Deep Learning From Scratch

# Neural Networks: Part 2

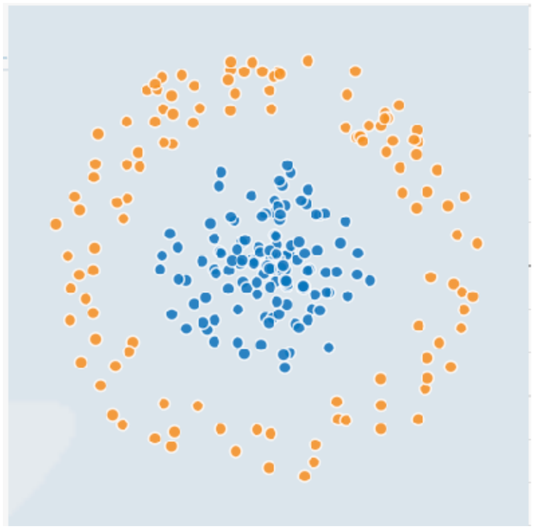
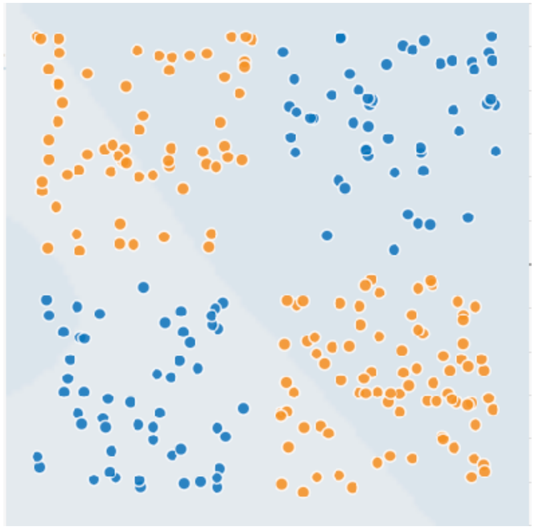
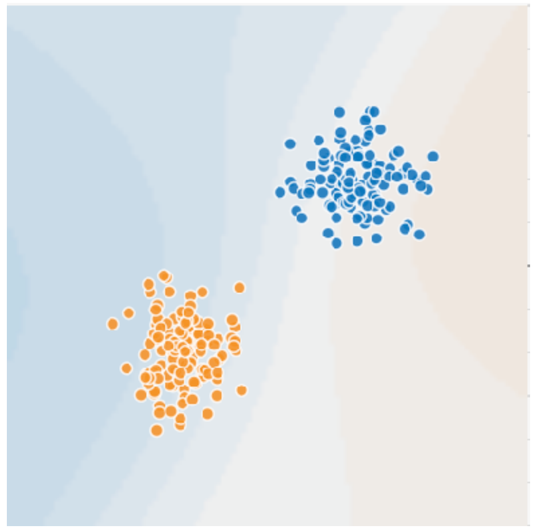


Santi Seguí

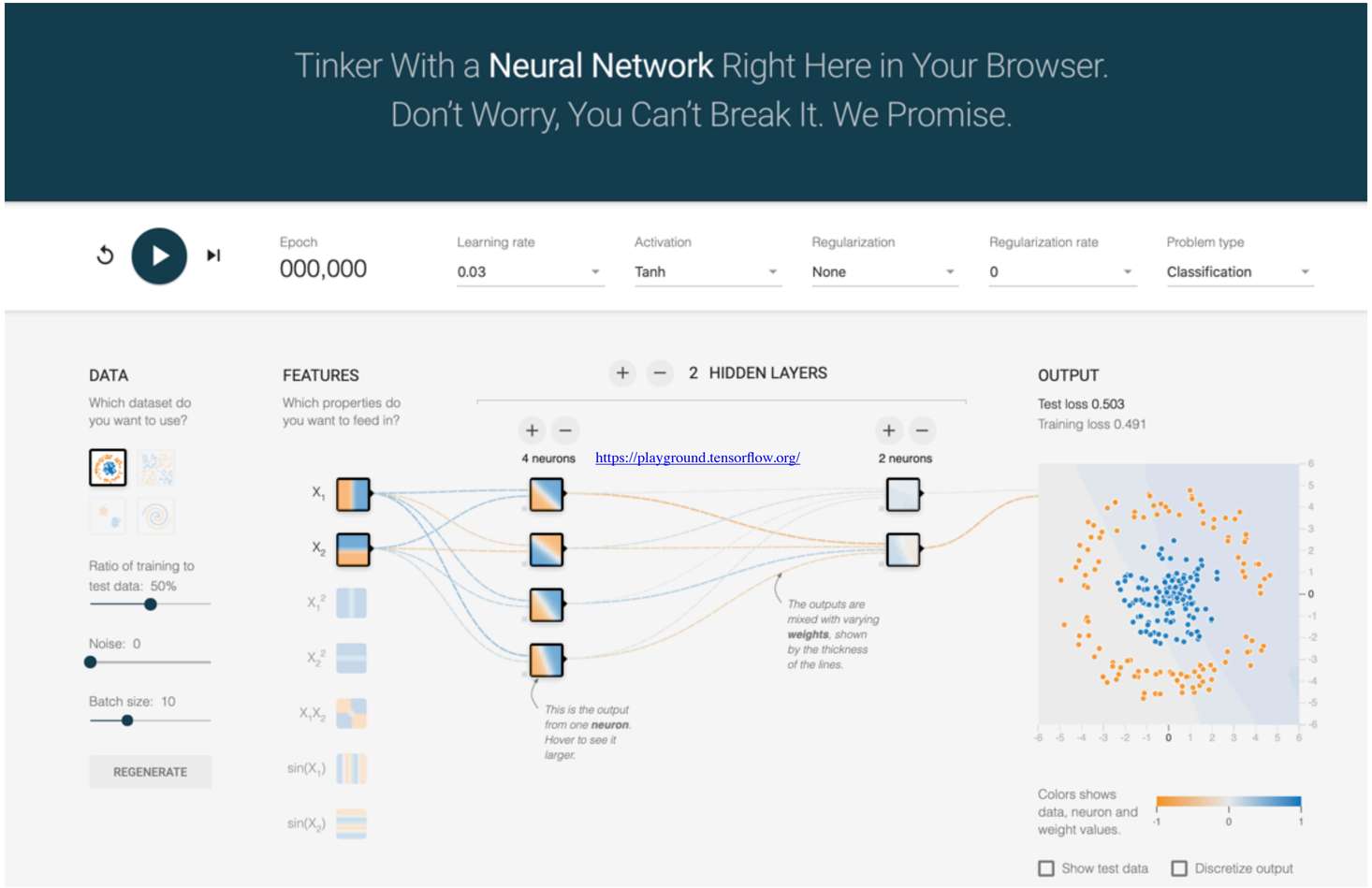
Associate Professor at the Department of Mathematics and Computer Science
from the University of Barcelona.
PhD in 2011 on Computer Vision and Machine Learning
Graduate on Computer Science in 2007

**Santi Seguí** **Email:** santi.segui@ub.edu

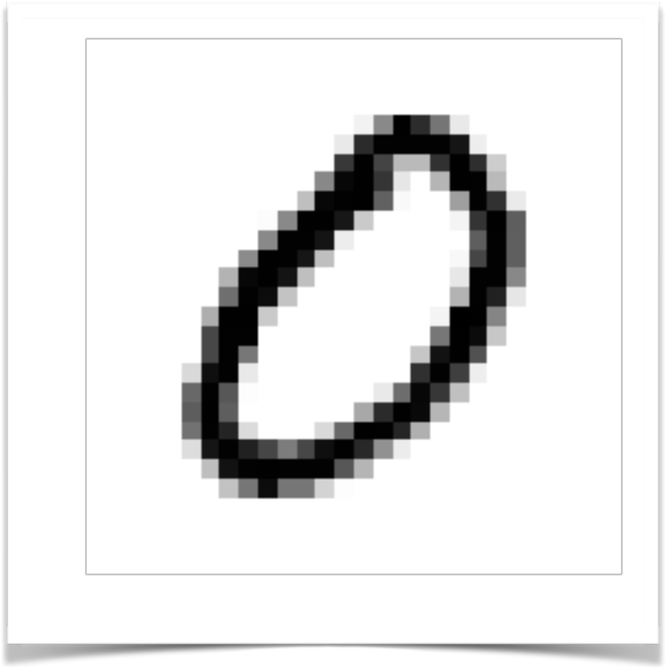
# wrap up!


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


Tinker With a Neural Network Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
https://playground.tensorflow.org/
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.503
Training loss 0.491
Colors shows data, neuron and weight values.
Show test data
Discretize output

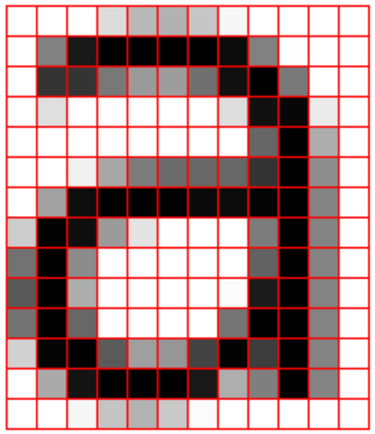
# wrap up!



How can we create a Neural Network able to detect the image digit?

# An image

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

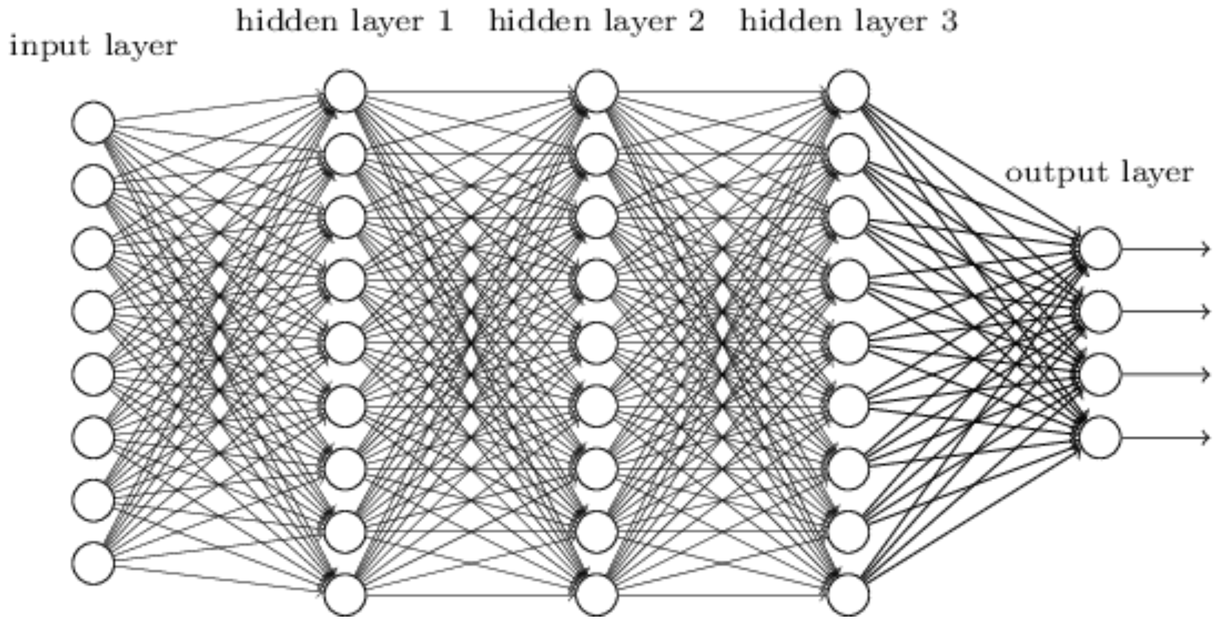


| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.0 | 1.0 | 1.0 | 0.9 | 0.6 | 0.6 | 0.6 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| 1.0 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.5 | 1.0 | 1.0 | 1.0 |
| 1.0 | 0.2 | 0.2 | 0.5 | 0.6 | 0.6 | 0.5 | 0.0 | 0.0 | 0.5 | 1.0 | 1.0 |
| 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 |

# Neural Networks for Images

# Neural Networks for Images

Multi Layer Perceptron




How many parameter does this MLP has?

