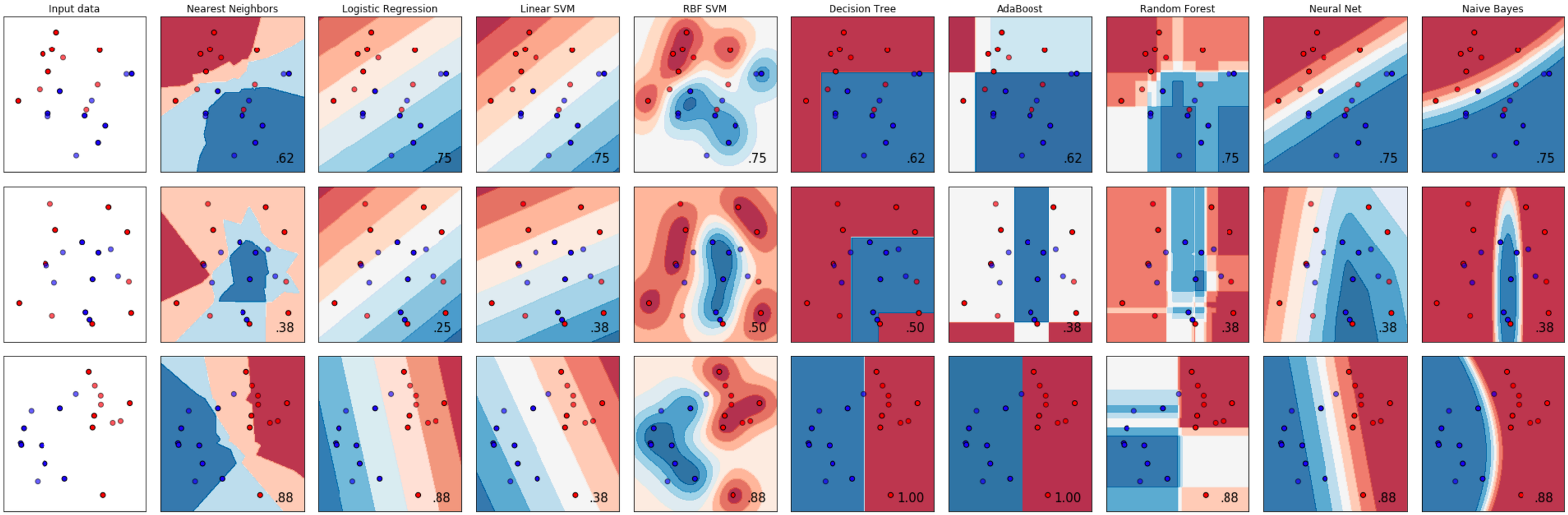
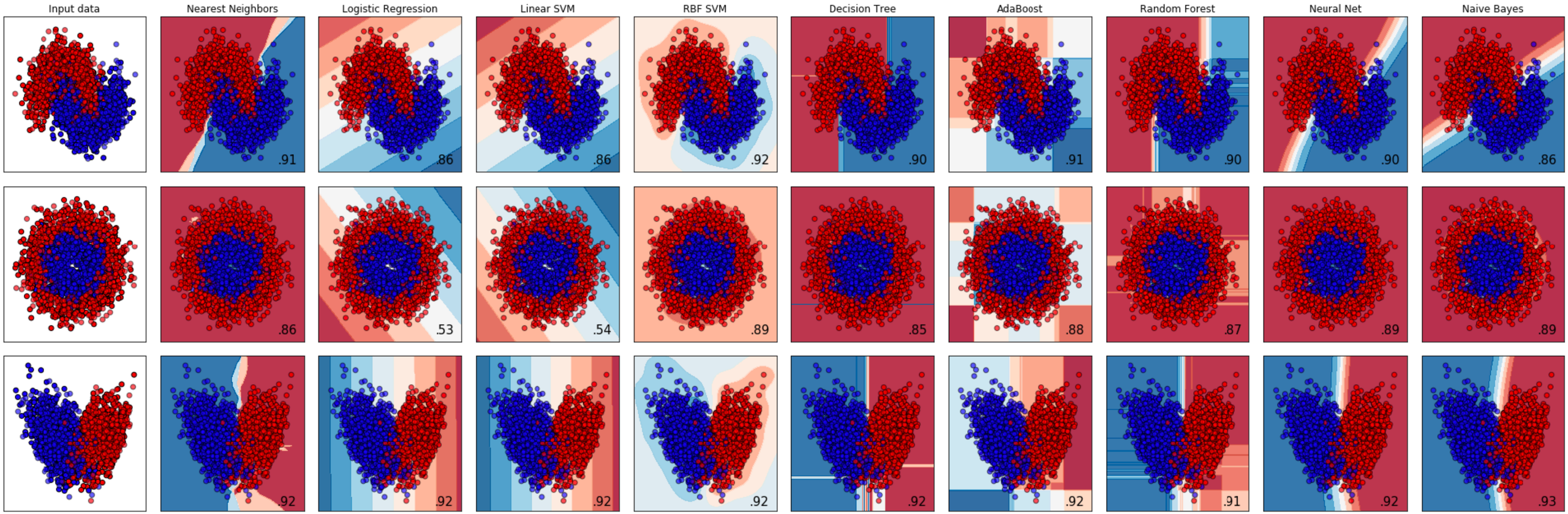
Machine Learning
Part 2

UNIVERSITAT DE BARCELONA

Classification Methods



Classification Methods



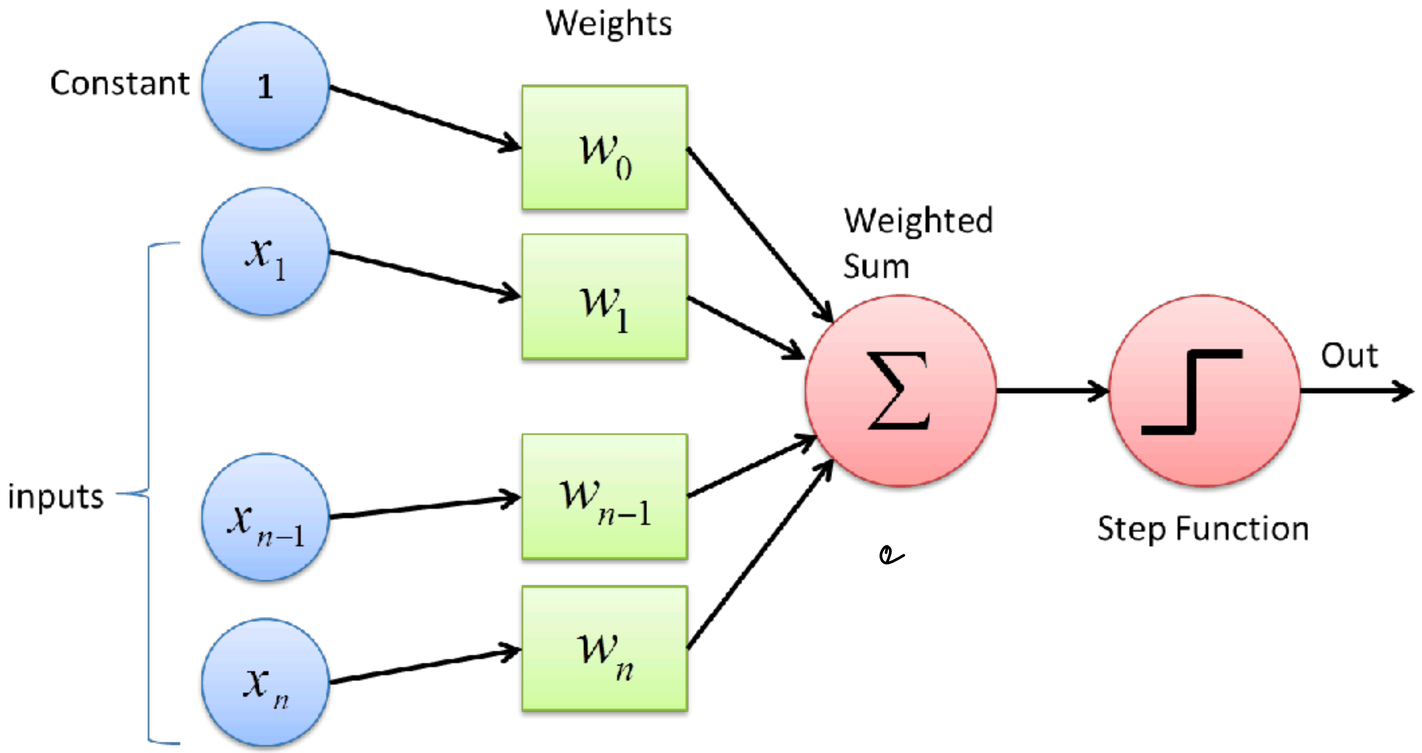
ML MODELS

Neural Networks

Nivell del cos u
Nivell del cos dos
Nivell del cos tres
Nivell del cos quatre
Nivell del cos cinc

What is a Neural Network

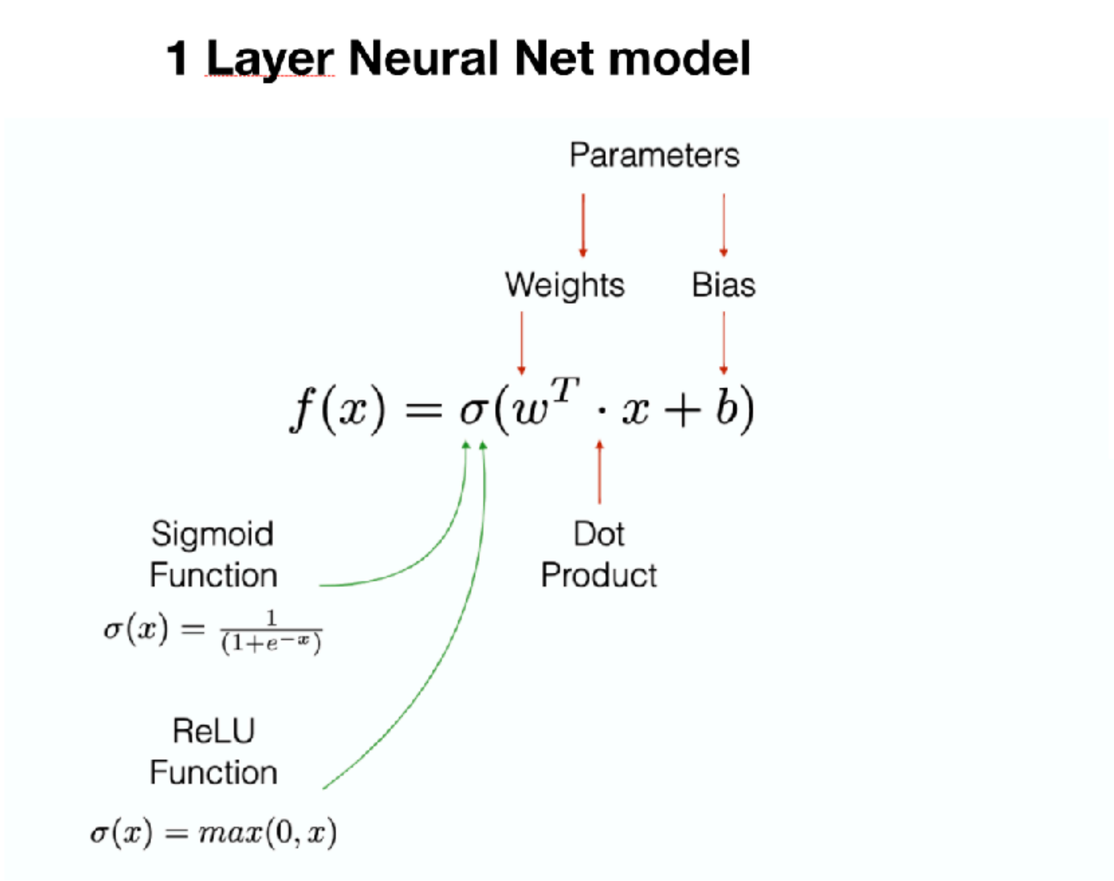
- It is a system **biologically inspired** that tries to emulate **human brain**.
- **Why** is it a good idea to try to **emulate** the **brain** when solving a recognition task?

