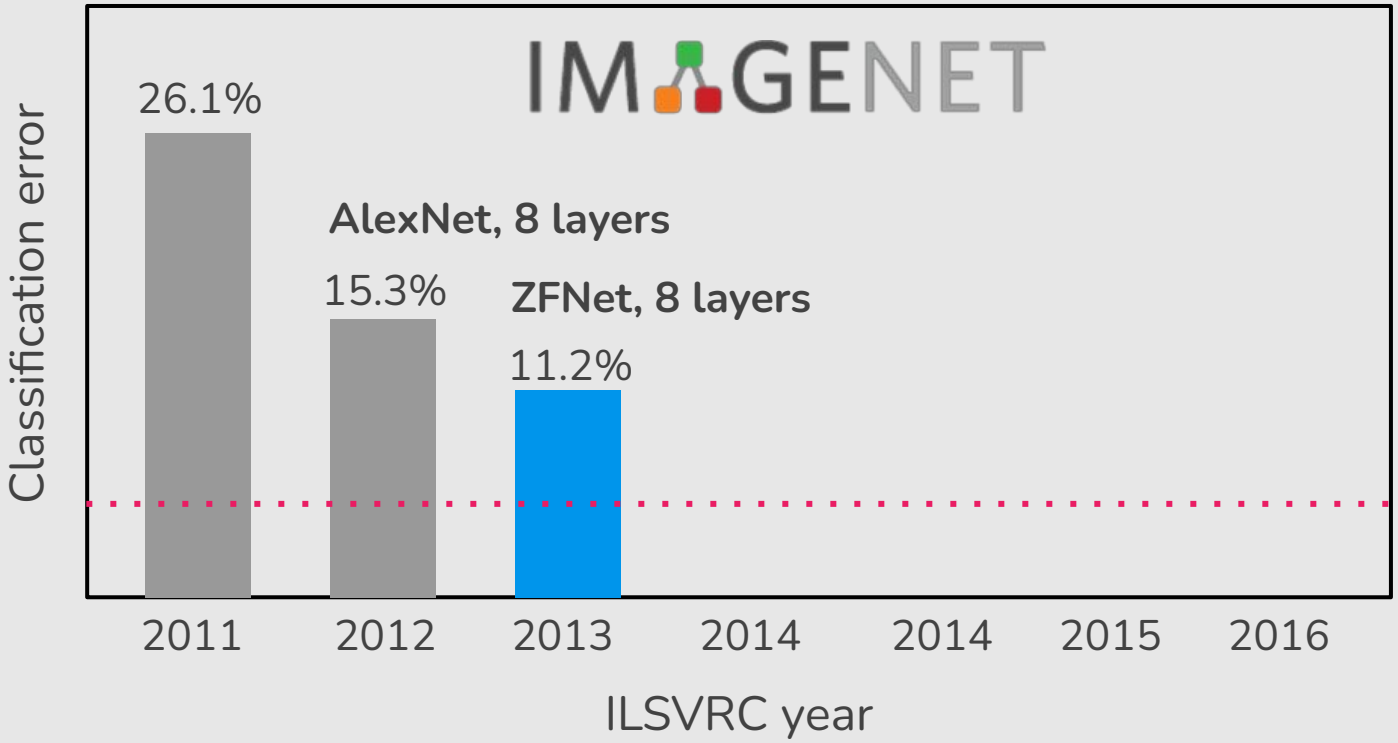
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.