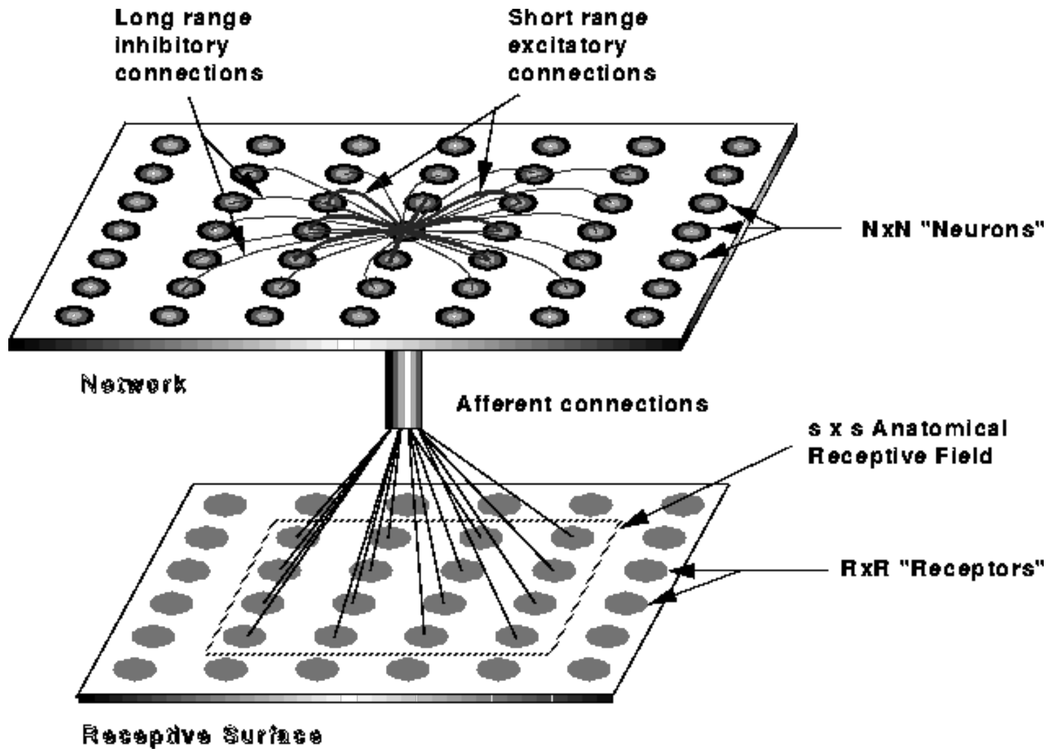
8 * 9 + 9 * 9 + 9 * 9  +9 *4 = 270

# 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 milion** 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)

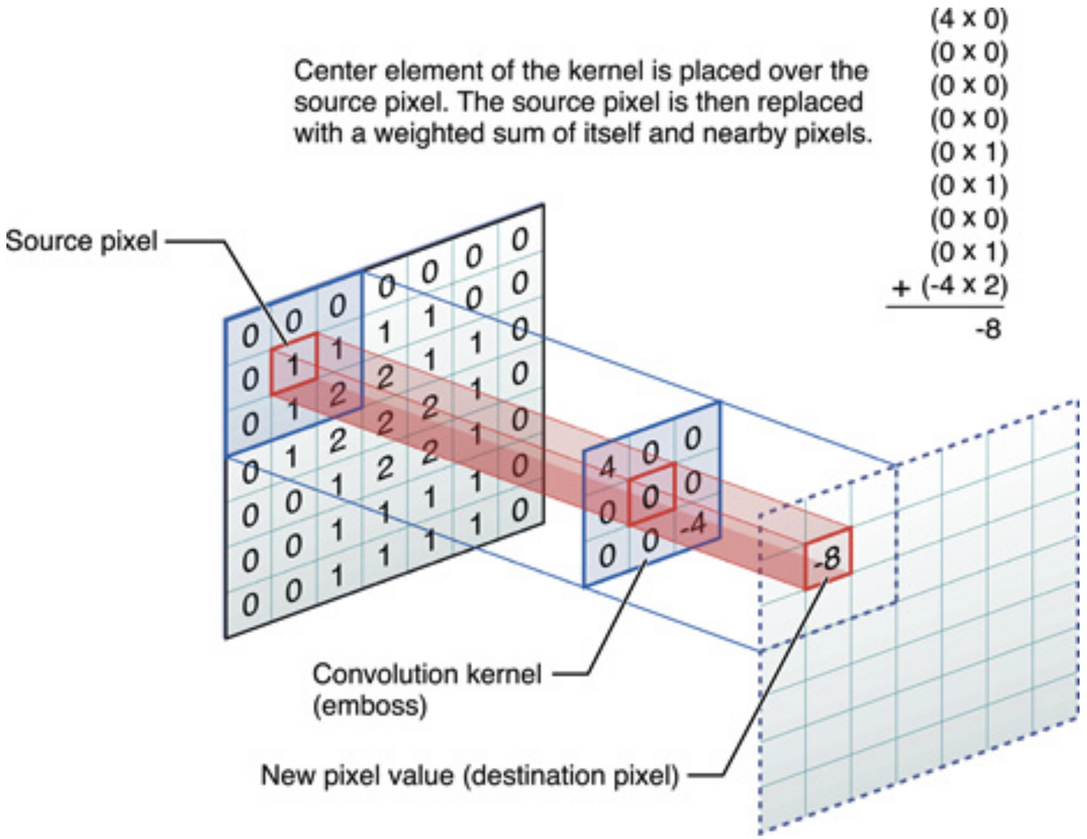
- Three main ideas:

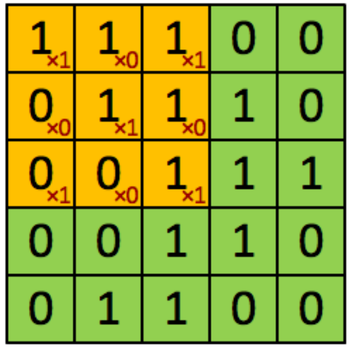
  1. **local receptive fields**,
  2. **shared weights**,
  3. **sub-sampling**

# Convolutional Neural Networks (CNNs)

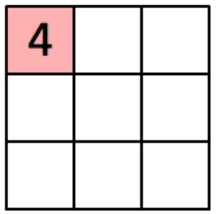
- Repetitive blocks of neurons that are applied across space (for images) or time (for audio signals etc).
- For images, these blocks of neurons can be interpreted as 2D convolutional kernels, repeatedly applied over each patch of the image.
- For speech, they can be seen as the 1D convolutional kernels applied across time-windows.
- At training time, the weights for these repeated blocks are 'shared', i.e. the weight gradients learned over various image patches are averaged.

# What is an image convolution?



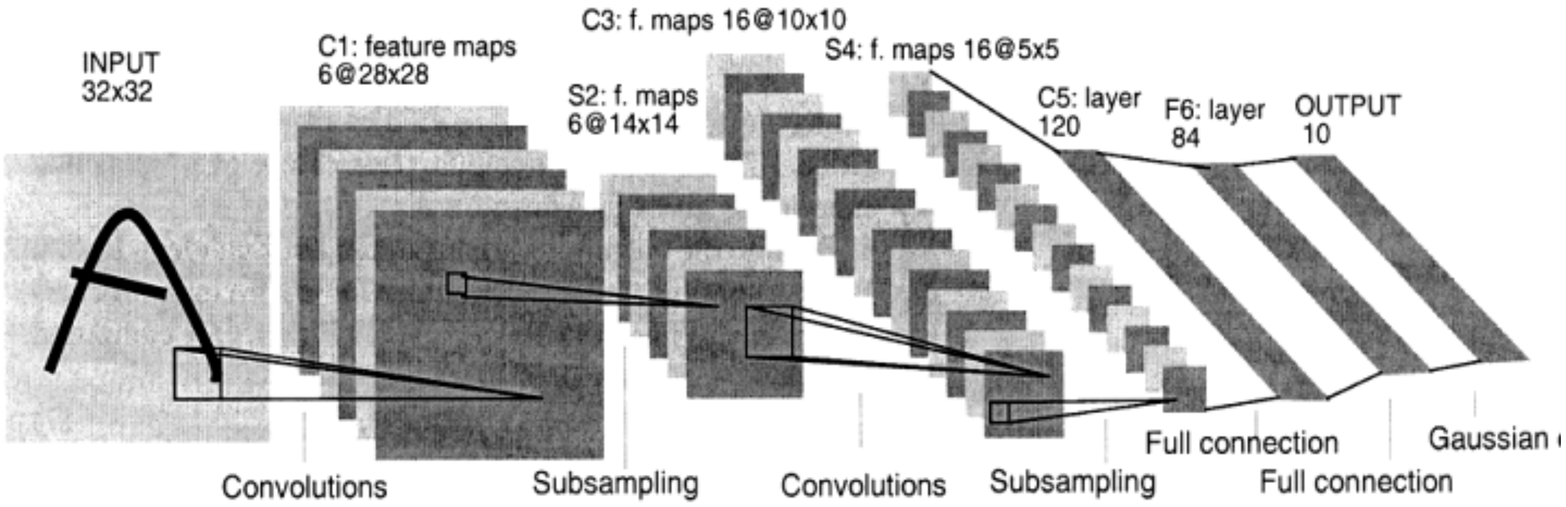

Image




Convolved Feature

# What is an image convolution?

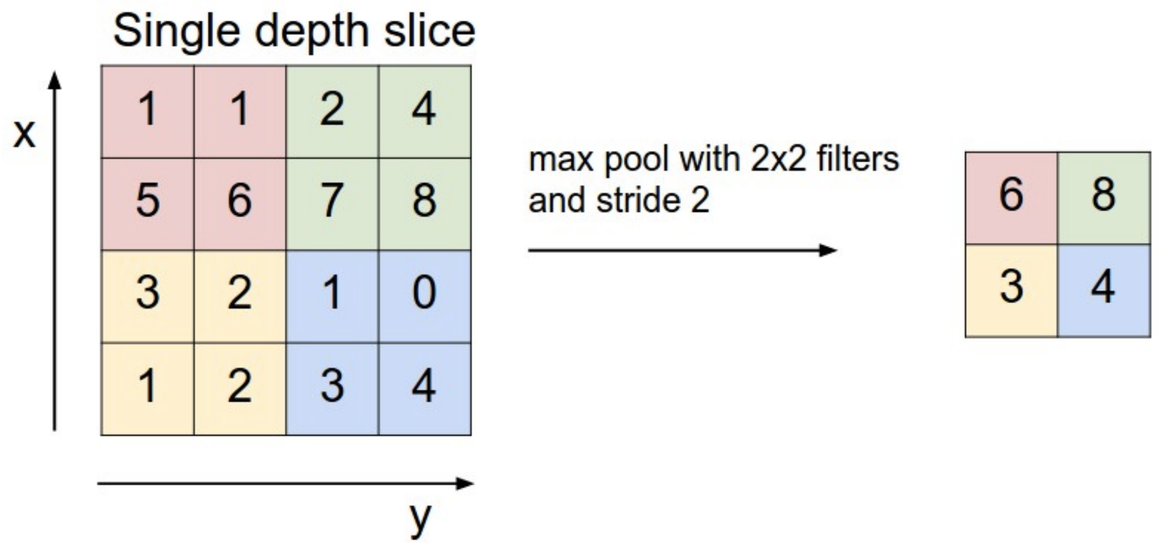
# “Nothing New”