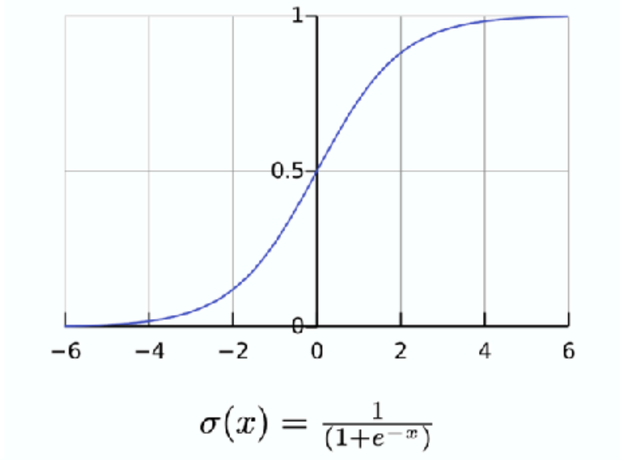


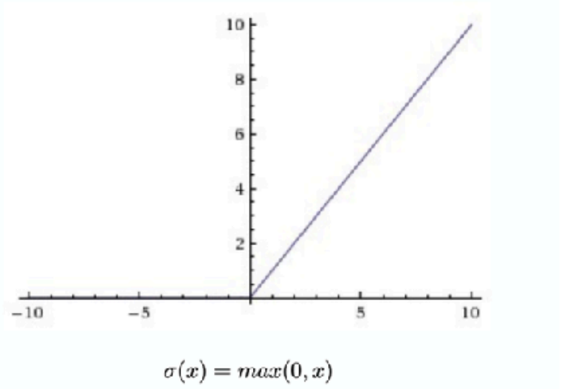
Perceptron

Perhaps the first AI model

Neural Networks

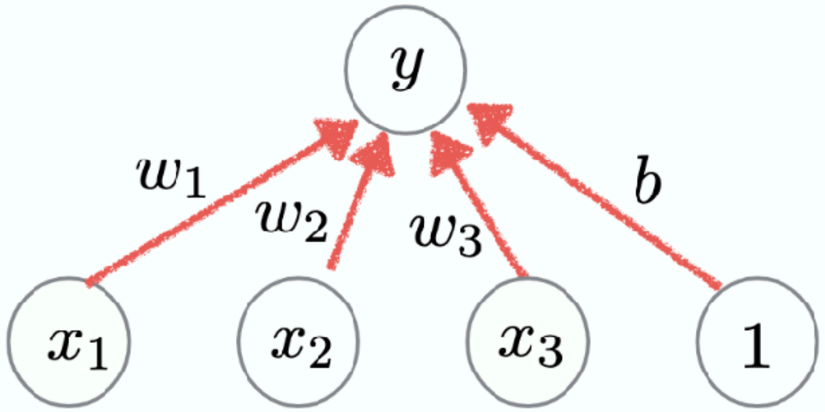




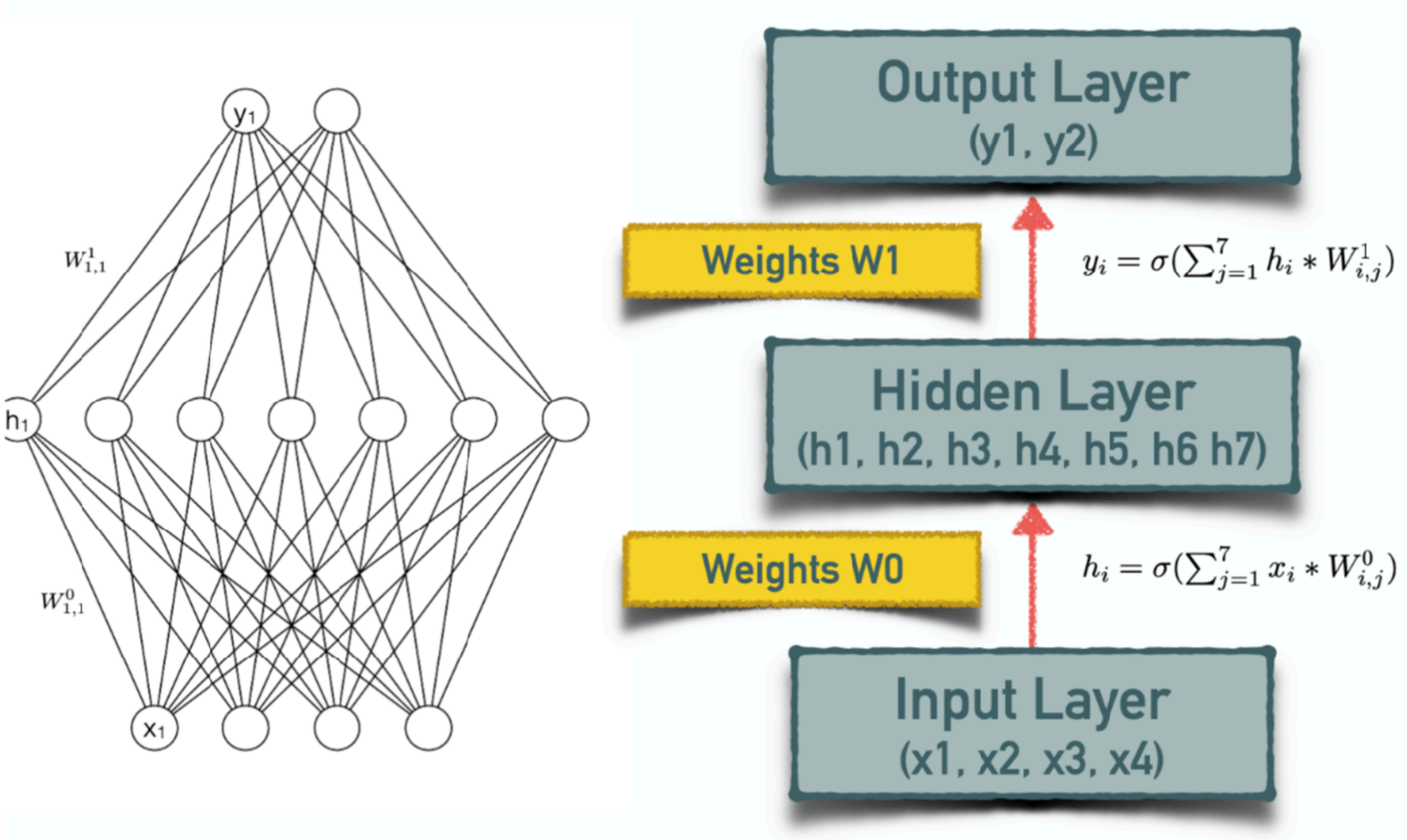


1 Layer Neural Net model

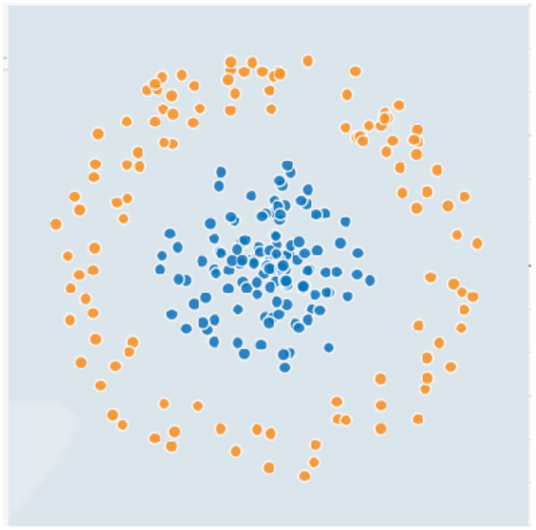
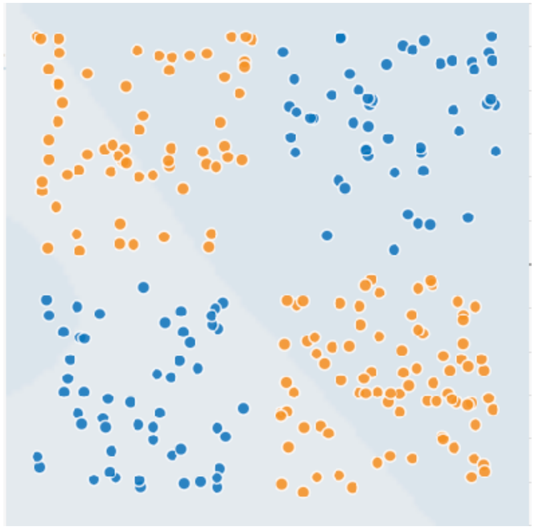
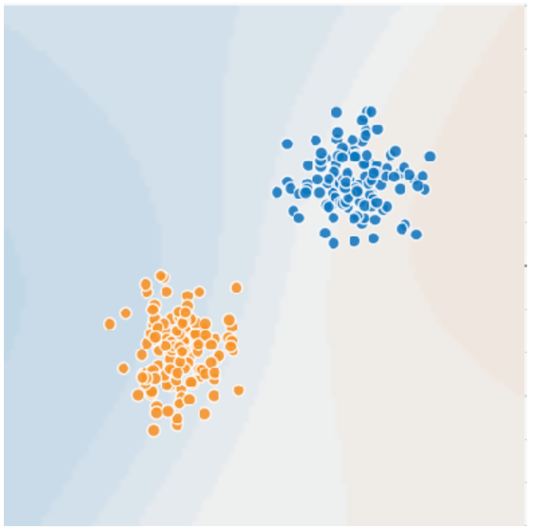
$$f(x) = \sigma(w^T \cdot x + b)$$



Graphical Representation

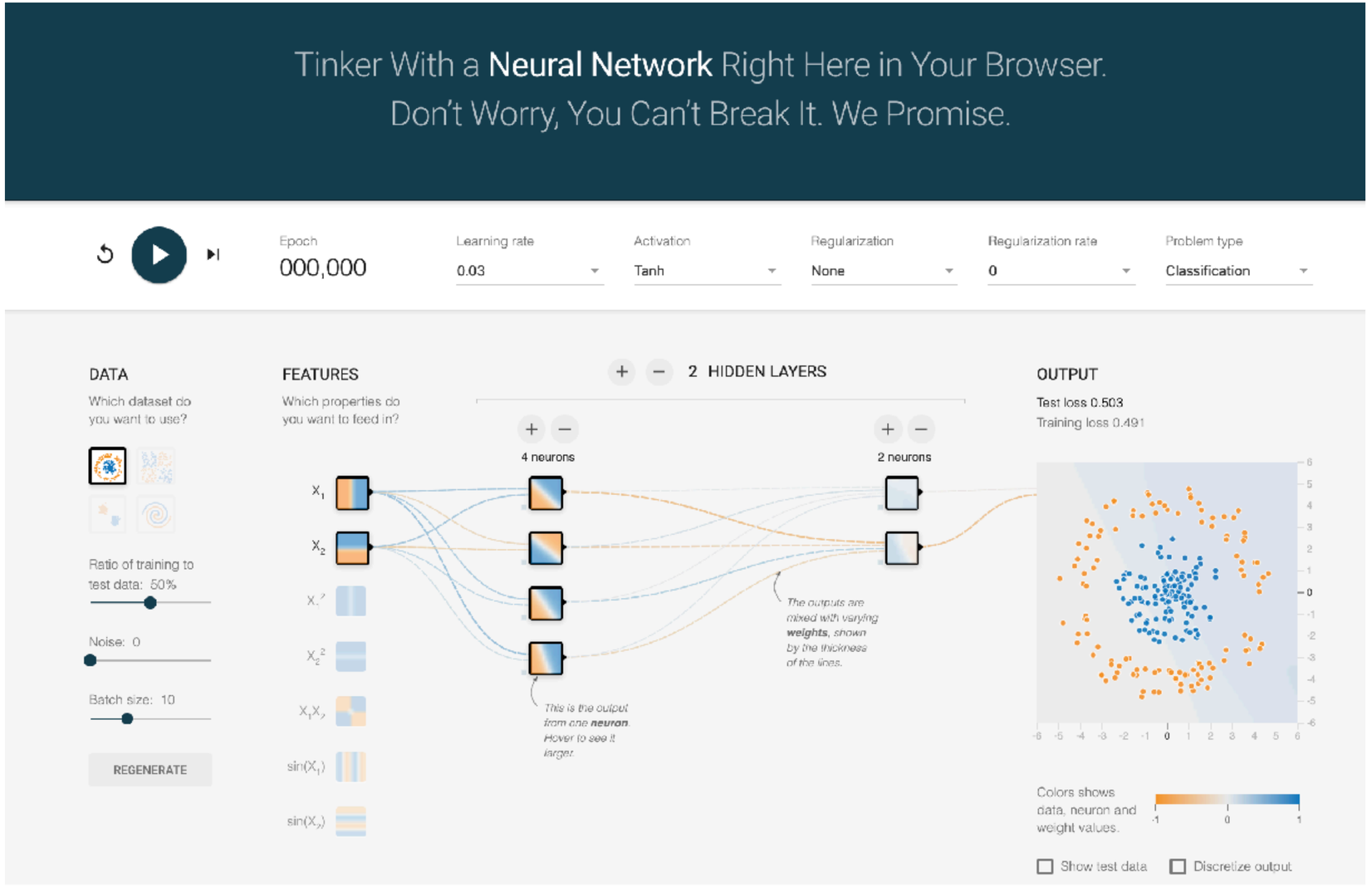


2 Layers Neural Net model

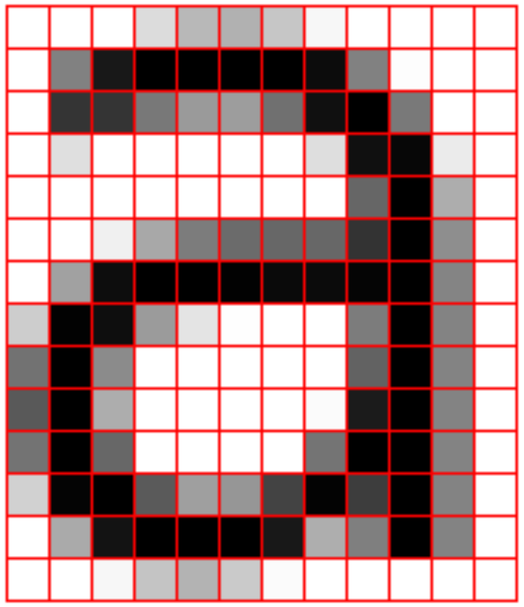

How can we create a Neural Network able to deal with these datasets?

https://playground.tensorflow.org/



Neural Networks for Images?

An image is a matrix of size **`mxnxc`** pixels



1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0

Toy Dataset: Fashion MNIST

Toy Dataset: Fashion MNIST

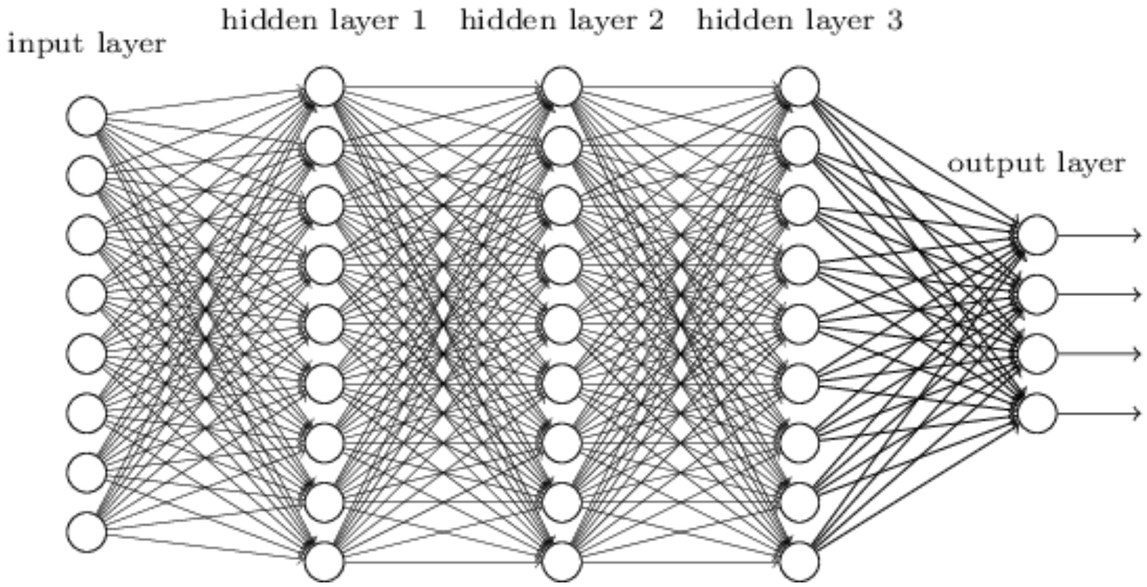
```
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
```