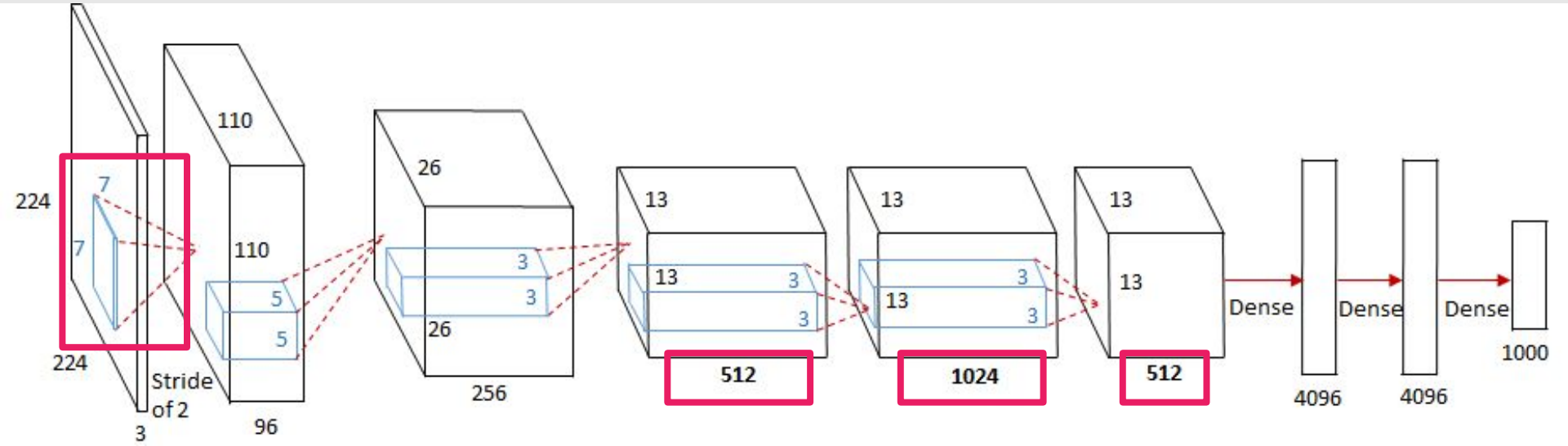






"Visualizing and Understanding Convolutional Networks", ECCV 2014.

ZFNet [Zeiler & Fergus, 2013]



AlexNet but:

CONV1: change from (11x11 stride 4) to (7x7 stride 2)

CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - **VGGNet (2014)**
  - GoogLeNet (2014)
  - ResNet (2015)

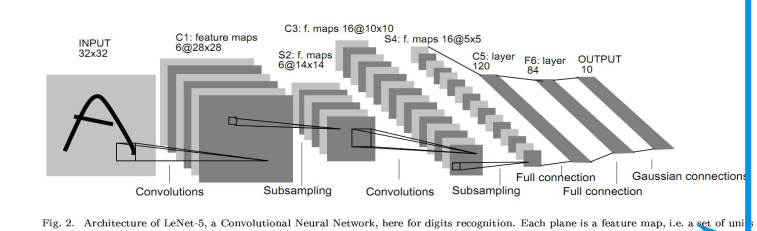
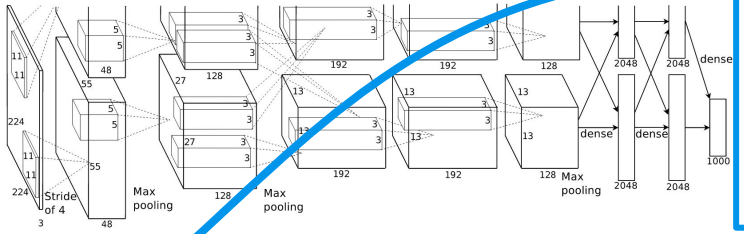
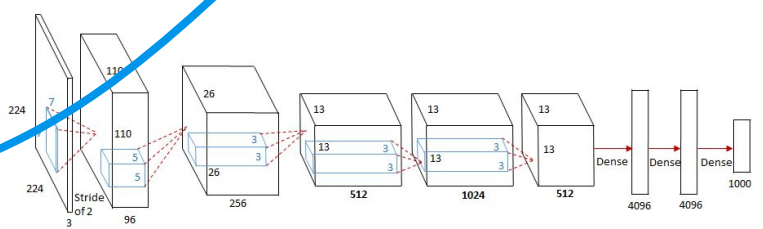
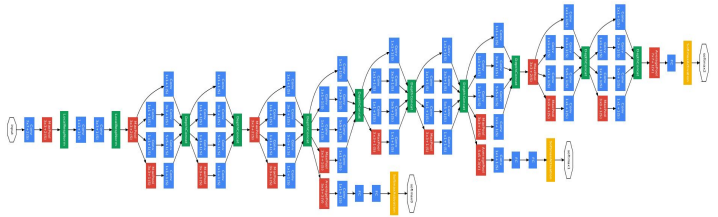
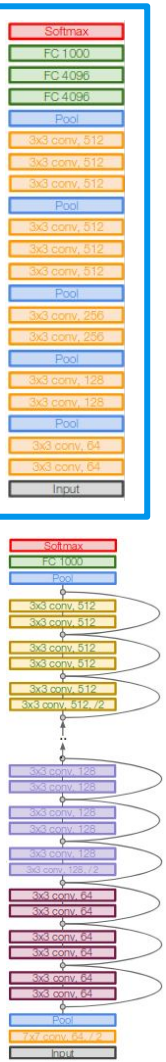