LeCun et al. 1998

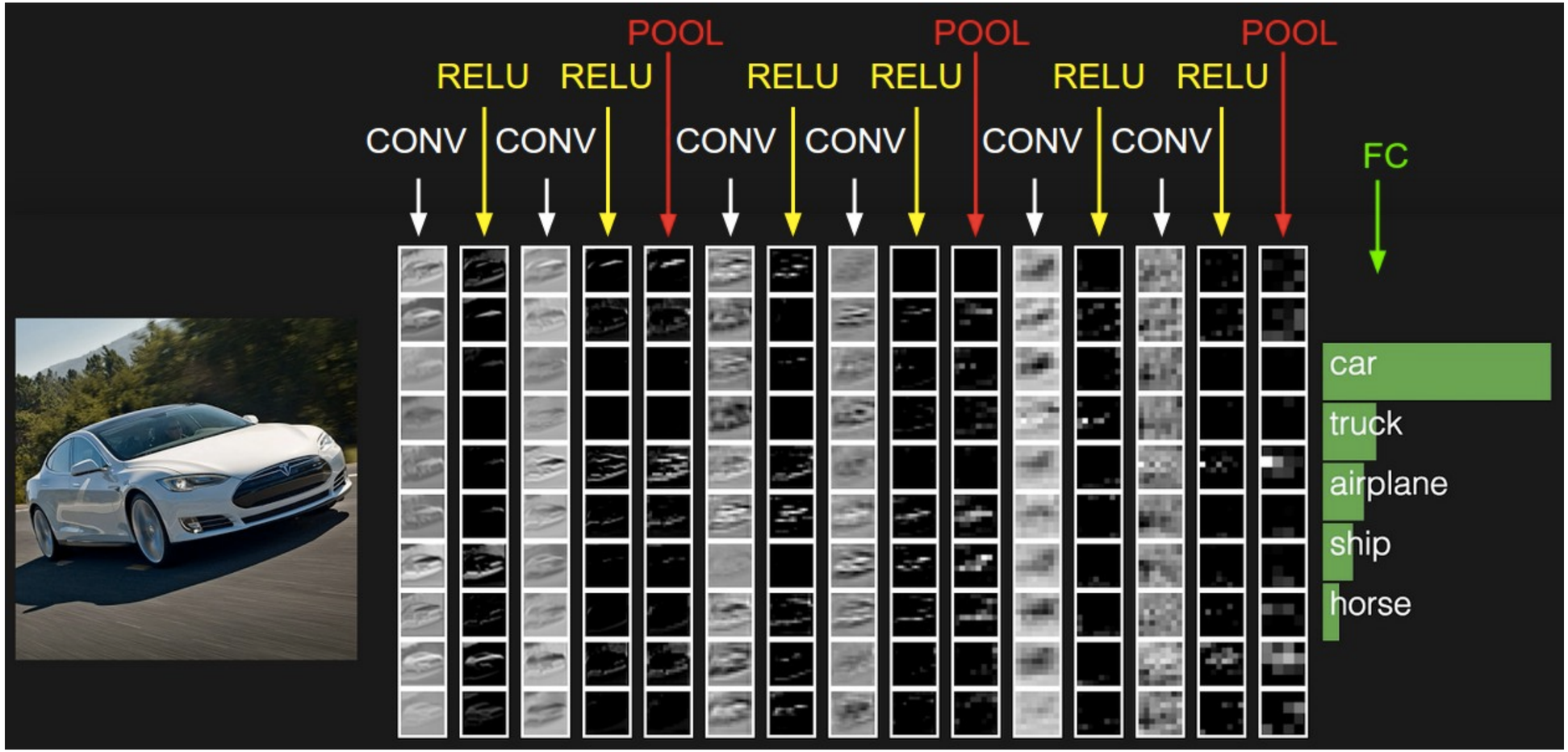
# Max pooling

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



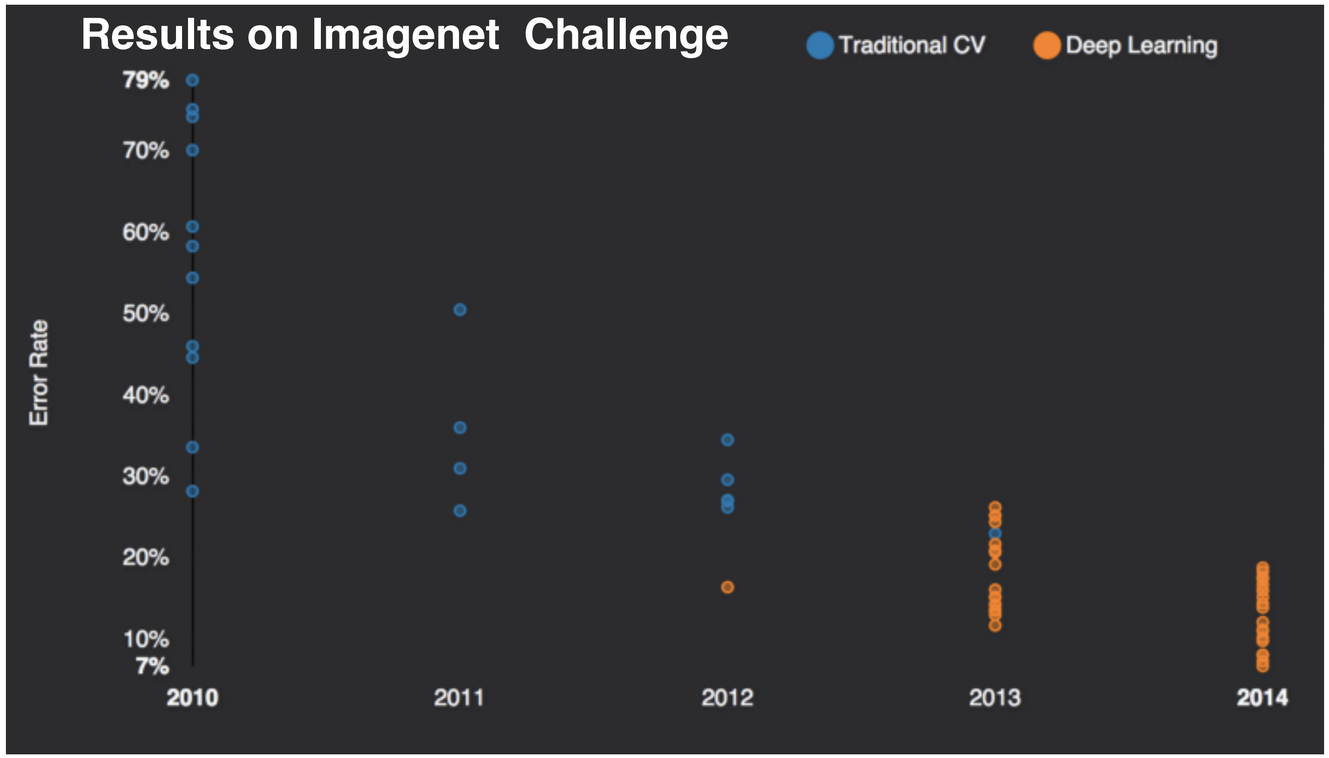

But it is also useful since it provides a form of translation **invariance**

# 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:

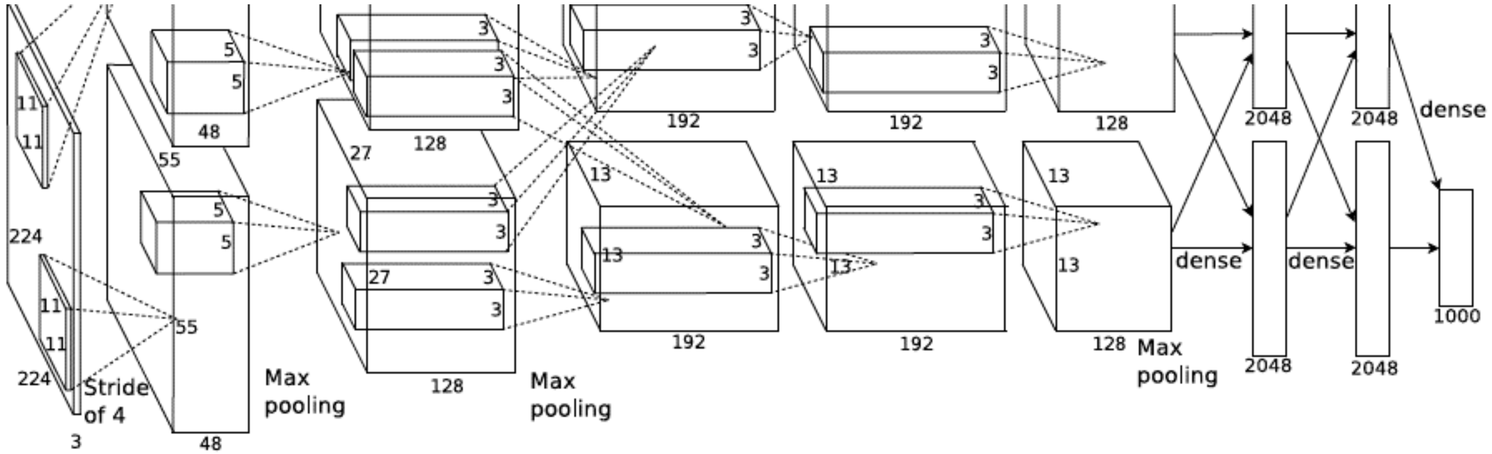
**Bigger model:**

7 hidden layers, 650.000 units, 60 million parameter

**More Data:**

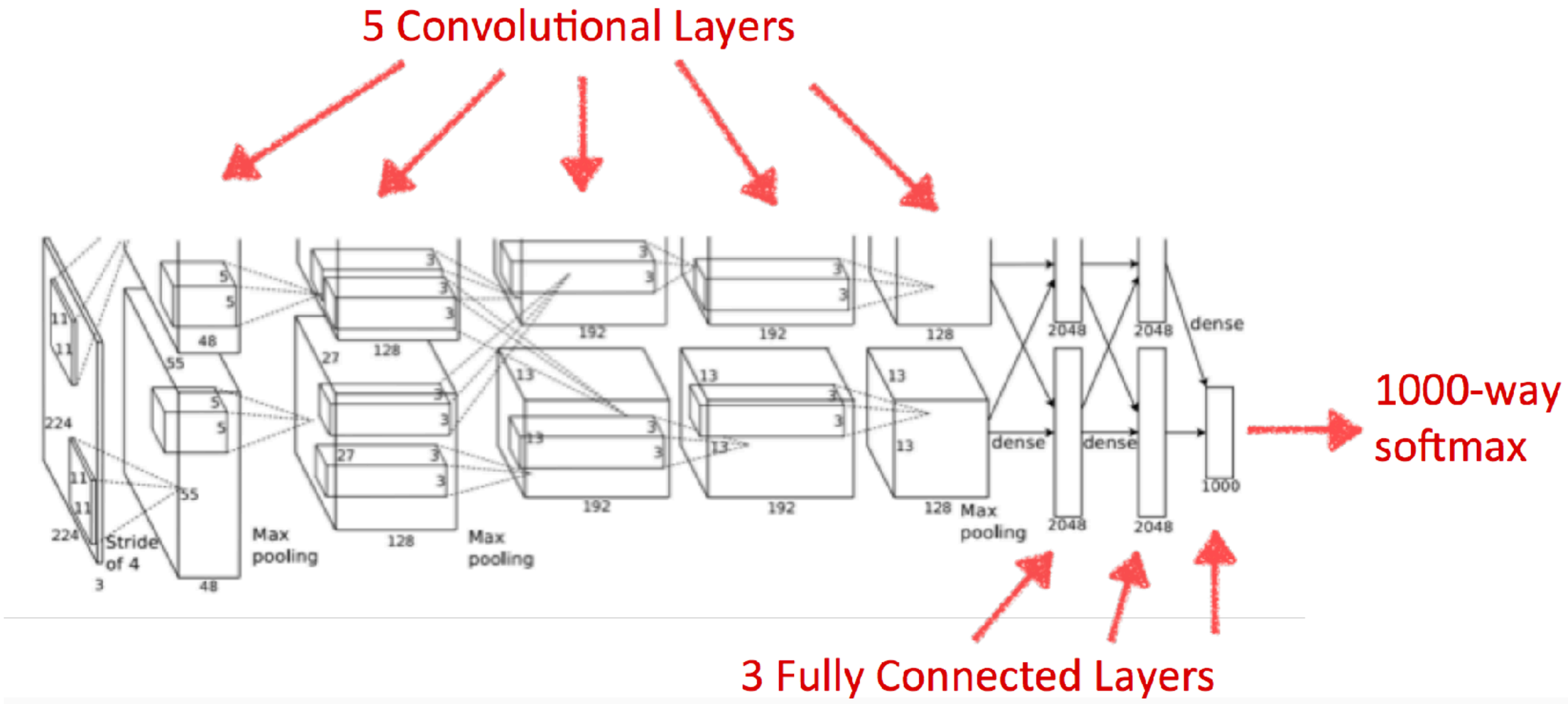
10^6 vs 10^3 images

GPU implementation (50x speedup over CPU)