this is equivalent to this

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(300, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(10, activation="softmax")
])
```

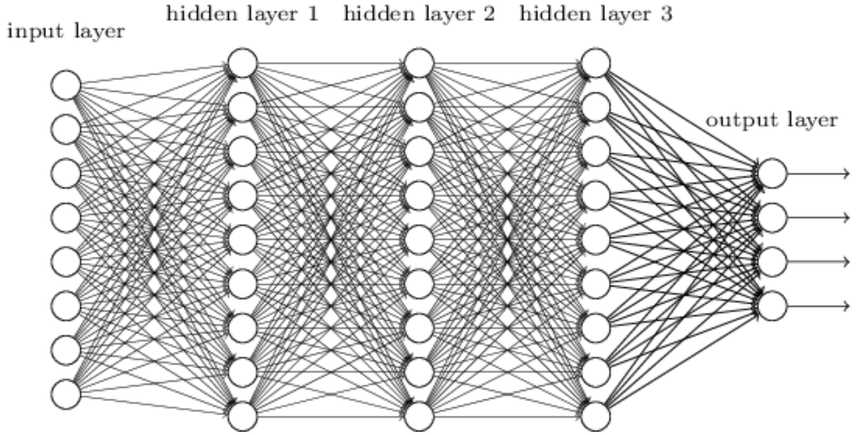
hands on!

Neural Networks for Images?

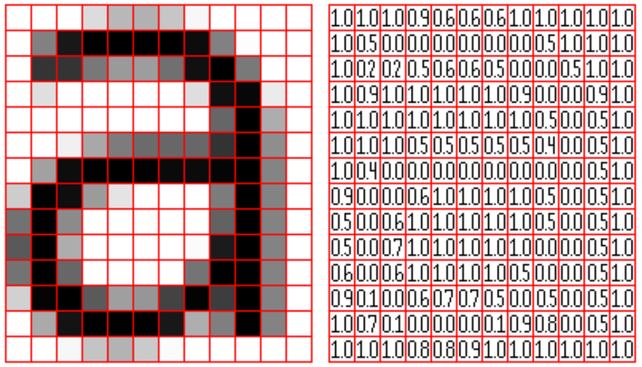


How many parameter does this MLP has?

Neural Networks for Images?



What happens if your data is an image like this one?



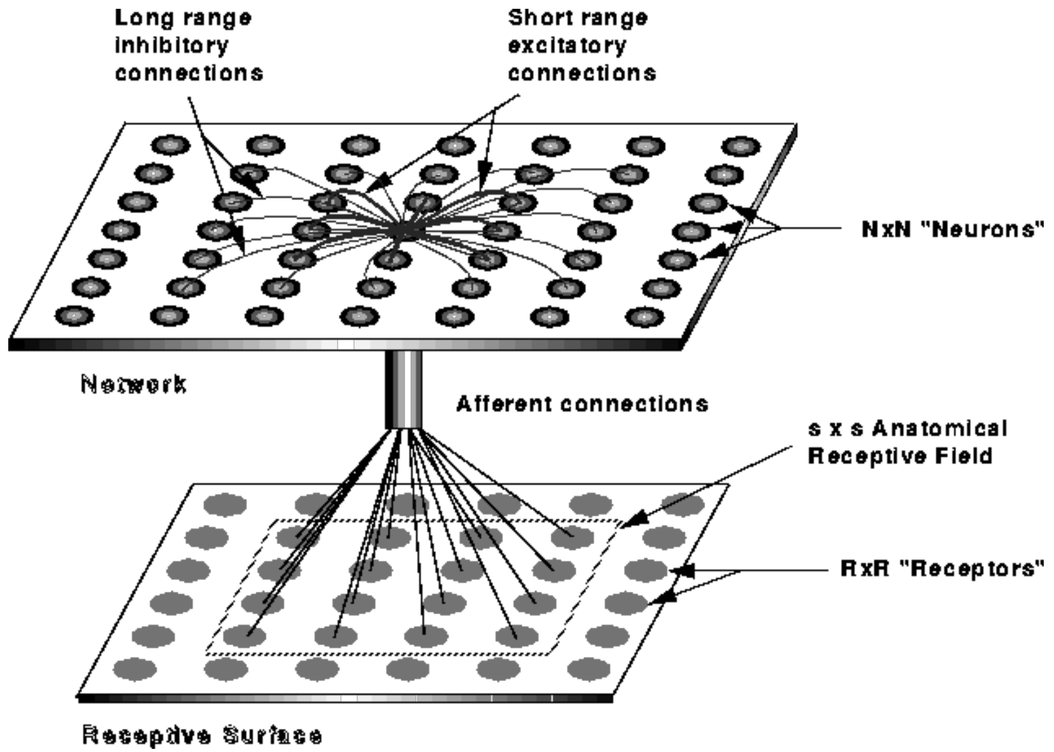
1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0		
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0	
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.0	0.5	1.0	1.0	
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.0	0.9	1.0	
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0		
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0		
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0		
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0		
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0		
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0		
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0		
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0		
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0		
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0		

Neural Networks for Images?

Neural Networks for Images

- Input is a standard vector of size N x M x C
  - Imagine an medium resolution color image of 256x256 pixels
  - If we think on a Multi Layer Perceptron with just one hidden layer of 256 neurons + an output layer of 1 neuron it will have more than **48 million** parameters.