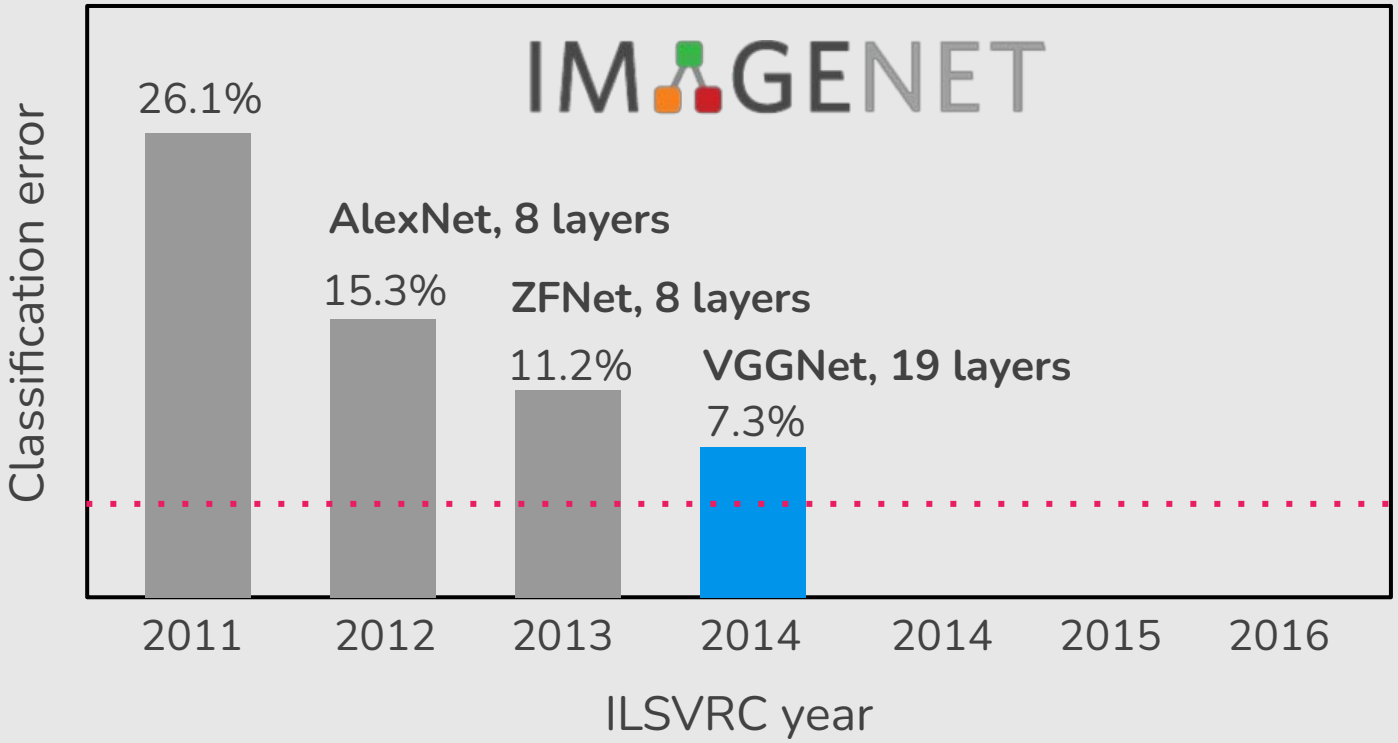
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.