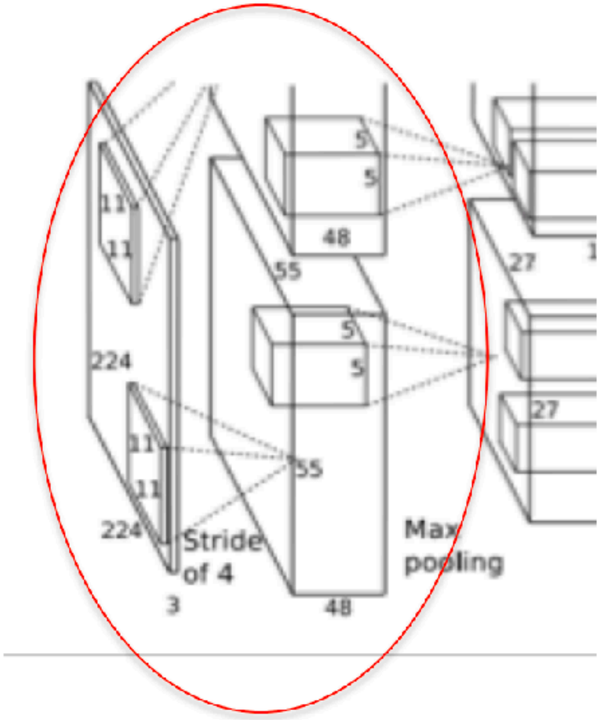



ImageNet Classification with Deep Convolutional Neural Networks. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

# AlexNet

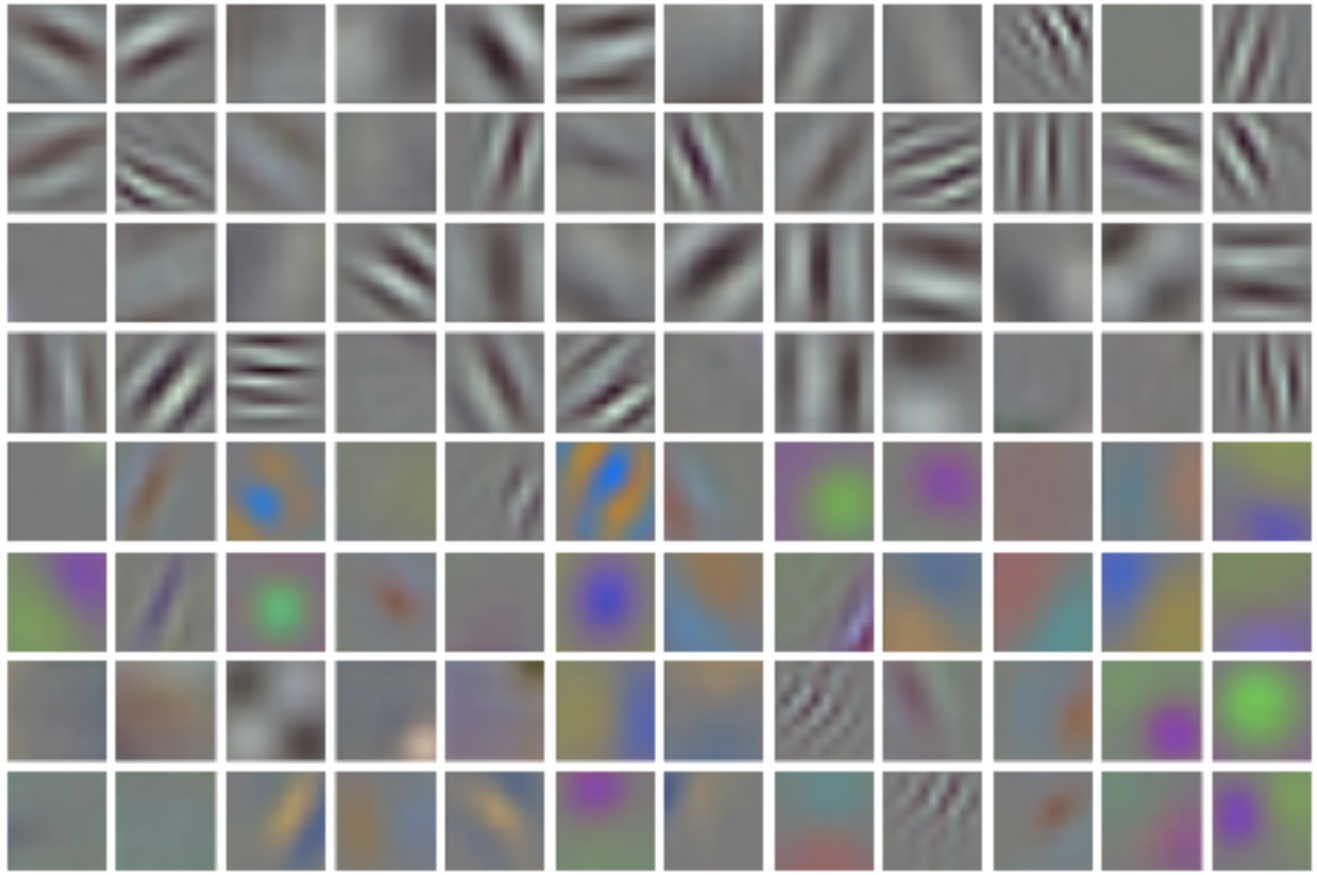

5 Convolutional Layers
224
224
3
11
11
Stride of 4
55
48
Max pooling
27
128
Max pooling
13
192
192
128
Max pooling
2048
2048
dense
dense
dense
1000
1000-way softmax
3 Fully Connected Layers

# AlexNet
# 1st Convolution Layer

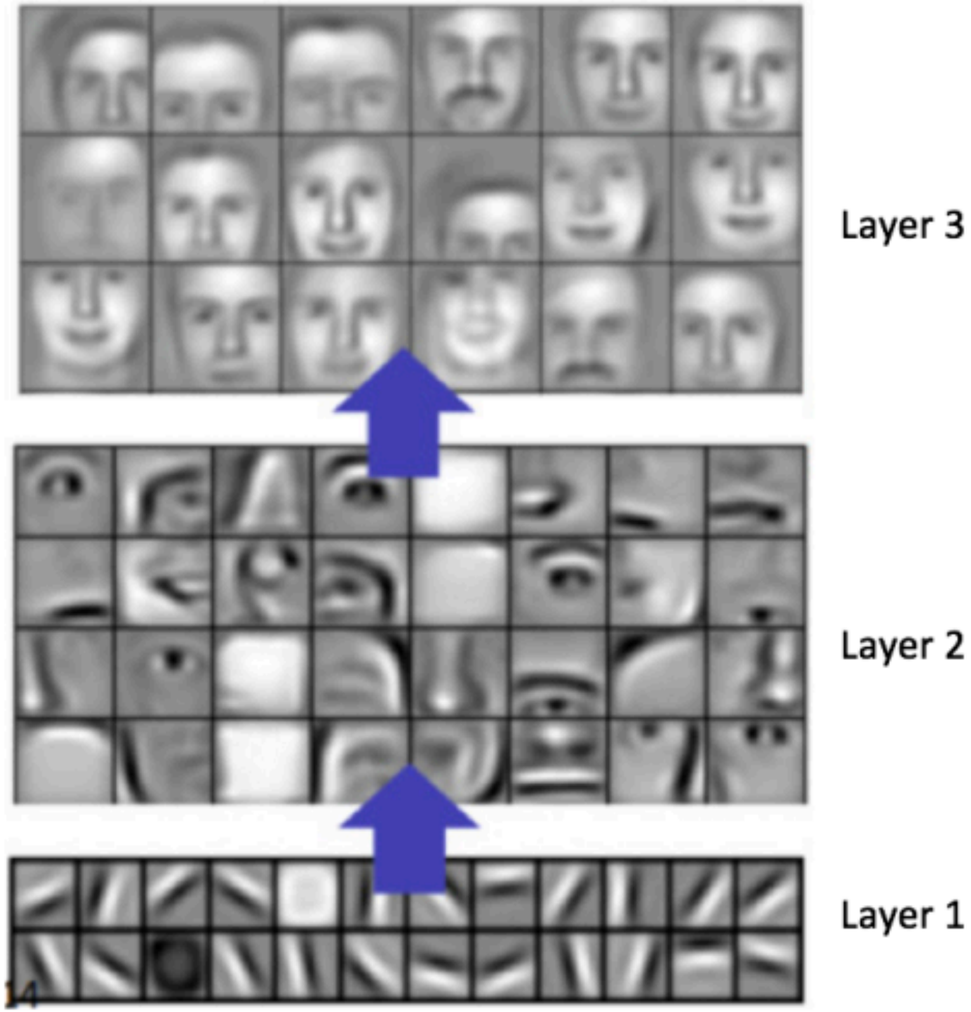



- Images: 227x227x3
- F (receptive field size): 11
- S (stride) = 4
- Conv layer output: 55x55x96