    - **Does it make sense? Can we do it better?**

Local Receptive Fields



David Hunter Hubel and Torsten Nils Wiesel, 1968

But, in an image:

A dog can appear **anywhere** in the image!

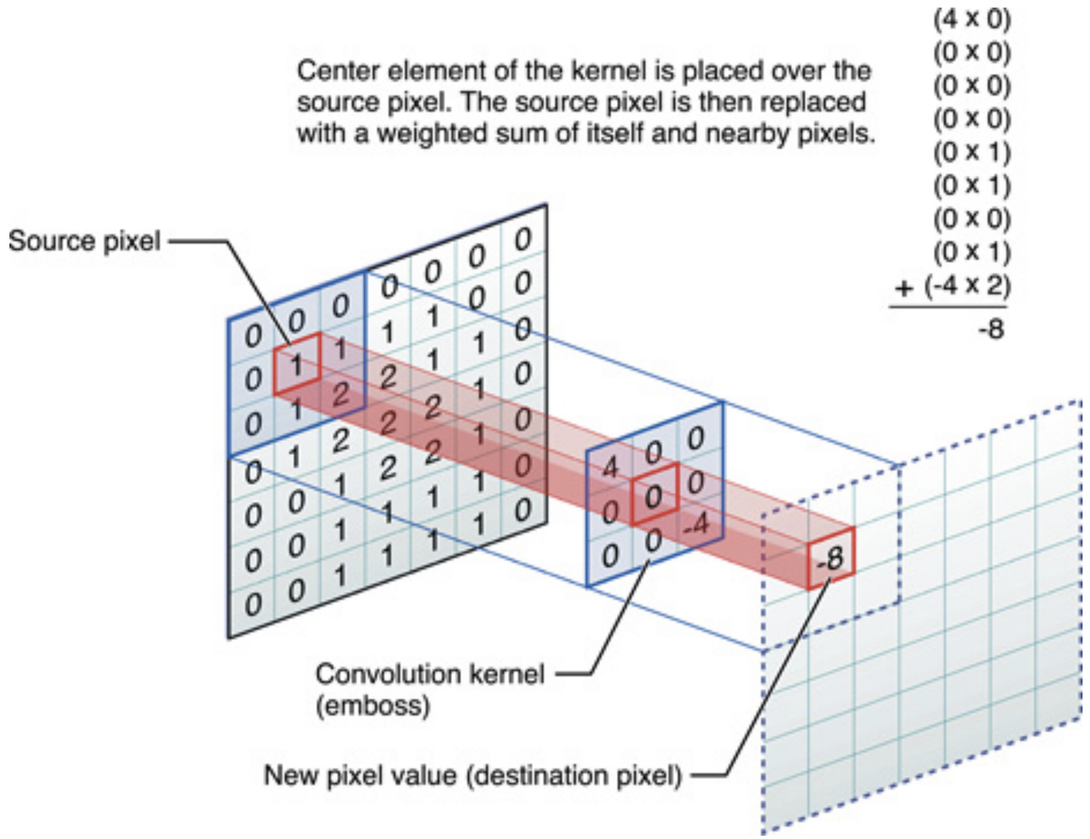
Doesn’t matter where they appear,
they look similar anywhere!

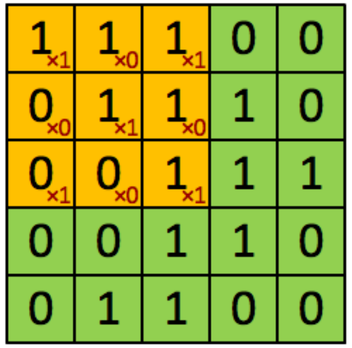
Convolutional Neural Networks (CNNs)

- Three main ideas:

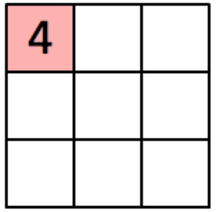
    1. **local receptive fields**,
    2. **shared weights**,
    3. **sub-sampling**

What is an image convolution?





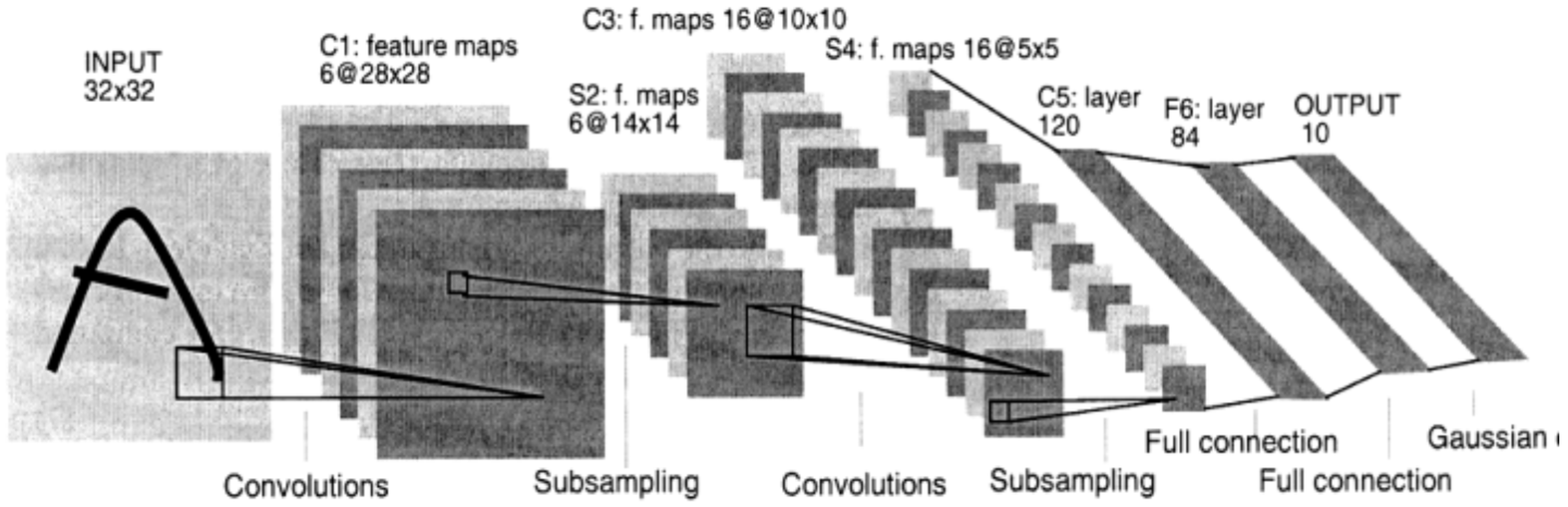
Image



Convolved Feature

What is an image convolution?

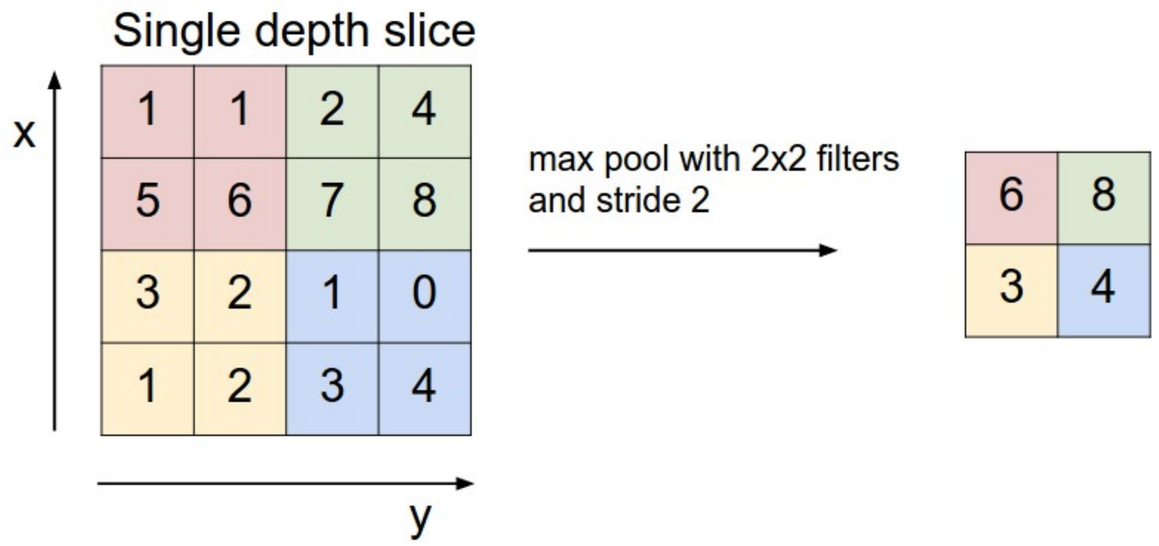
"Nothing New"



LeCun et al. 1998

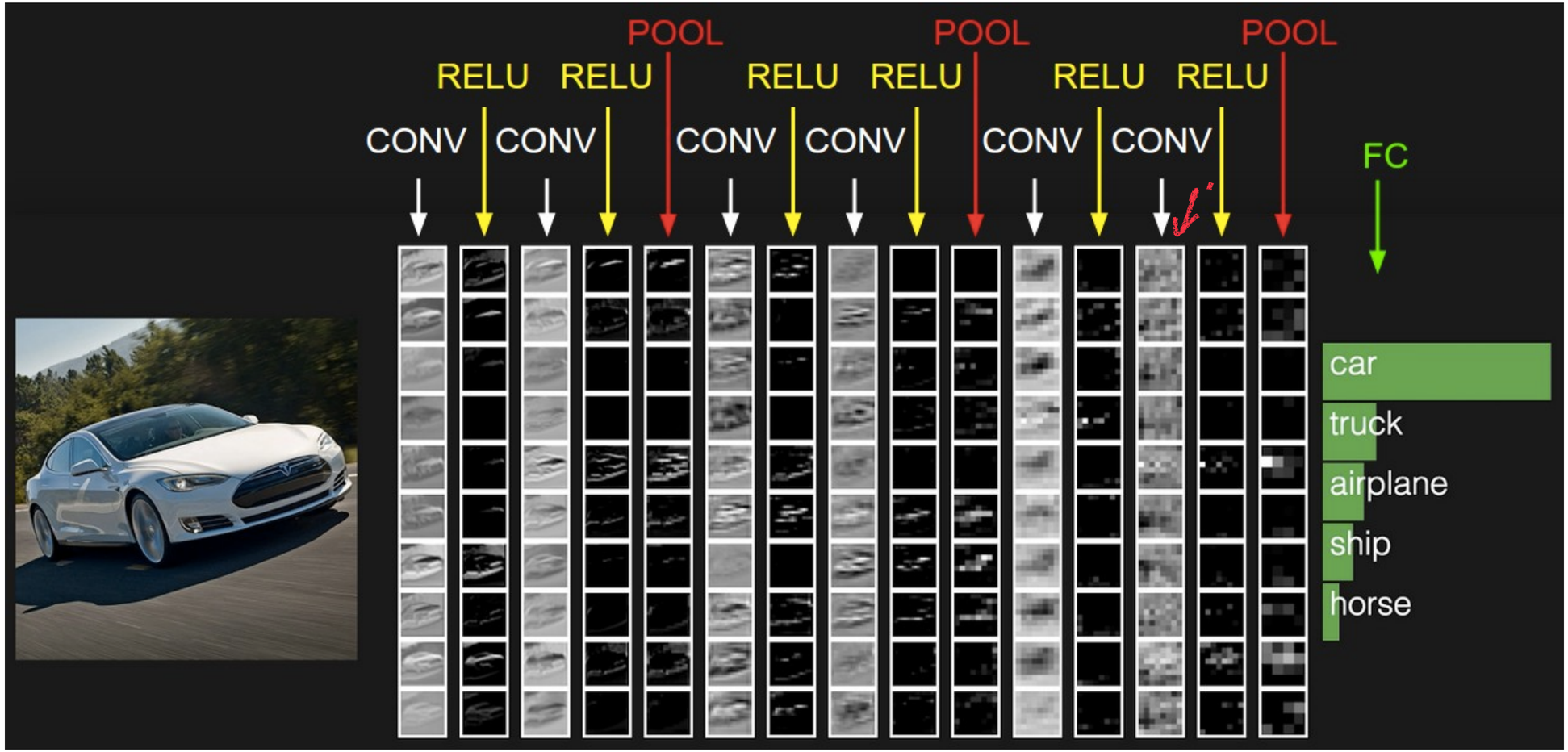
Max pooling

Pooling is a way of sub-sampling, i.e. reducing the dimension of the input (or at some hidden layer).
It is usually done after some of the convolutional layers



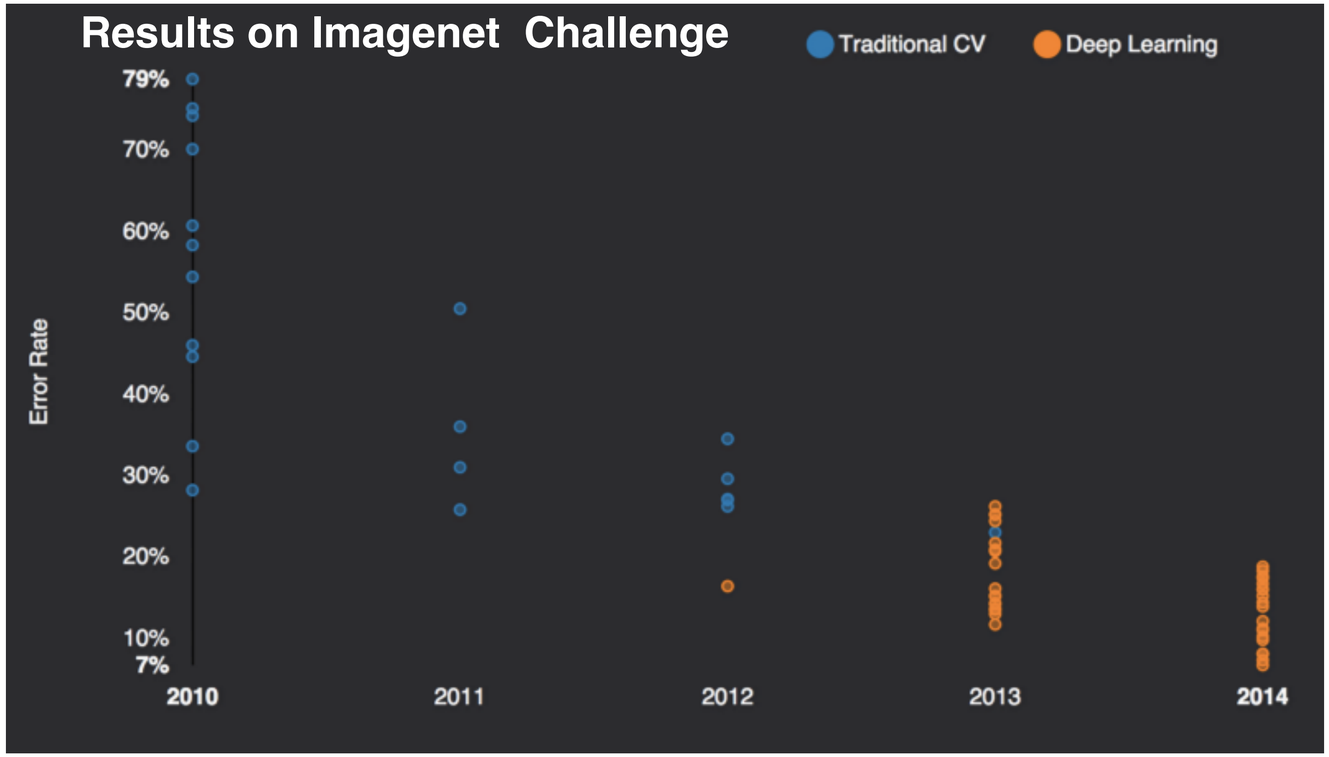
But it is also useful since it provides a form of translation **invariance**

Finally..


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

Convolutional Neural Networks (CNNs)

In computer Vision the breakthrough resulted in 2011 when Ciresan et.al introduced an algorithm to train these networks by using graphical cards (GPUs)



AlexNet

Similar framework to LeCun'98 but:

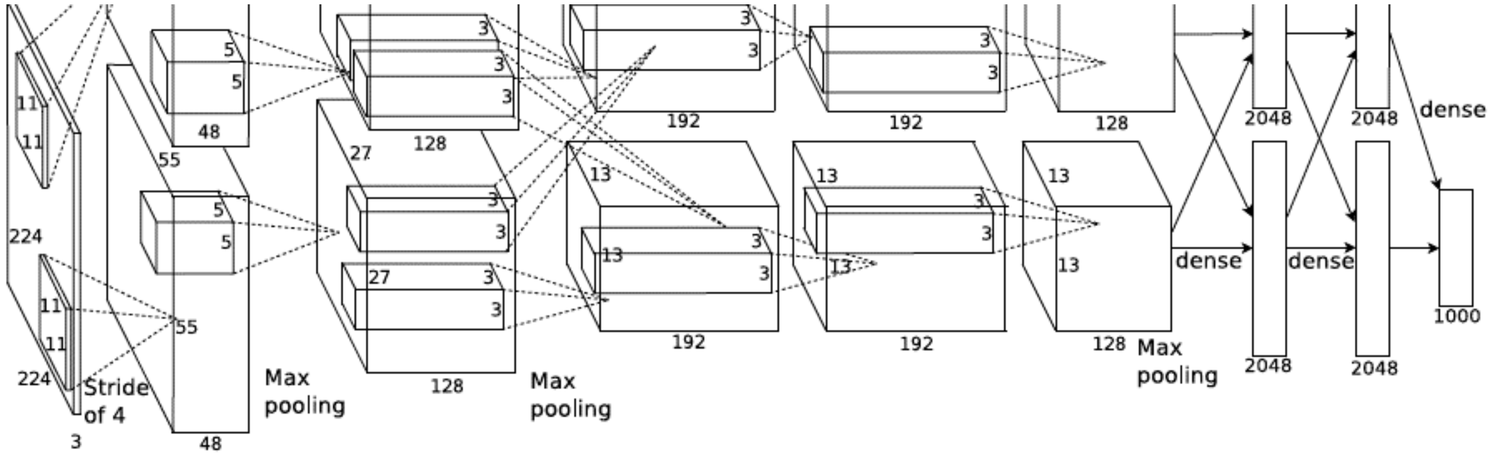
Bigger model:

7 hidden layers, 650.000 units, 60 million parameter

More Data:

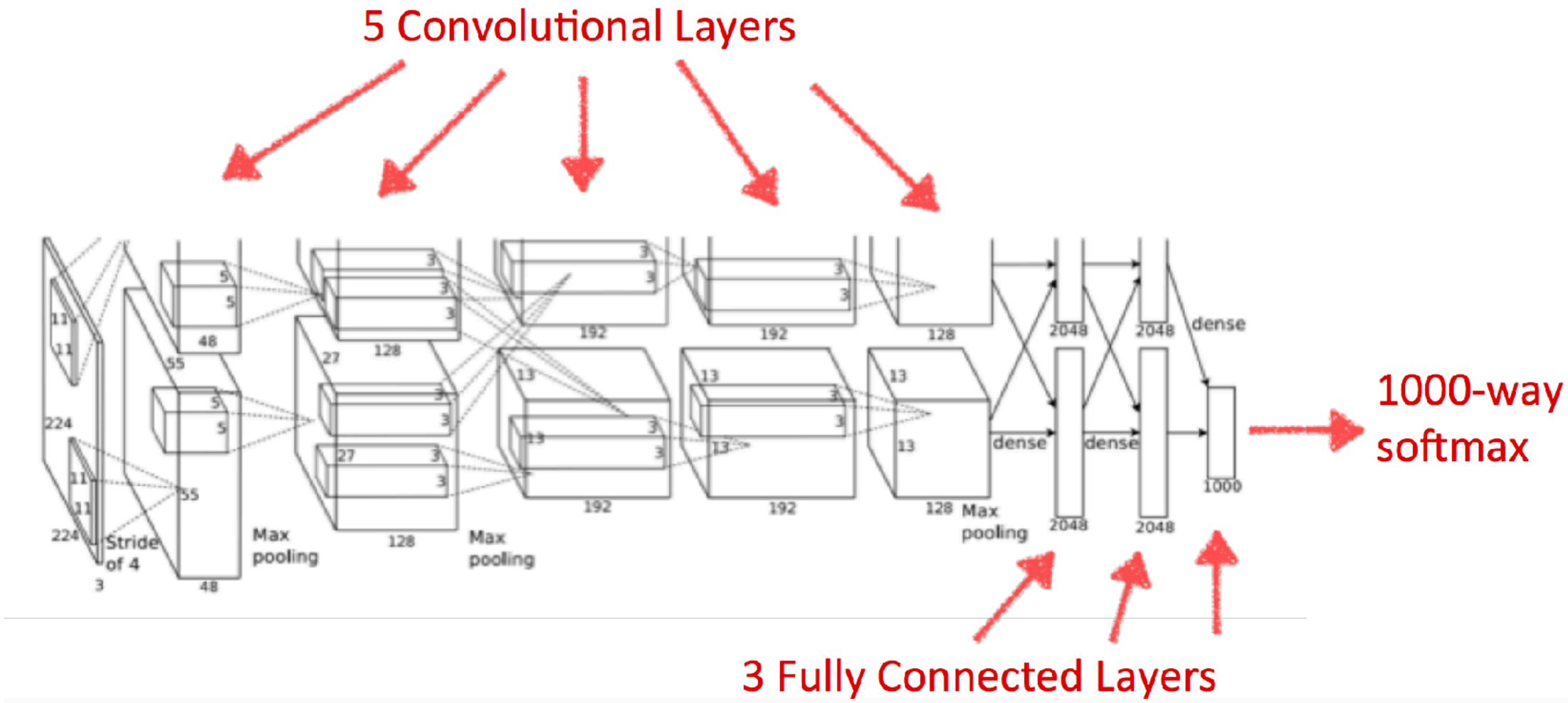
10^6 vs 10^3 images