"Very Deep Convolutional Networks for Large-Scale Image Recognition", https://arxiv.org/pdf/1409.1556

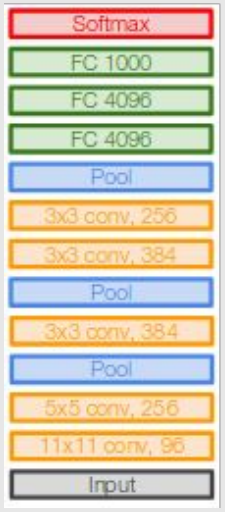
VGGNet [Simonyan & Zisserman, 2014]

Small filters, Deeper networks

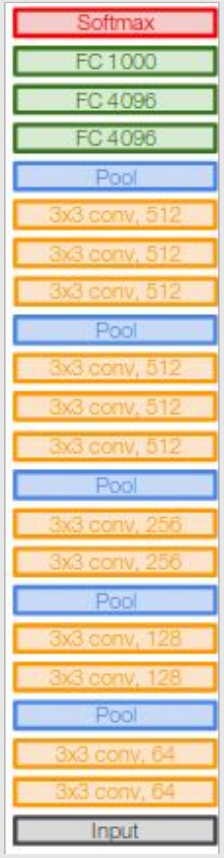
8 layers (AlexNet)
16-19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

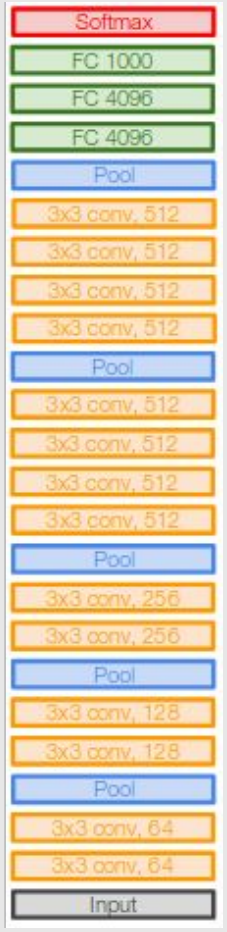
11.2% in ILSVRC'13 (ZFNet)
7.3% in ILSVRC'14



AlexNet



VGG16

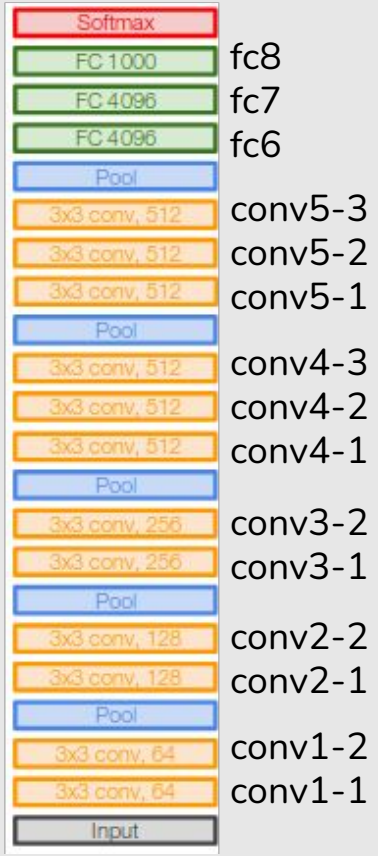


VGG19

VGGNet [Simonyan & Zisserman, 2014]

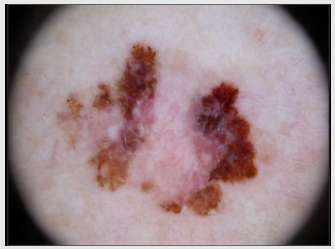
Details:

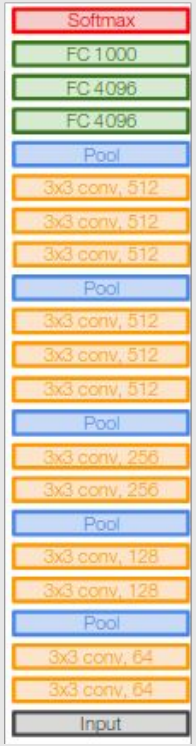
- 138M parameters
- 2nd in classification, 1st in localization
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks



VGG16

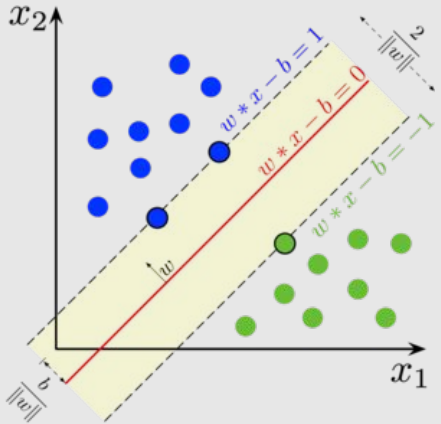
VGGNet [Simonyan & Zisserman, 2014]





[0.01 0.8 1 0.5 … 0.3 0.07 0 0.4 0.6 0 0]
4096-d

Train a classifier (e.g., SVM)



VGG as Feature Extractor

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - **GoogLeNet (2014)**
  - ResNet (2015)

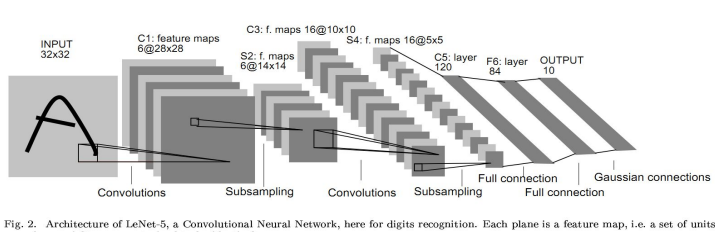
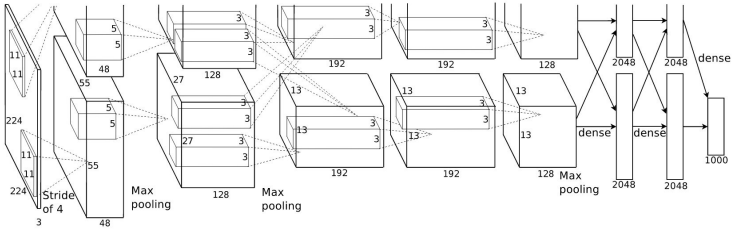
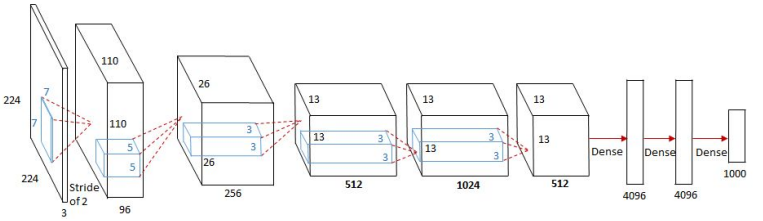
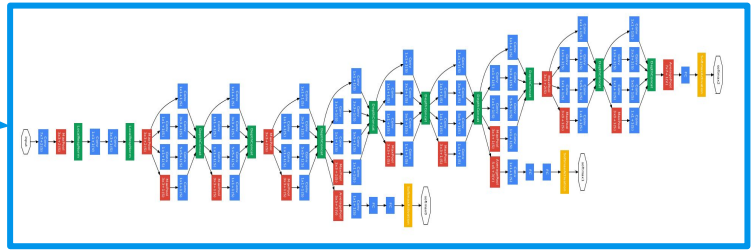
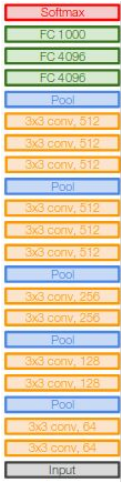


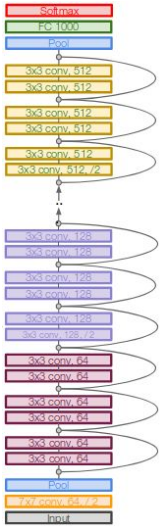
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.











To be continued ...