# Alexnet 1st Conv Filters

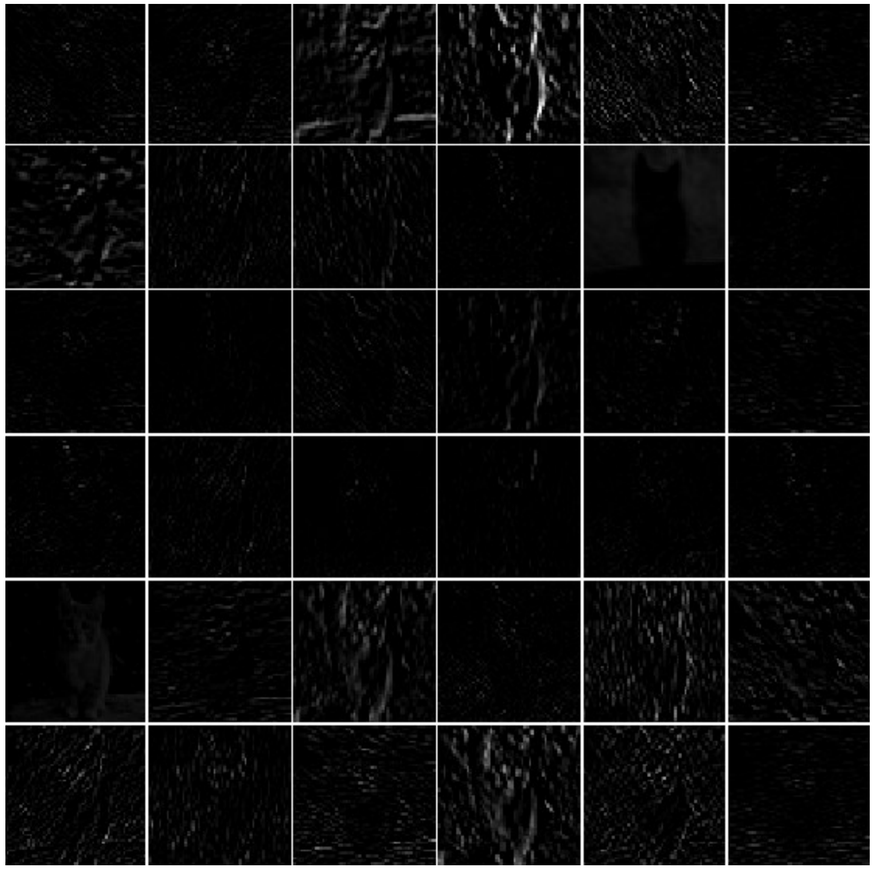


# Alexnet

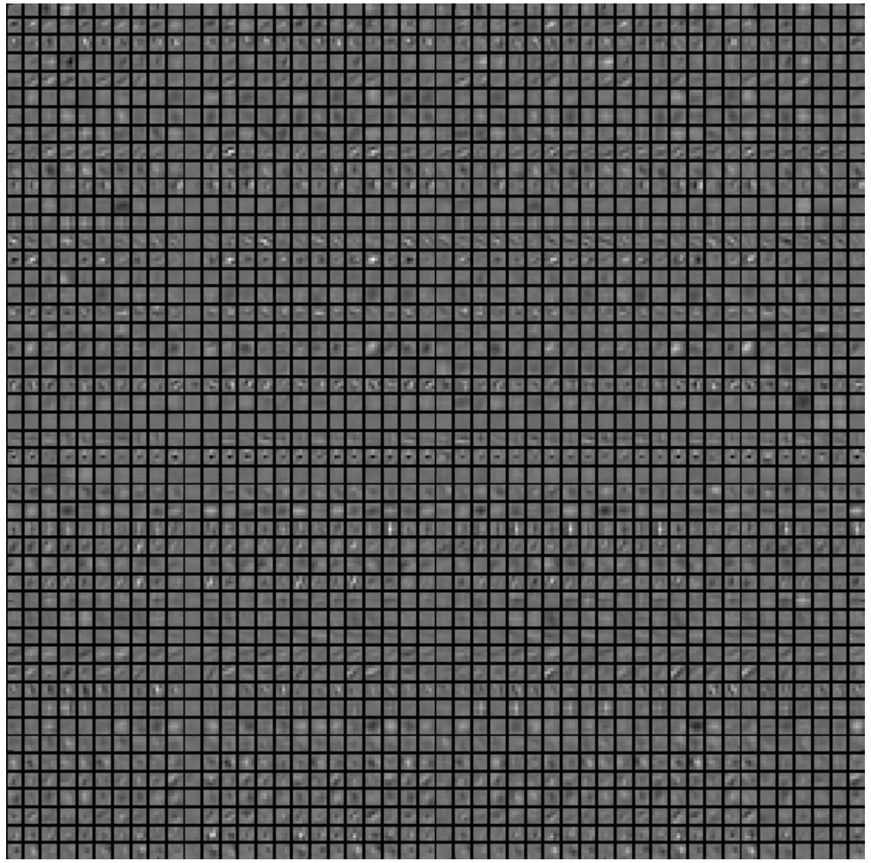


# Alexnet

Feature Map Conv1

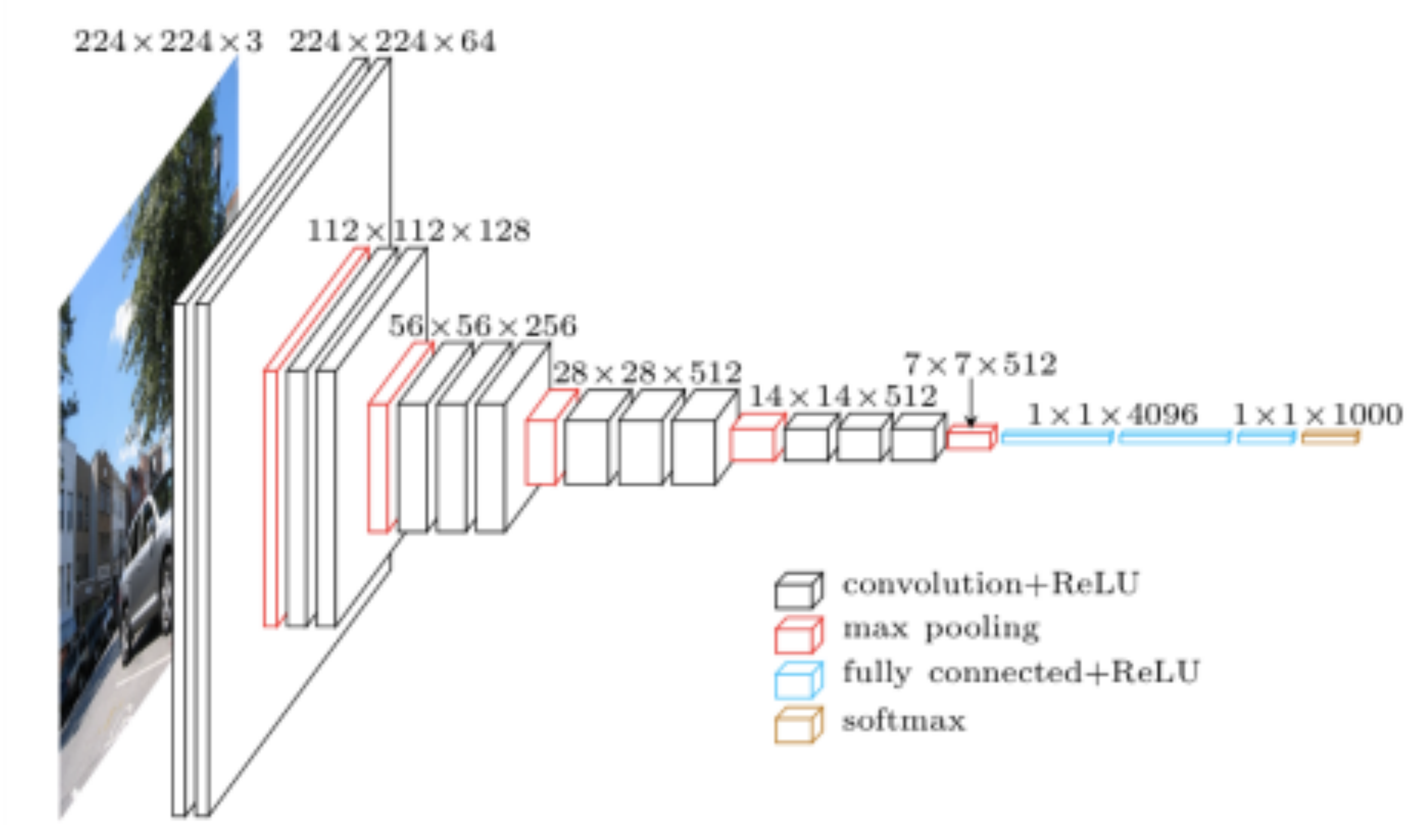


Conv2



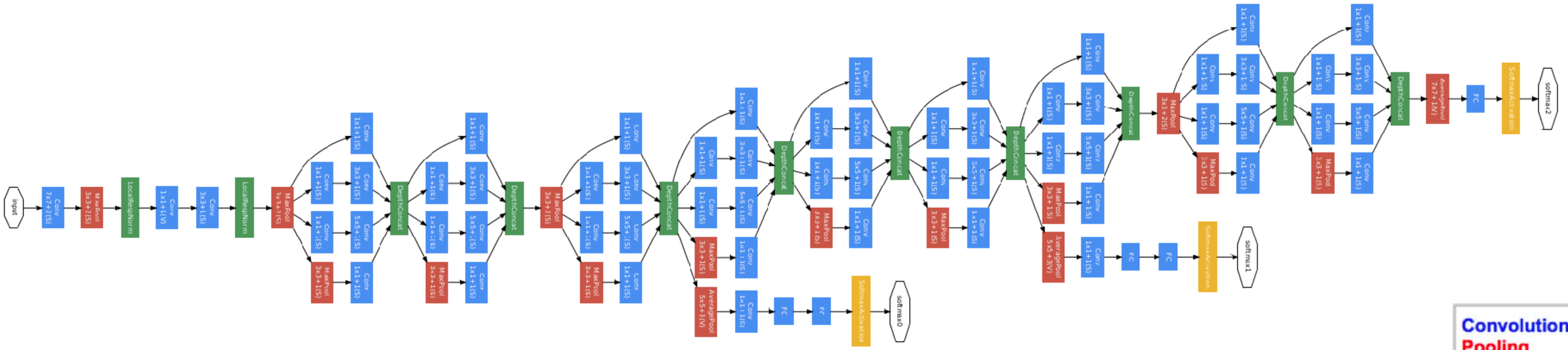

WE NEED TO GO
DEEPER
troll-face.ru

# VGG Net

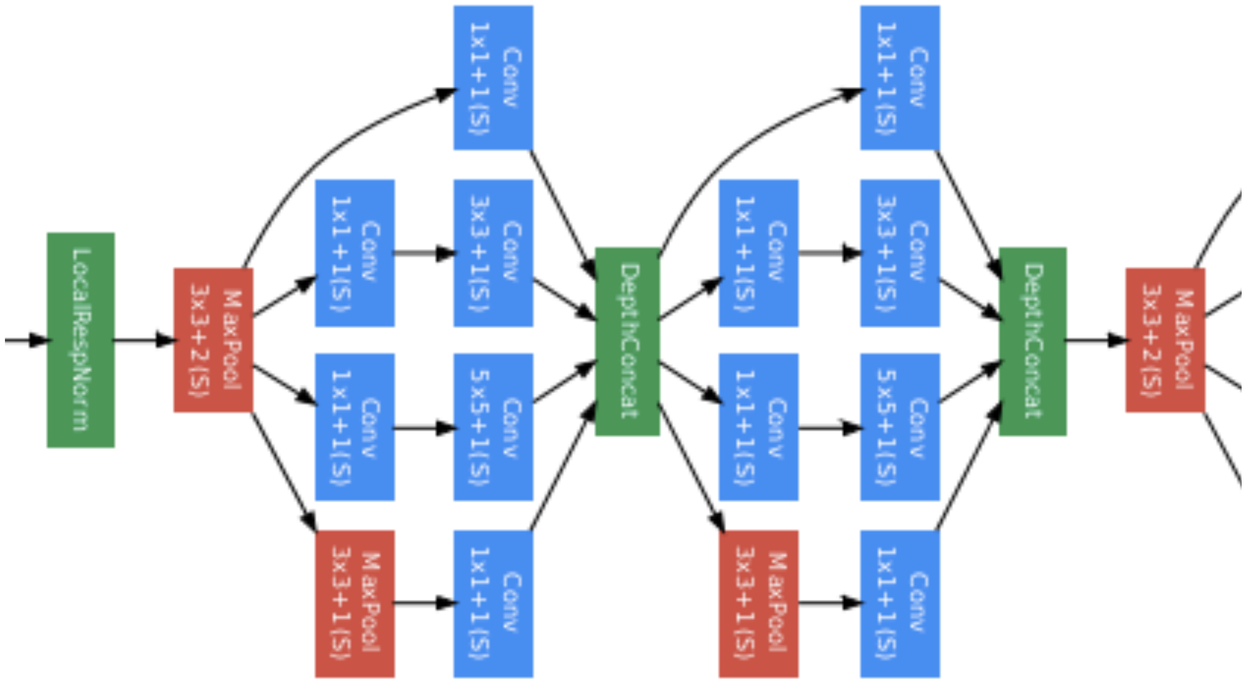

224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

# GoogleNet

22 layers, but 12 times less parameters than AlexNet

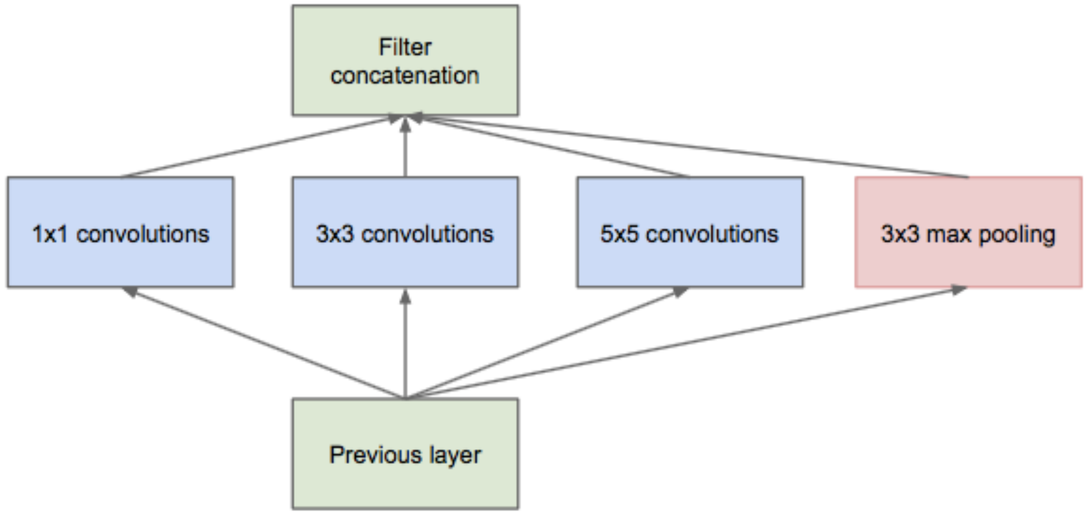


# GoogleNet


LocalRespNorm
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 1x1+1(S)
Conv 5x5+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 1x1+1(S)
Conv 5x5+1(S)
MaxPool 3x3+1(S)
Conv 1x1+1(S)
DepthConcat
MaxPool 3x3+2(S)