GPU implementation (50x speedup over CPU)

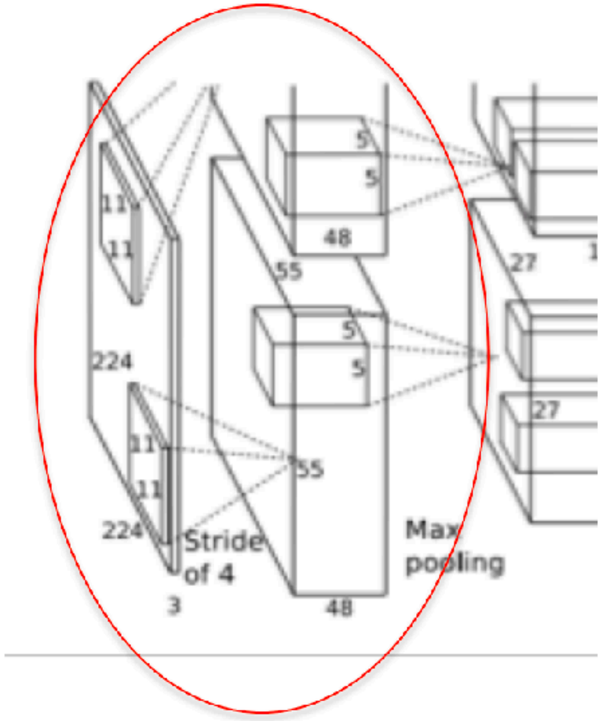


ImageNet Classification with Deep Convolutional Neural Networks. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

AlexNet

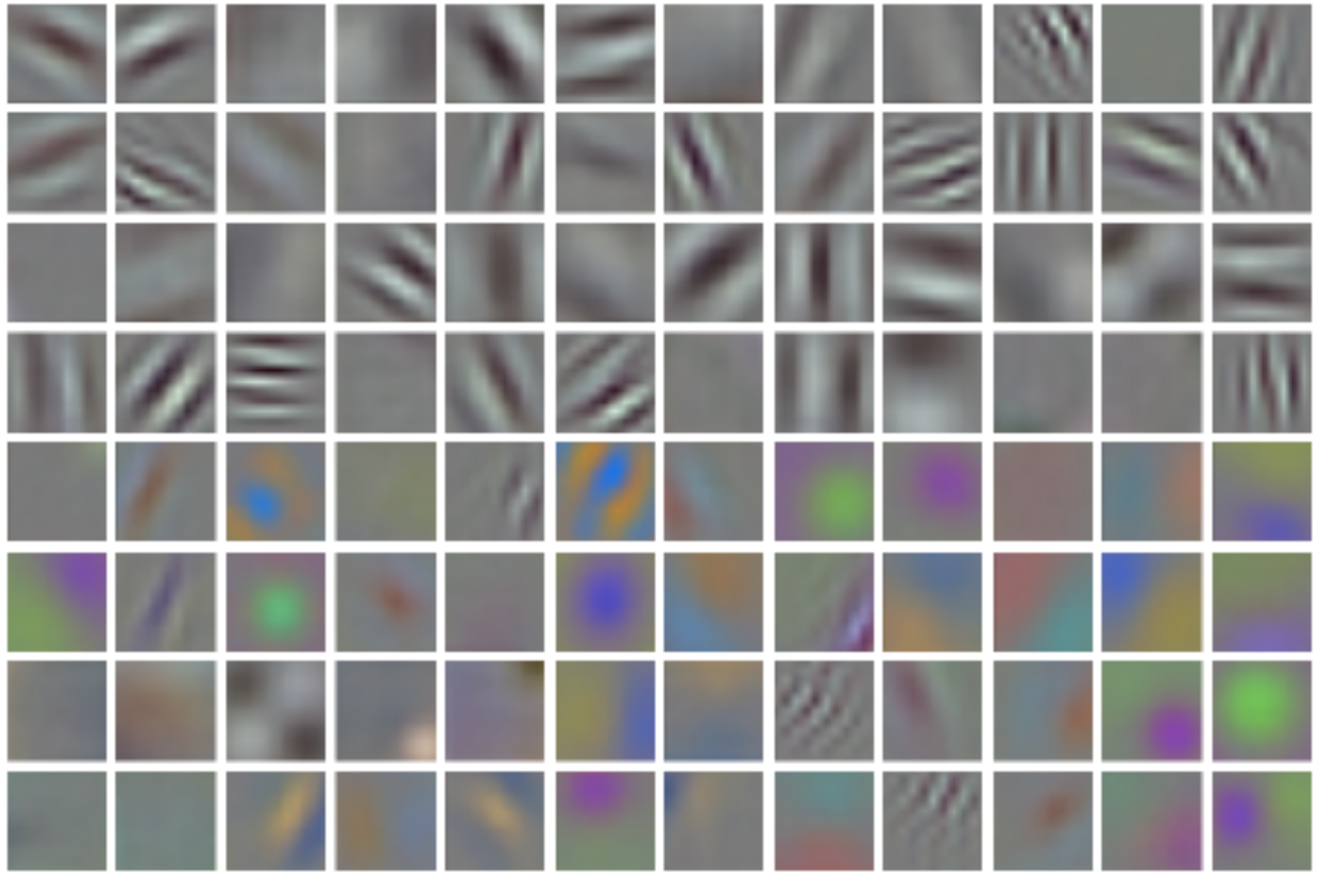

5 Convolutional Layers
1000-way softmax
3 Fully Connected Layers
Stride of 4
Max pooling
Max pooling
Max pooling
dense
dense
dense
224
55
27
13
48
128
192
2048
1000

AlexNet
1st Convolution Layer

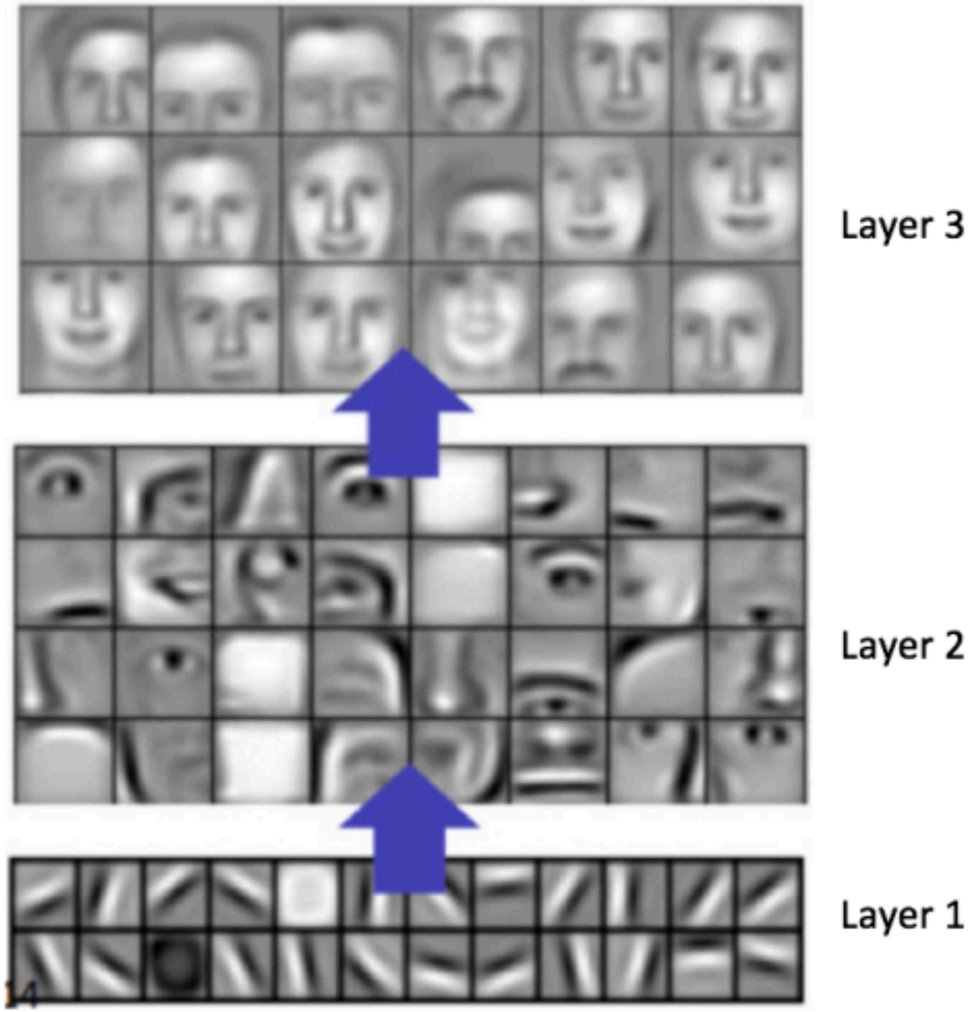


- Images: 227x227x3
- F (receptive field size): 11
- S (stride) = 4
- Conv layer output: 55x55x96

Alexnet 1st Conv Filters

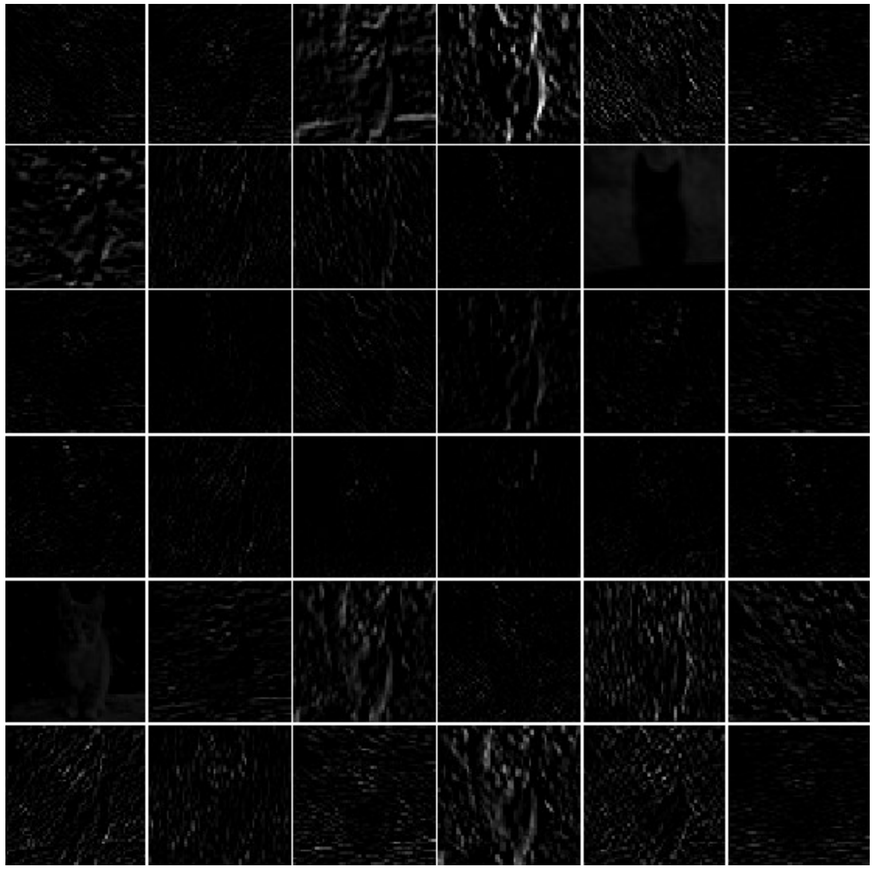


Alexnet

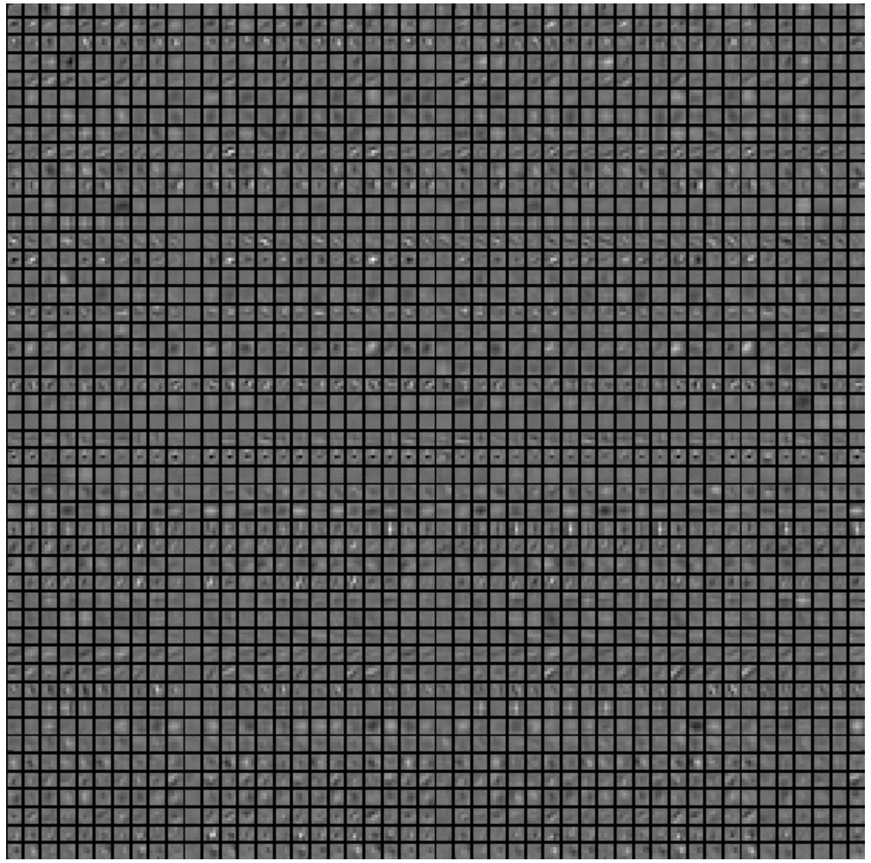


Alexnet

Feature Map Conv1



Conv2



Training a CNN

- Back-propagation + stochastic gradient descent with momentum
- Dropout
- Data Augmentation
- Batch Normalization
- Initialization
  - Transfer Learning
- The make use of **GPU's** in order to optimize the weights.

Reducing Overfitting

Data Augmentation

60 million parameters, 650,000 neurons
-> overfits a lot

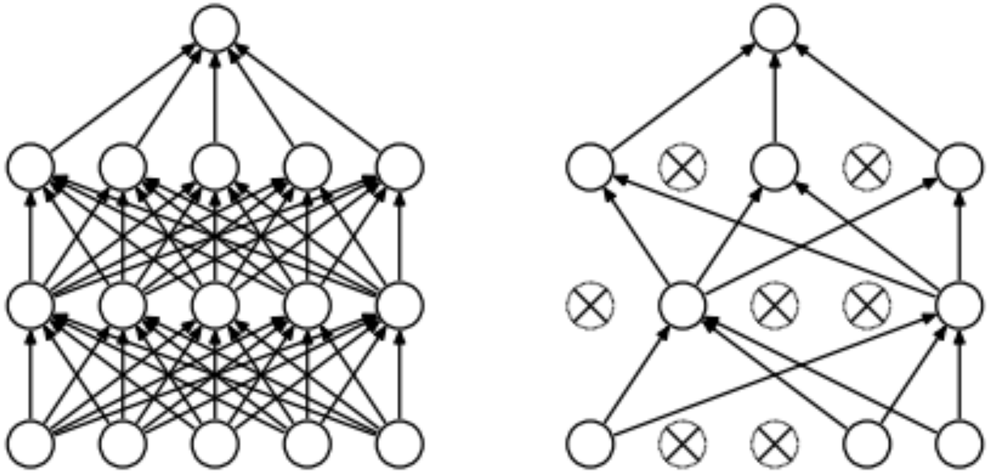
Crop 224x224 patches (and their horizontal reflections)

Data Augmentation at test
average the predictions on the 10 patches.

Reducing Overfitting

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Journal of Machine Learning Research 15 (2014) 1929-1958



(a) Standard Neural Net

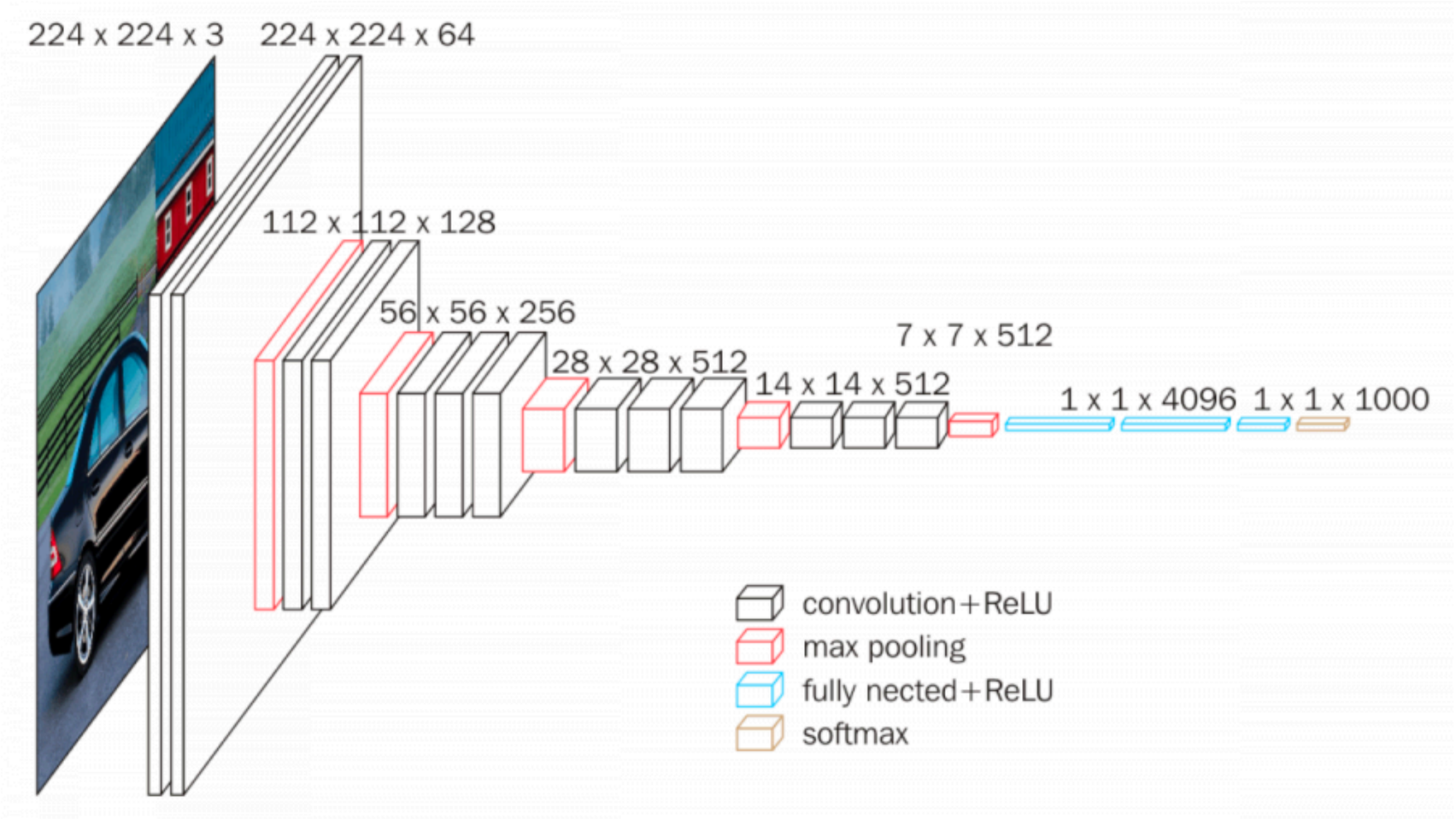
(b) After applying dropout.

Exercice

- How many parameter do the model is a RGB image of 50x50:
  - 1 convolution layer with10 filters of size 5x5