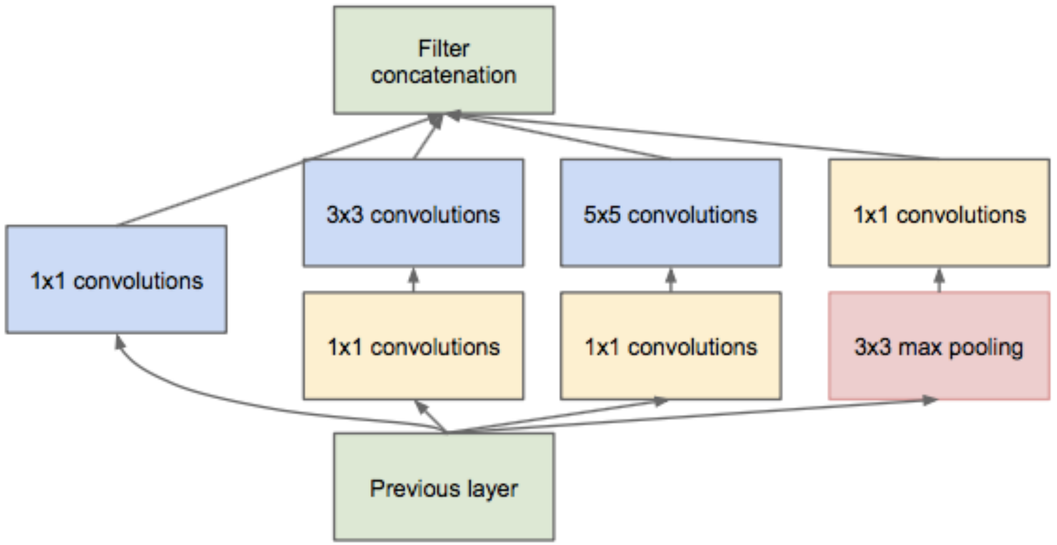
# Inception Module: Naive Version

**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



# GoogleNet

| type | patch size/ stride | output size | depth | #1×1 | #3×3 reduce | #3×3 | #5×5 reduce | #5×5 | pool proj | params | ops |
|---|---|---|---|---|---|---|---|---|---|---|---|
| convolution | 7×7/2 | 112×112×64 | 1 | | | | | | | 2.7K | 34M |
| max pool | 3×3/2 | 56×56×64 | 0 | | | | | | | | |
| convolution | 3×3/1 | 56×56×192 | 2 | | 64 | 192 | | | | 112K | 360M |
| max pool | 3×3/2 | 28×28×192 | 0 | | | | | | | | |
| inception (3a) | | 28×28×256 | 2 | 64 | 96 | 128 | 16 | 32 | 32 | 159K | 128M |
| inception (3b) | | 28×28×480 | 2 | 128 | 128 | 192 | 32 | 96 | 64 | 380K | 304M |
| max pool | 3×3/2 | 14×14×480 | 0 | | | | | | | | |
| inception (4a) | | 14×14×512 | 2 | 192 | 96 | 208 | 16 | 48 | 64 | 364K | 73M |
| inception (4b) | | 14×14×512 | 2 | 160 | 112 | 224 | 24 | 64 | 64 | 437K | 88M |
| inception (4c) | | 14×14×512 | 2 | 128 | 128 | 256 | 24 | 64 | 64 | 463K | 100M |
| inception (4d) | | 14×14×528 | 2 | 112 | 144 | 288 | 32 | 64 | 64 | 580K | 119M |
| inception (4e) | | 14×14×832 | 2 | 256 | 160 | 320 | 32 | 128 | 128 | 840K | 170M |
| max pool | 3×3/2 | 7×7×832 | 0 | | | | | | | | |
| inception (5a) | | 7×7×832 | 2 | 256 | 160 | 320 | 32 | 128 | 128 | 1072K | 54M |
| inception (5b) | | 7×7×1024 | 2 | 384 | 192 | 384 | 48 | 128 | 128 | 1388K | 71M |
| avg pool | 7×7/1 | 1×1×1024 | 0 | | | | | | | | |
| dropout (40%) | | 1×1×1024 | 0 | | | | | | | | |
| linear | | 1×1×1000 | 1 | | | | | | | 1000K | 1M |
| softmax | | 1×1×1000 | 0 | | | | | | | | |

# ImageNet Challenge 2012-2014

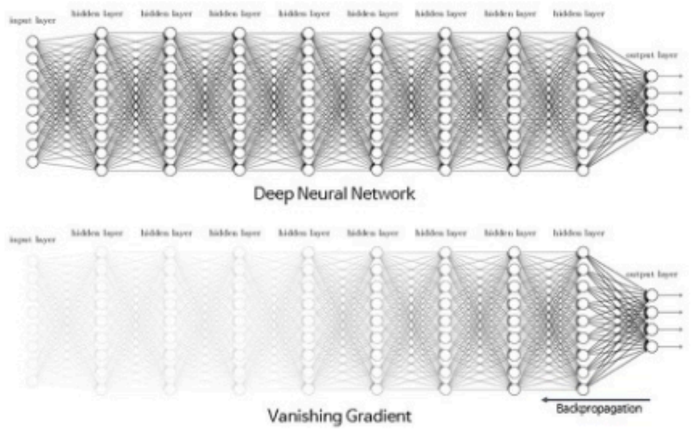
| Team | Year | Place | Error (top-5) | External data |
|---|---|---|---|---|
| SuperVision – Toronto (7 layers) | 2012 | - | 16.4% | no |
| SuperVision | 2012 | 1st | 15.3% | ImageNet 22k |
| Clarifai – NYU (7 layers) | 2013 | - | 11.7% | no |
| Clarifai | 2013 | 1st | 11.2% | ImageNet 22k |
| VGG – Oxford (16 layers) | 2014 | 2nd | 7.32% | no |
| GoogLeNet (19 layers) | 2014 | 1st | 6.67% | no |
| Human expert* | | | 5.1% | |

Conclusion:

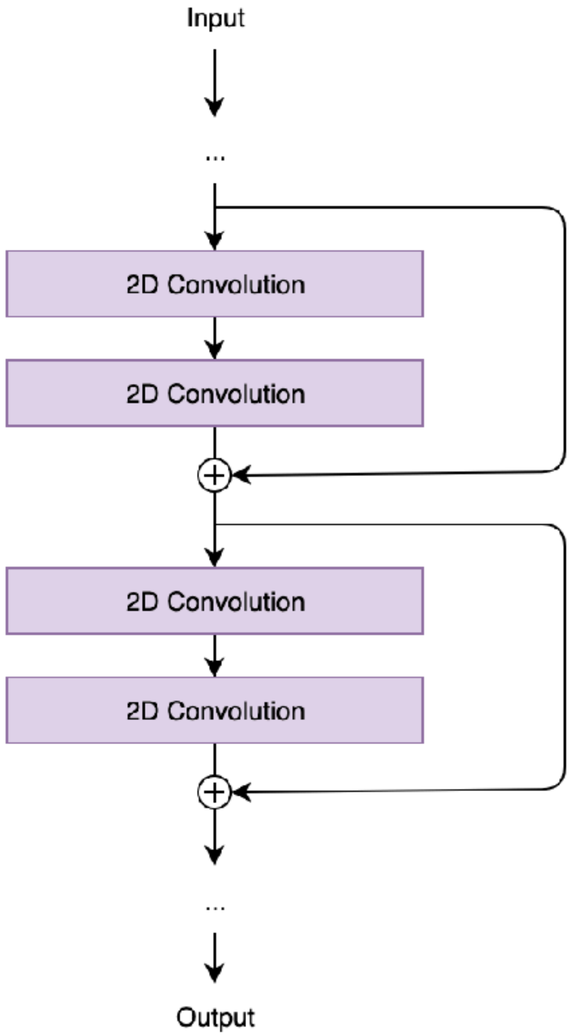
**More layers is better**


WE NEED TO GO
DEEPER
troll-face.ru

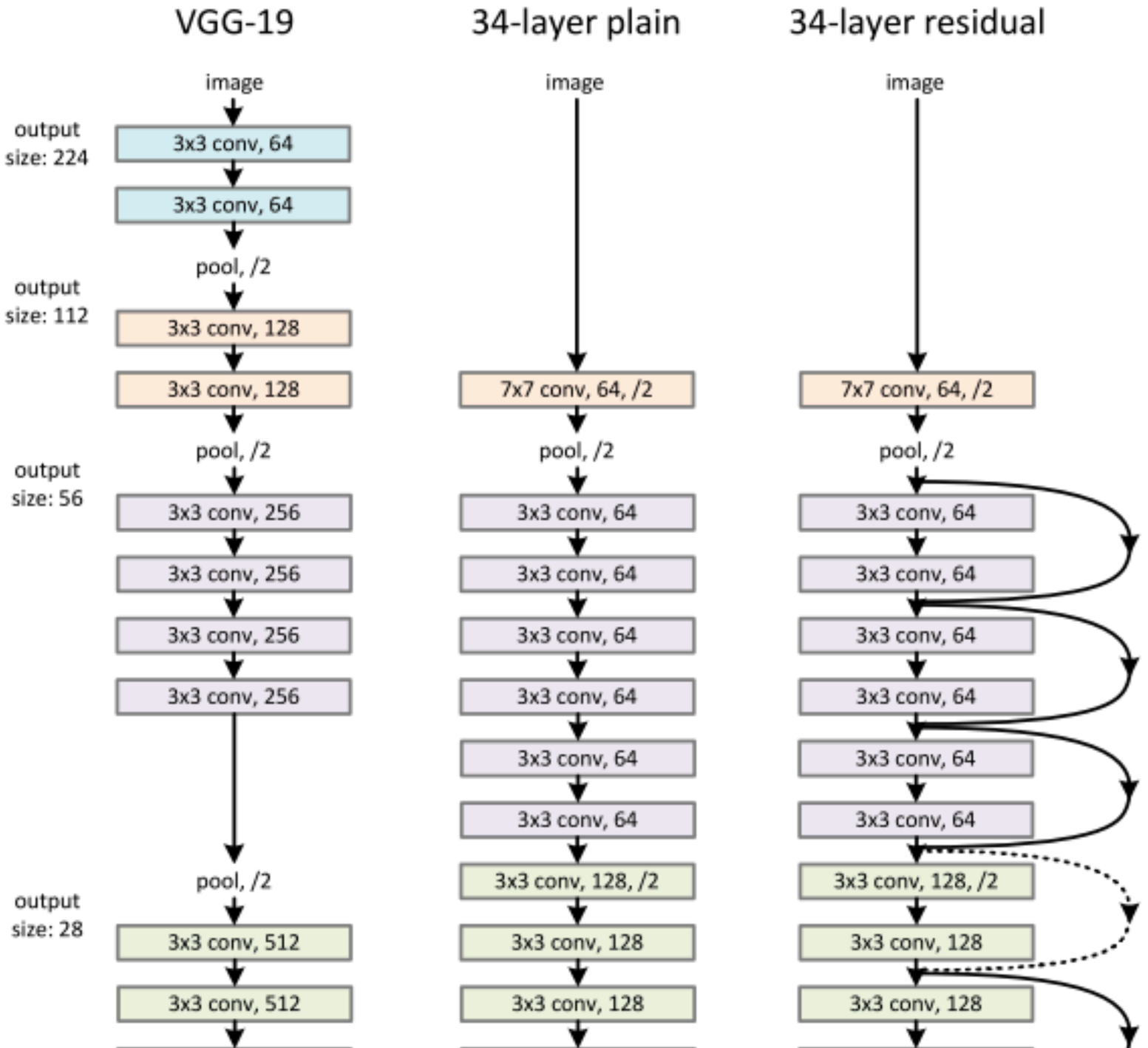
# Not everything is that simple:
# **Vanishing Gradient Problem**

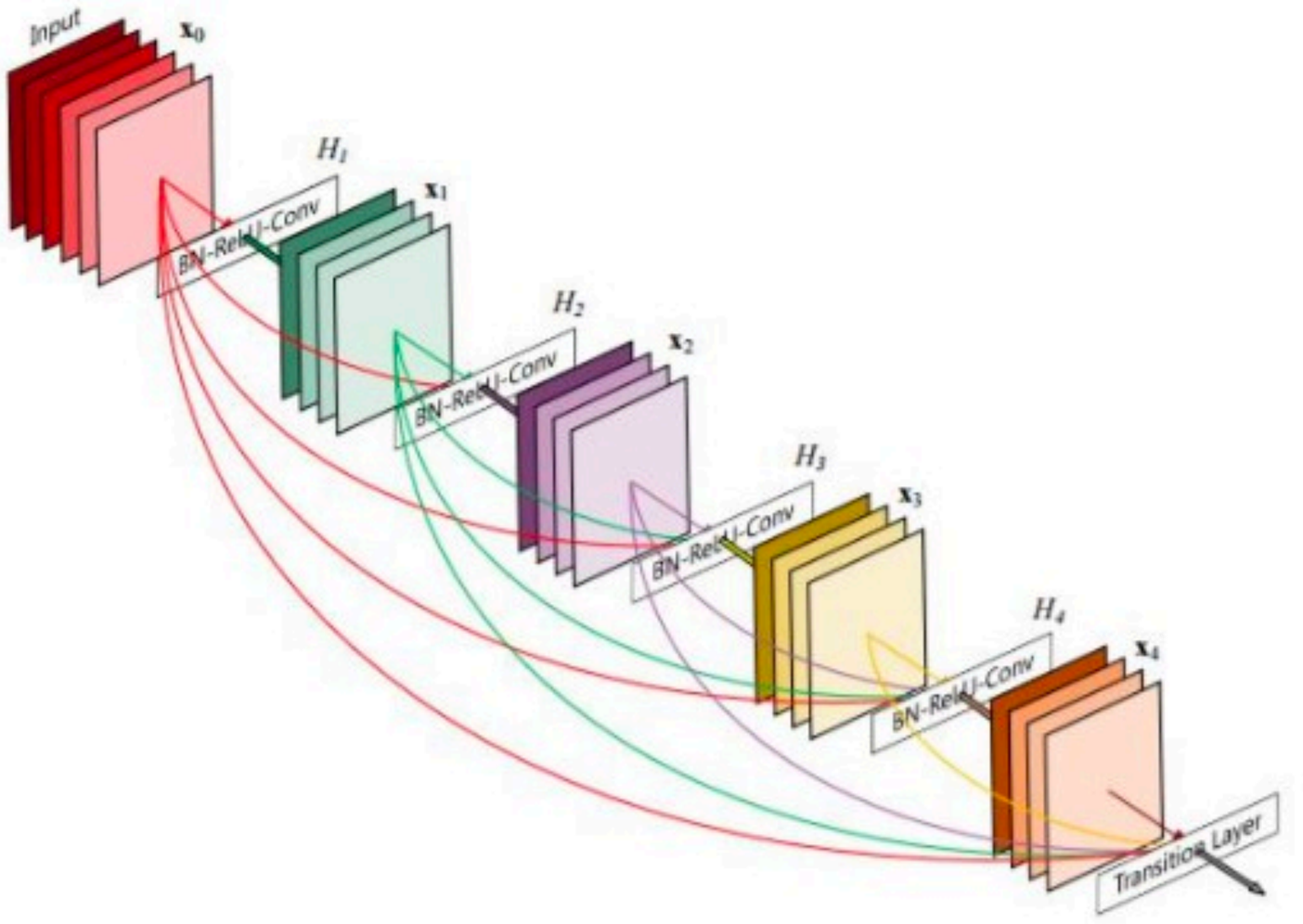

As the gradient keeps flowing backward to the initial layers, this value keeps getting multiplied by each local gradient. Hence, the gradient becomes smaller and smaller, making the updates to the initial layers very small, increasing the training time considerably.




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

# There is more… DenseNet (a variant of ResNet)



# Training a CNN

- Backpropagation + stochastic gradient descent with momentum
- Dropout
- Data Augmetation
- Batch Normalization
- Initialization
  - Transfer Learning

# Reducing Overfitting

**Data Augmentation**

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

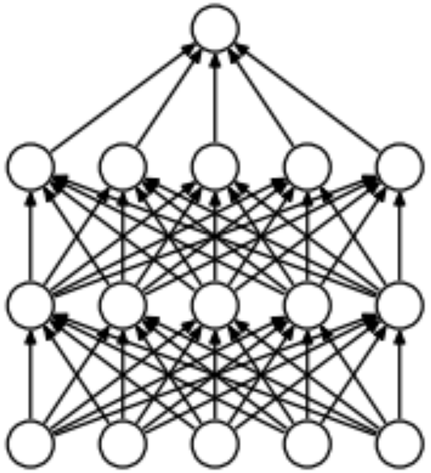
Crop 224x224 patches (and their horizontal reflections)

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

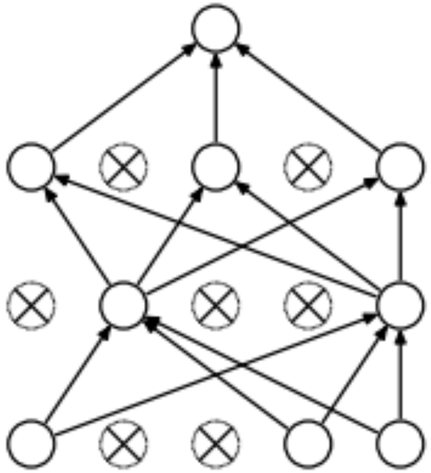
# Reducing Overfitting

**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.

# Hands on time!

# Sentiment Analysis

# Sentiment Analysis

# Sentiment Analysis: on IMDB dataset

- Kaggle Challenge for Binary sentiment classification on movie reviews
- Online demo:
  - http://nlp.stanford.edu:8080/sentiment/rntnDemo.html

# Sentiment Analysis: on IMDB dataset

## User Reviews

★★★★★★★★★★ **Brutal, honest, and a must see movie**

30 July 2001 | by tdao360 (Los Angeles, CA) – See all my reviews

This ranks up there as one of the three most powerful movies I have ever seen in my lifetime (Full Metal Jacket and Grave of The Fireflies being the other two). This movie shows the brutal honest side of addiction and over-indulgence. Not just drugs, although it heavily shows drug addiction. Also shows how one addiction can lead to another and how damaging it can be for you. I watched this alone, and felt so stunned afterwards, I had to call a friend just to calm my nerves. Seriously, this is a brutal (one more time) BRUTAL film. The acting is wonderful - Ellyn Burnstyn and Jenniffer Connely are just wonderful in this movie, and Marlon Wayons was such a shocker in a serious role. Everyone must watch it, for it's entertainment value, and more importantly, it's educational value. But it leaves chills down your spine for it's honesty and unforgiving lessons.

# Dataset for IMDB Movie reviews sentiment classification:

- Dataset of 50,000 movies reviews from IMDB, labeled by sentiment (positive/negative).

- Reviews have been **preprocessed**, and each review is **encoded as a sequence of word** indexes (integers). For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes the 3rd most frequent word in the data. This allows for quick filtering operations such as: "only consider the top 10,000 most common words, but eliminate the top 20 most common words".

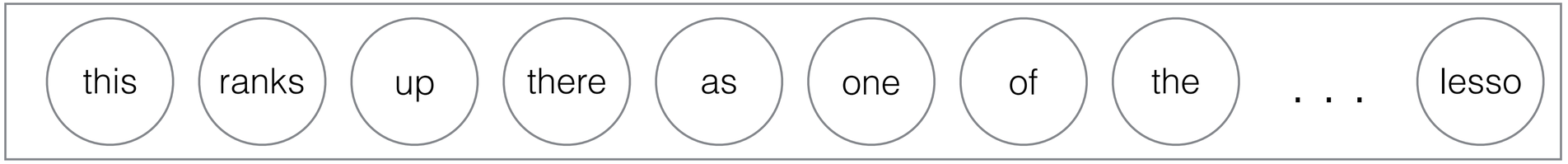
# Sentiment Analysis: on IMDB dataset

## User Reviews

★★★★★★★★★★ **Brutal, honest, and a must see movie**

30 July 2001 | by tdao360 (Los Angeles, CA) – See all my reviews

This ranks up there as one of the three most powerful movies I have ever seen in my lifetime (Full Metal Jacket and Grave of The Fireflies being the other two). This movie shows the brutal honest side of addiction and over-indulgence. Not just drugs, although it heavily shows drug addiction. Also shows how one addiction can lead to another and how damaging it can be for you. I watched this alone, and felt so stunned afterwards, I had to call a friend just to calm my nerves. Seriously, this is a brutal (one more time) BRUTAL film. The acting is wonderful - Ellyn Burnstyn and Jenniffer Connely are just wonderful in this movie, and Marlon Wayons was such a shocker in a serious role. Everyone must watch it, for it's entertainment value, and more importantly, it's educational value. But it leaves chills down your spine for it's honesty and unforgiving lessons.



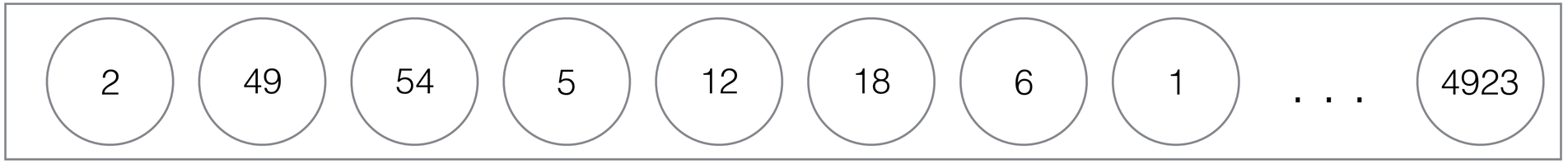
# Sentiment Analysis: on IMDB dataset

## User Reviews

★★★★★★★★★★ **Brutal, honest, and a must see movie**

30 July 2001 | by tdao360 (Los Angeles, CA) – See all my reviews

This ranks up there as one of the three most powerful movies I have ever seen in my lifetime (Full Metal Jacket and Grave of The Fireflies being the other two). This movie shows the brutal honest side of addiction and over-indulgence. Not just drugs, although it heavily shows drug addiction. Also shows how one addiction can lead to another and how damaging it can be for you. I watched this alone, and felt so stunned afterwards, I had to call a friend just to calm my nerves. Seriously, this is a brutal (one more time) BRUTAL film. The acting is wonderful - Ellyn Burnstyn and Jenniffer Connely are just wonderful in this movie, and Marlon Wayons was such a shocker in a serious role. Everyone must watch it, for it's entertainment value, and more importantly, it's educational value. But it leaves chills down your spine for it's honesty and unforgiving lessons.