  - Max-pooling 2x2 with stride 2
  - 1 convolution layer with 20 filters of size 3x3
  - Dense Layer with 100 neurons
  - Dense Layer with 10 neurons

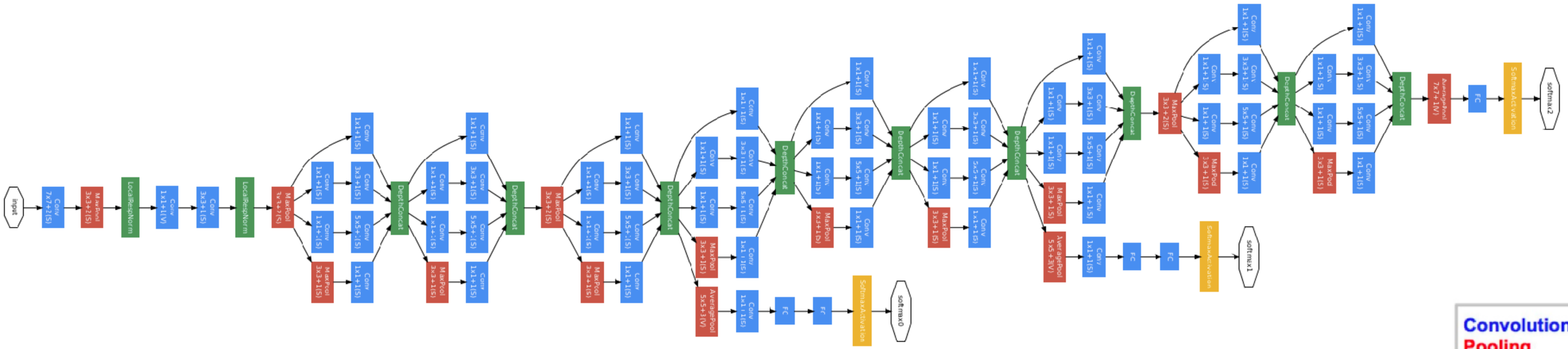

WE NEED TO GO
DEEPER
troll-face.ru

VGG Net


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

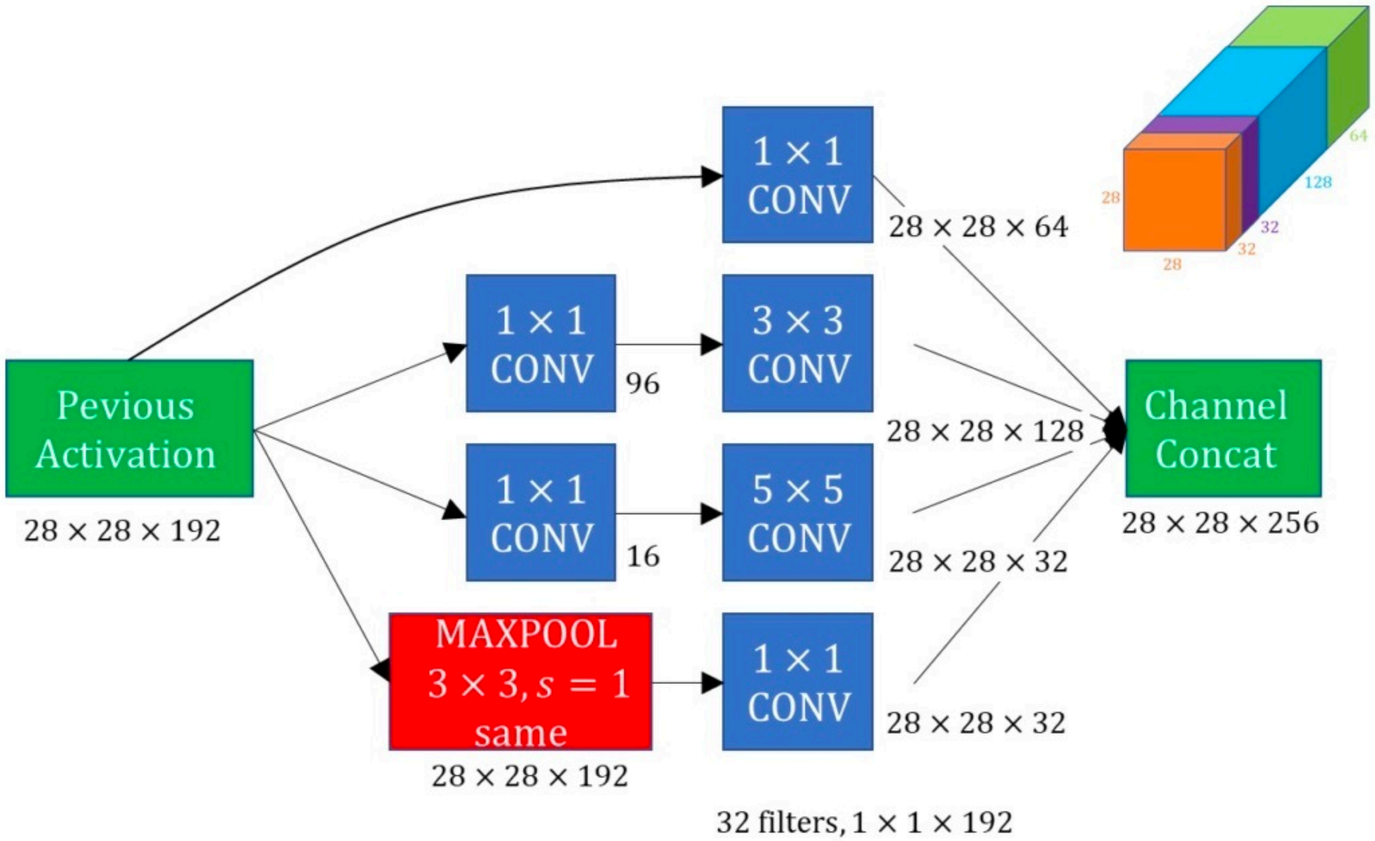
GoogleNet

22 layers, but **12 times less parameters** than AlexNet



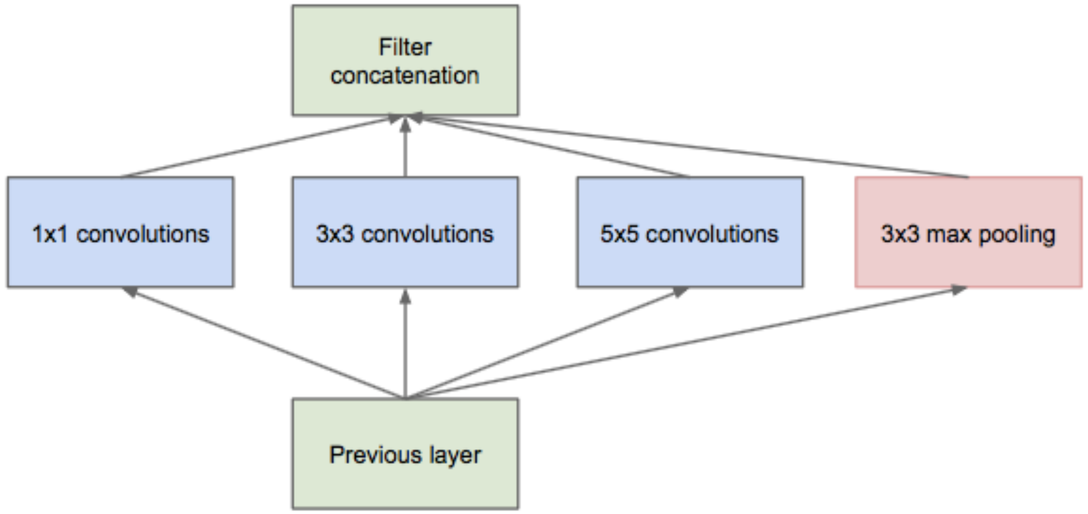
GoogleNet

An example of an Inception module



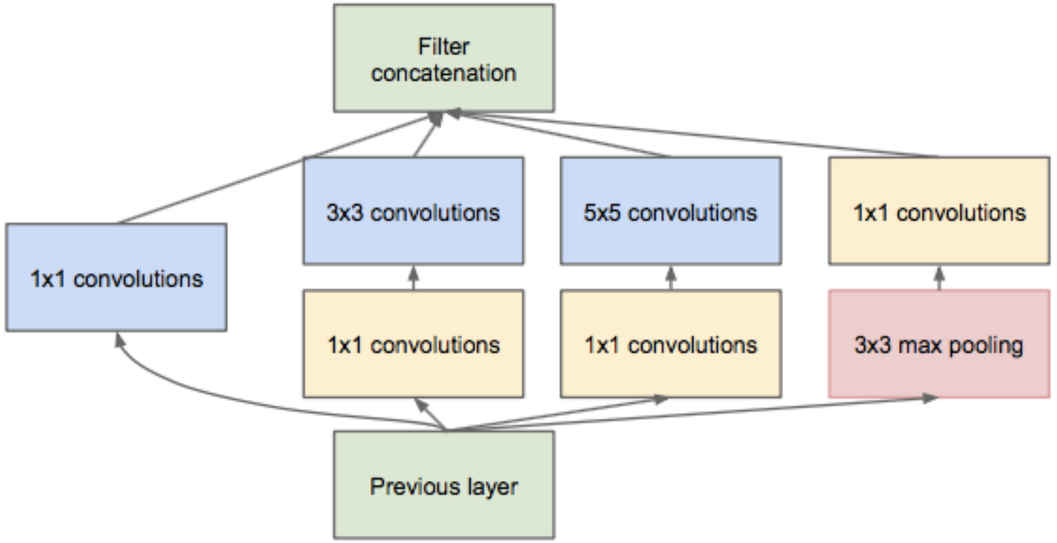
Inception Module:
Naive Version

Convolutional filters with different sizes can cover different clusters of information.
By finding the optimal local construction and repeating it spatially, they approximate the optimal sparse structure with dense components.
For convenience of computation, they use 1 x 1, 3 x 3 and 5 x 5 filters + pooling.
Together these made up the naive Inception module.



Inception Module:
Naive Version

Stacking these inception modules on top of each would lead to an exploding number of outputs
Solution: inspired by "Network in Network" add 1x1 convolutions for dimensionality reduction

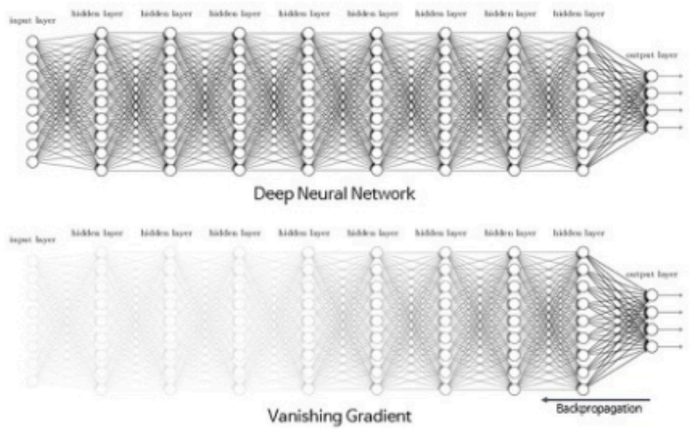


Conclusion:

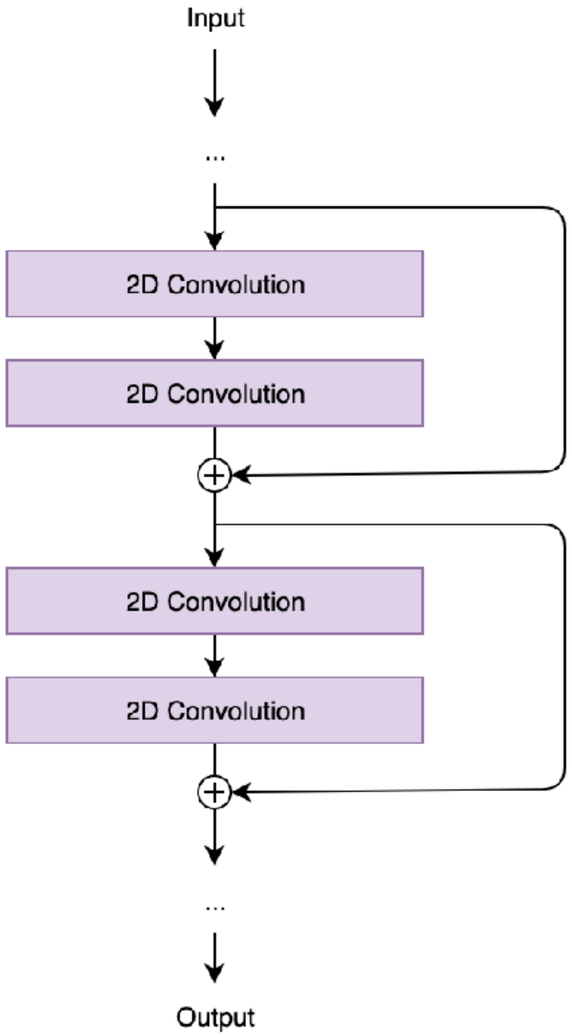
More layers is better


WE NEED TO GO
DEEPER
troll-face.ru

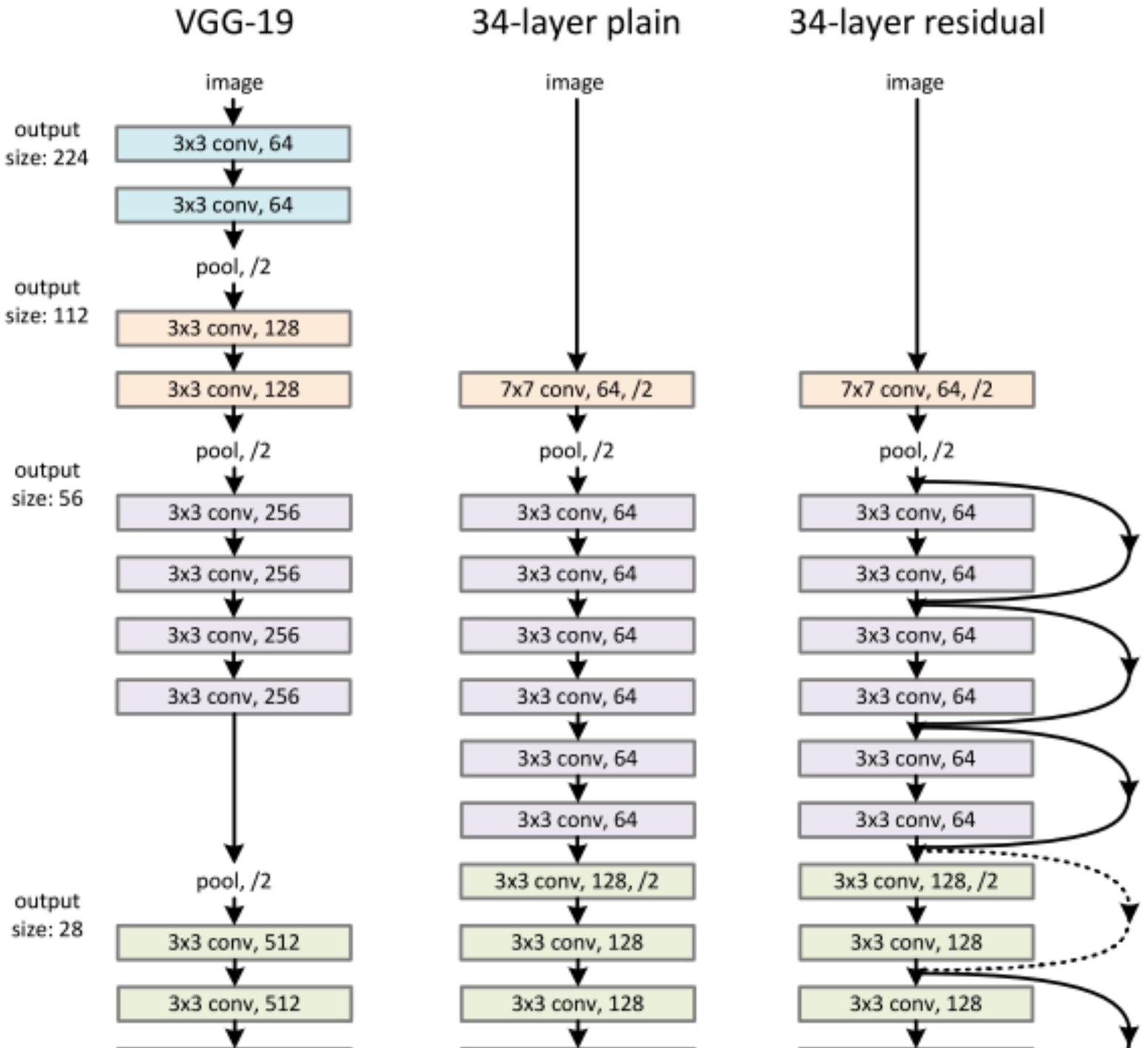
Not everything is that simple:
Vanishing Gradient Problem



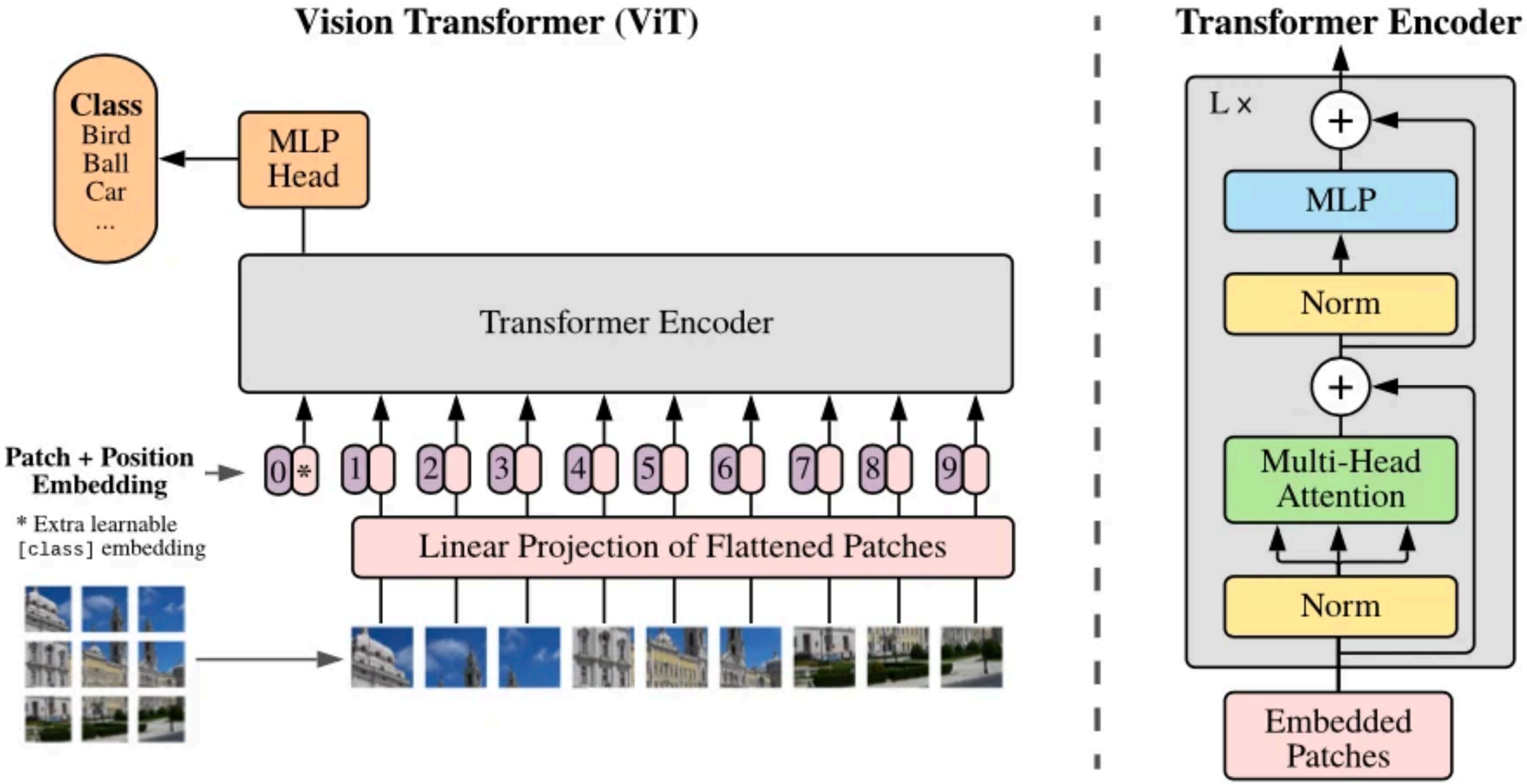
As the gradient keeps flowing backward to the initial layers, this value keeps getting multiplied by each local gradient. Hence, the gradient becomes smaller and smaller, making the updates to the initial layers very small, increasing the training time considerably.



These ***skip connections*** act as gradient *superhighways*, allowing the gradient to flow unhindered.


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128

Visual TRANSFORMERS



Learning algorithm

- Learning the parameters is a complex and time-consuming task.
  - **Huge amount** of **data** is needed.
  - **Back progragation** method is used.
  - Great **packages** like Caffe, Theano or TensorFlow.
  - The make use of **GPU's** in order to optimize the weights.

Unsupervised learning

Layout

- Autoencoders
- Learning unsupervised representations
- Sparse coding
- A manifold learning view
- Deep patient
- Self-Supervised Learning

Unsupervised learning tries to understand the properties of a particular set of data. There are different ways of doing this

- Clustering - Divide data in groups according to some notion of similarity.

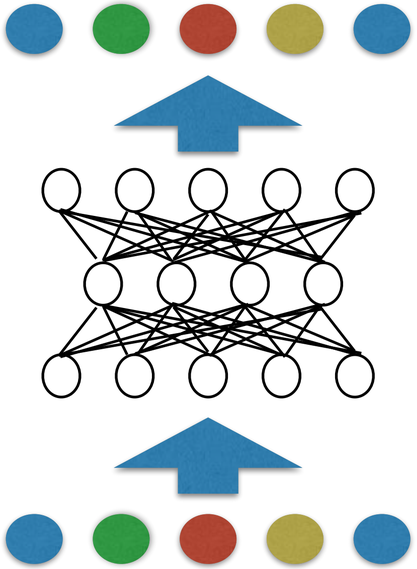
- Manifold learning - Understanding how data is distributed in the space, parameterising a manifold.

Autoencoder

- Build a network with the aim of reconstruction.

D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In Parallel Distributed Processing. Vol 1: Foundations. MIT Press, Cambridge, MA, 1986.

Autoencoders



Reconstruction $\tilde{x}$

Input x

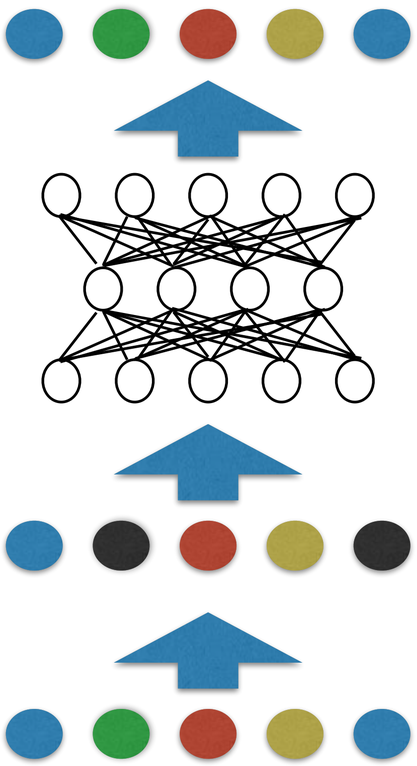
Loss

$$\mathcal{L}(\tilde{x}, x) = \sum(\tilde{x} - x)^2$$

Problems

- In large networks it may learn the identity mapping rendering the auto encoder representation useless.
- In order to correct this issue and furthermore give robustness to the auto encoder, denonising auto encoders are proposed.

Denoising Autoencoders



Reconstruction

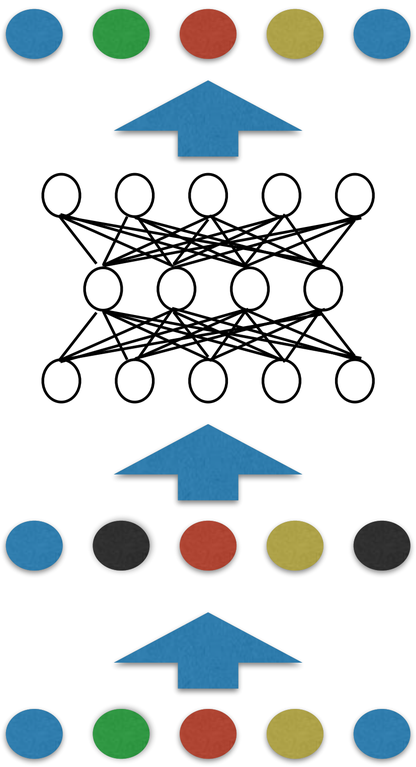
Noise: dropout input values

Input

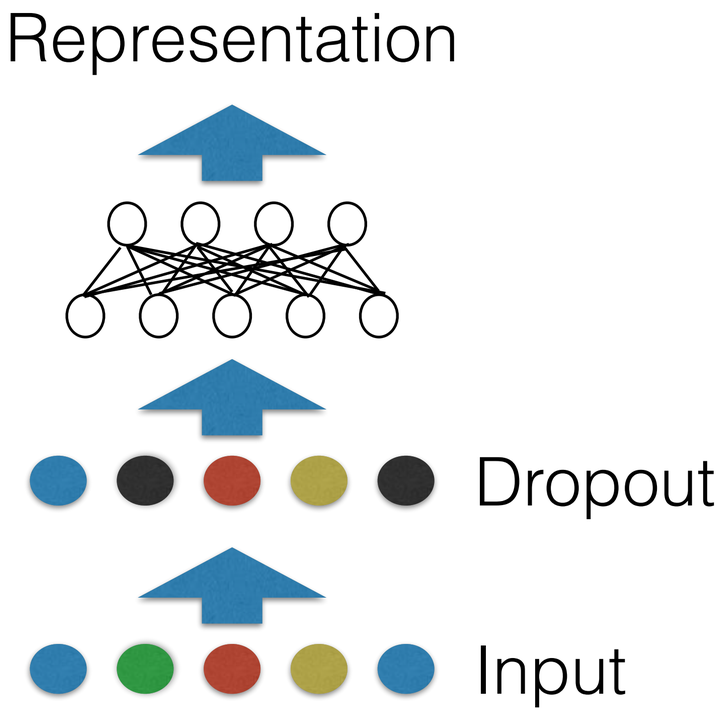
Learning representations

- What can we use these representations for?
    - Transfer learning
        - Pure transfer
        - Pretraining
    - Compression

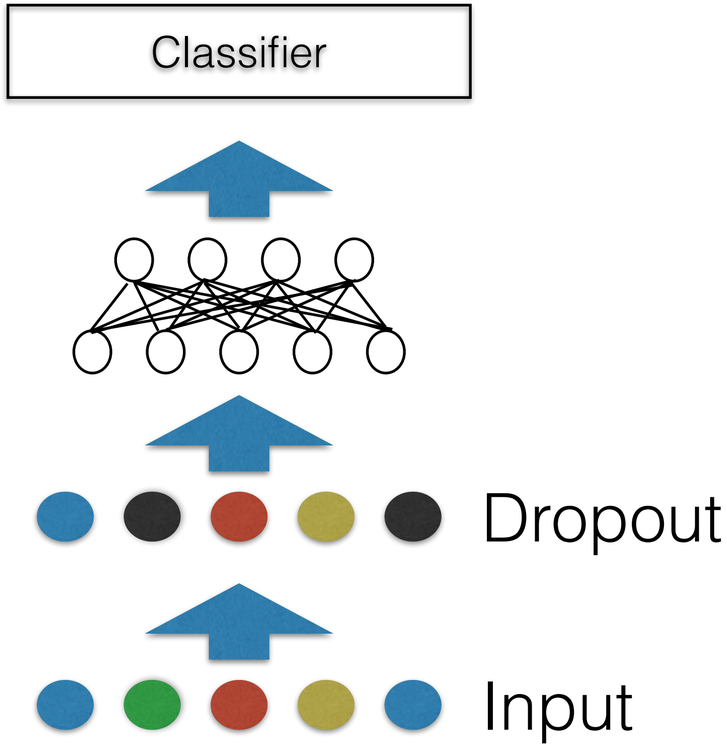
Pretraining and transfer



Pretraining and transfer

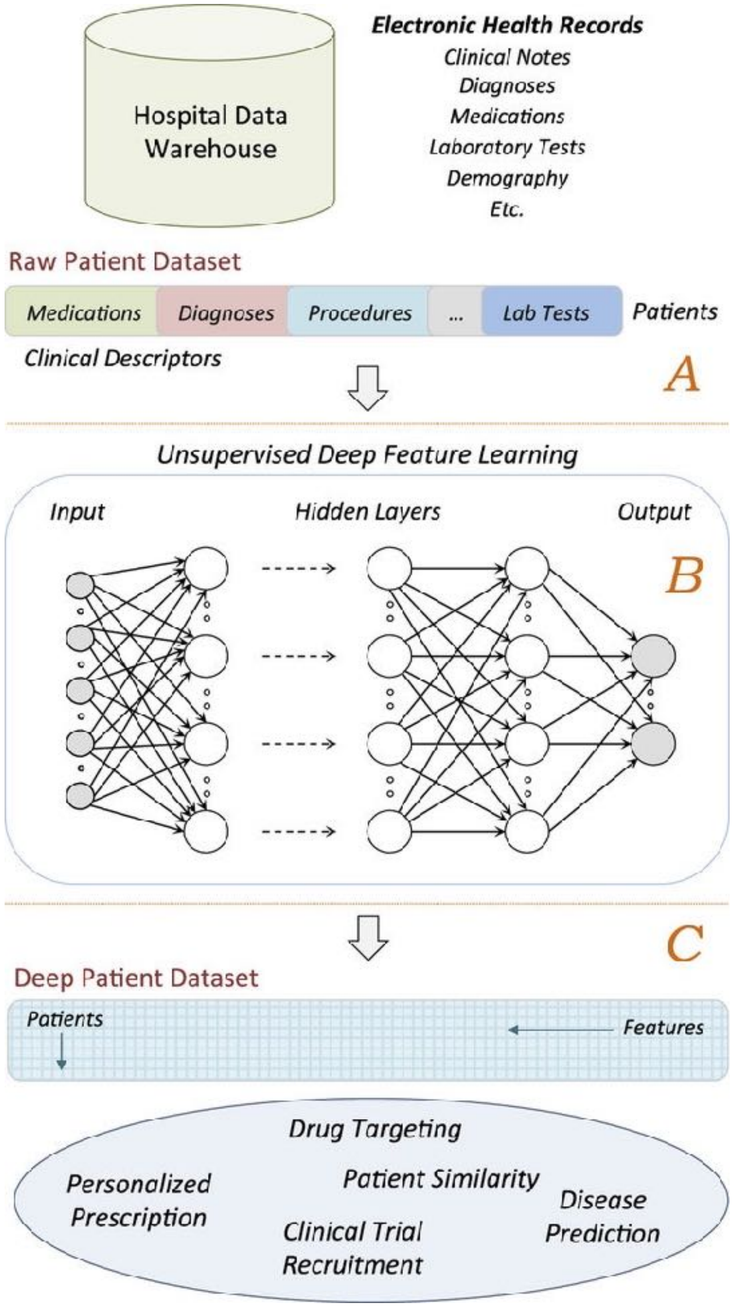


Pretraining and transfer

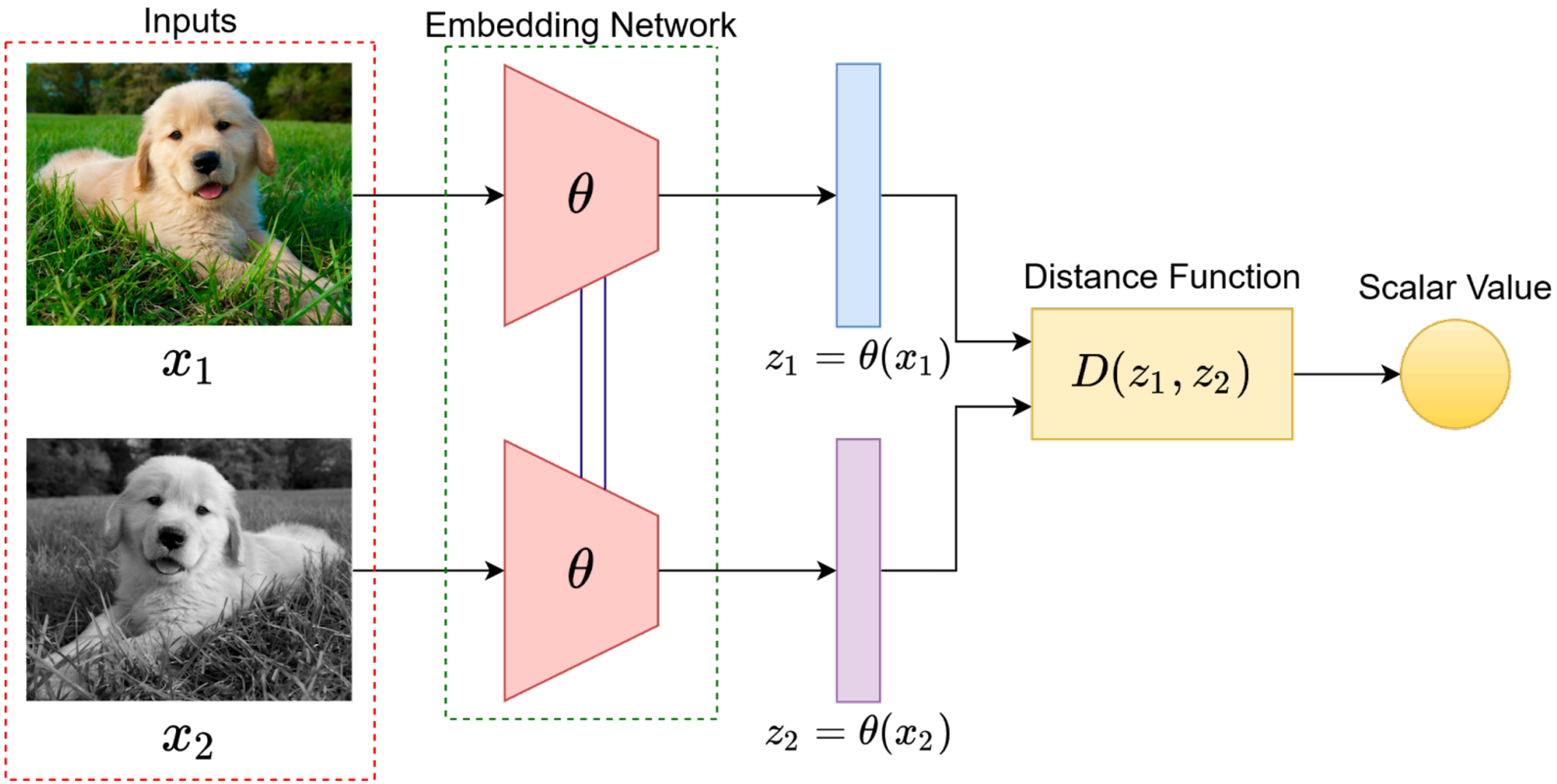


Application

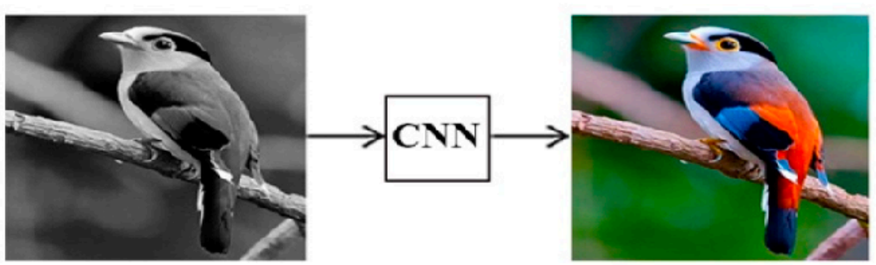
- Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records


Hospital Data Warehouse
Electronic Health Records
Clinical Notes
Diagnoses
Medications
Laboratory Tests
Demography
Etc.
Raw Patient Dataset
Medications
Diagnoses
Procedures
...
Lab Tests
Patients
Clinical Descriptors
A
Unsupervised Deep Feature Learning
Input
Hidden Layers
Output
B
C
Deep Patient Dataset
Patients
Features
Drug Targeting
Personalized Prescription
Patient Similarity
Clinical Trial Recruitment
Disease Prediction

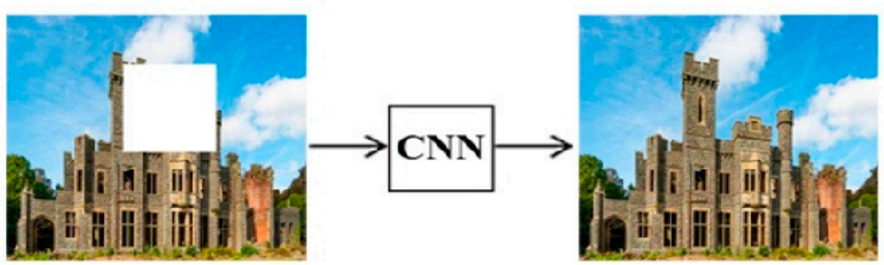
Self-Supservised Learning



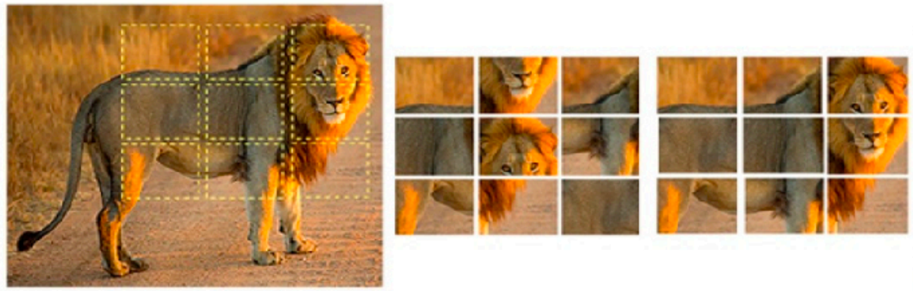
Self-Supservised Learning



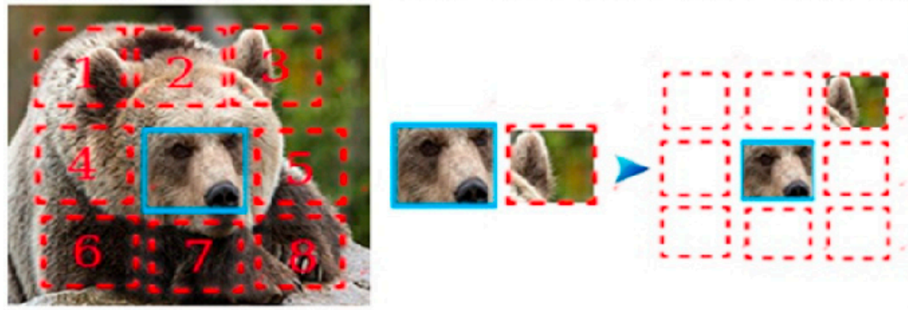
a) Colorizing an image



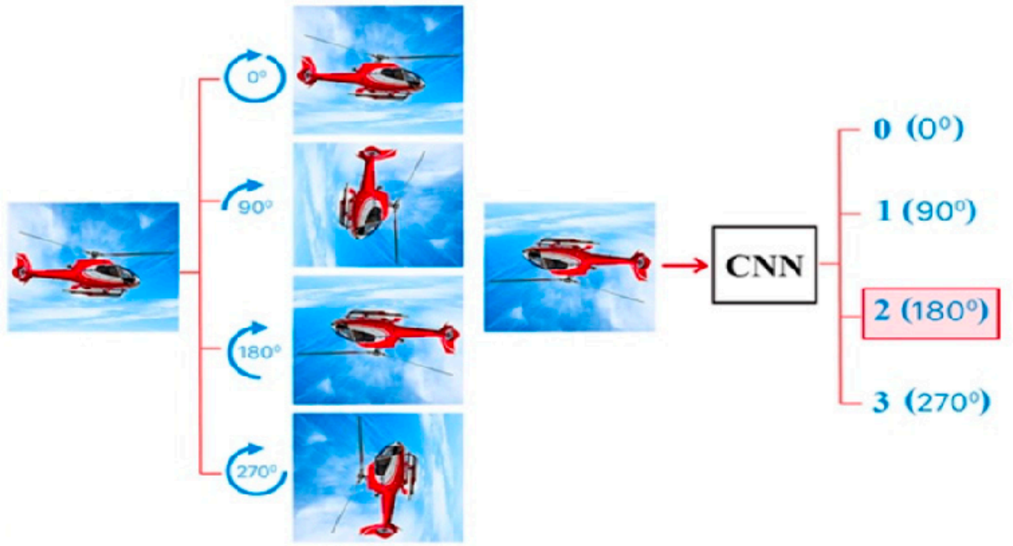
b) Inpainting



c) Solving Jigsaw Puzzle



d) Predicting relative position



e) Estimating the rotation angle