# Word Embedding

- A recent breakthrough in the field of natural language processing is called word embedding.

- Words are encoded as real-valued vectors in a high dimensional space, where the similarity between words in terms of meaning translates to closeness in the vector space.

# Learning Dense Embedding

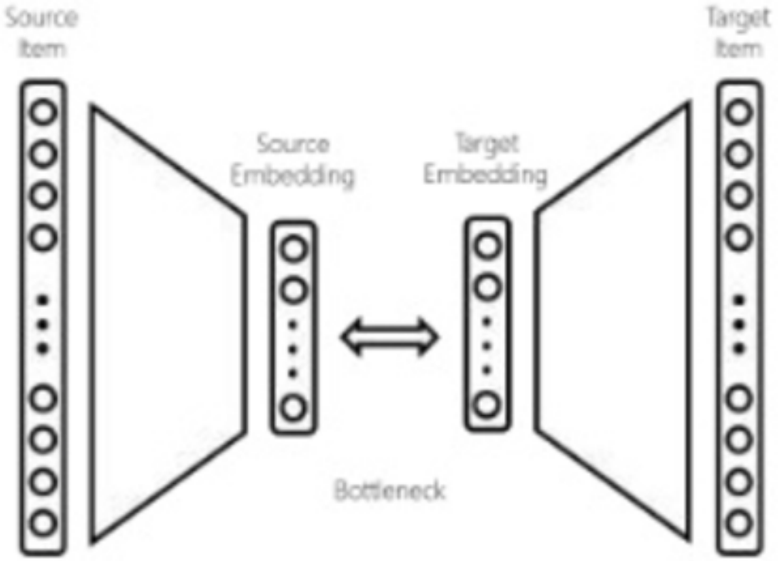
- Matrix Factorization

E.g. LDA, GloVe

| | $Context_1$ | $Context_1$ | .... | $Context_k$ |
|---|---|---|---|---|
| $Word_1$ | | | | |
| $Word_2$ | | | | |
| ⋮ | | | | |
| $Word_n$ | | | | |

- Neural Networks

word2vect



# Sentiment Analysis with Keras

- Keras provides access to the IMDB dataset built-in.

- The words have been replaced by integers that indicate the absolute popularity of the word in the dataset.

- Sentences in each review are comprised of a sequence of integers.

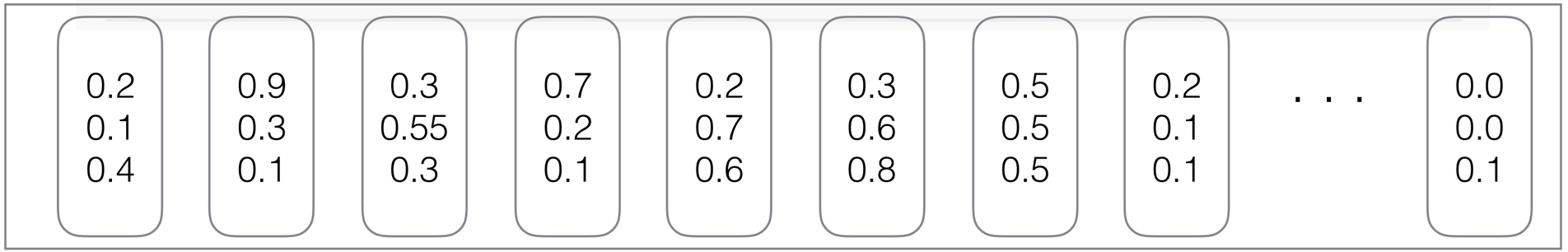
# Sentiment Analysis: on IMDB dataset

## User Reviews

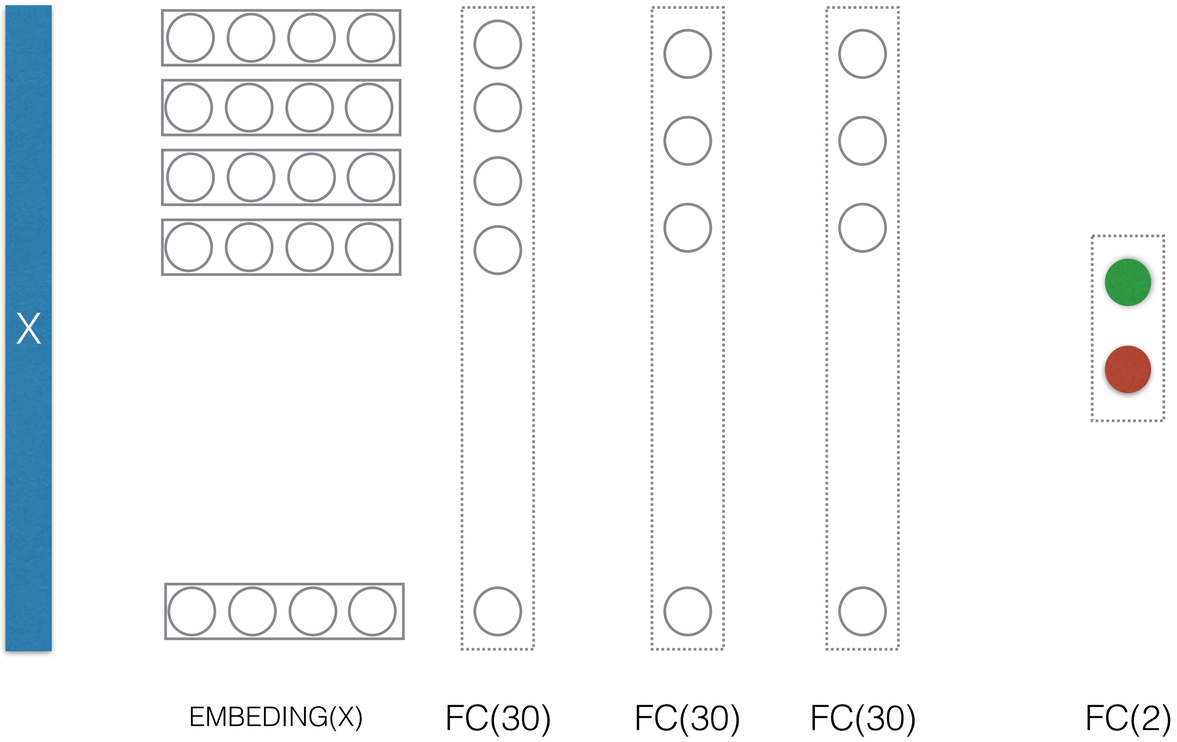
★★★★★★★★★★ **Brutal, honest, and a must see movie**

30 July 2001 | by tdao360 (Los Angeles, CA) – See all my reviews

This ranks up there as one of the three most powerful movies I have ever seen in my lifetime (Full Metal Jacket and Grave of The Fireflies being the other two). This movie shows the brutal honest side of addiction and over-indulgence. Not just drugs, although it heavily shows drug addiction. Also shows how one addiction can lead to another and how damaging it can be for you. I watched this alone, and felt so stunned afterwards, I had to call a friend just to calm my nerves. Seriously, this is a brutal (one more time) BRUTAL film. The acting is wonderful - Ellyn Burnstyn and Jenniffer Connely are just wonderful in this movie, and Marlon Wayons was such a shocker in a serious role. Everyone must watch it, for it's entertainment value, and more importantly, it's educational value. But it leaves chills down your spine for it's honesty and unforgiving lessons.



# Let’s try to do it..



# But

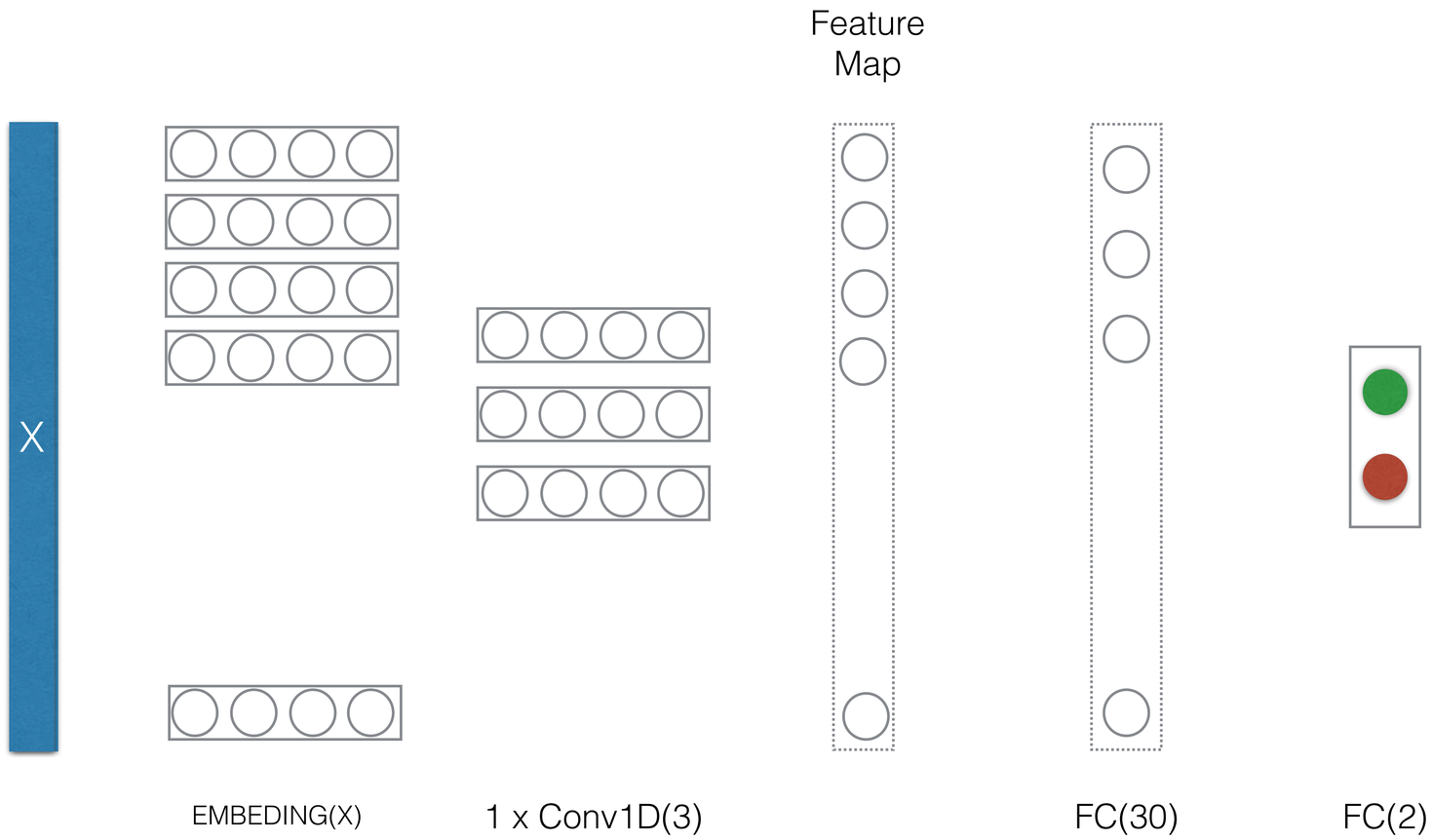
**“really nice"**

is not the same than

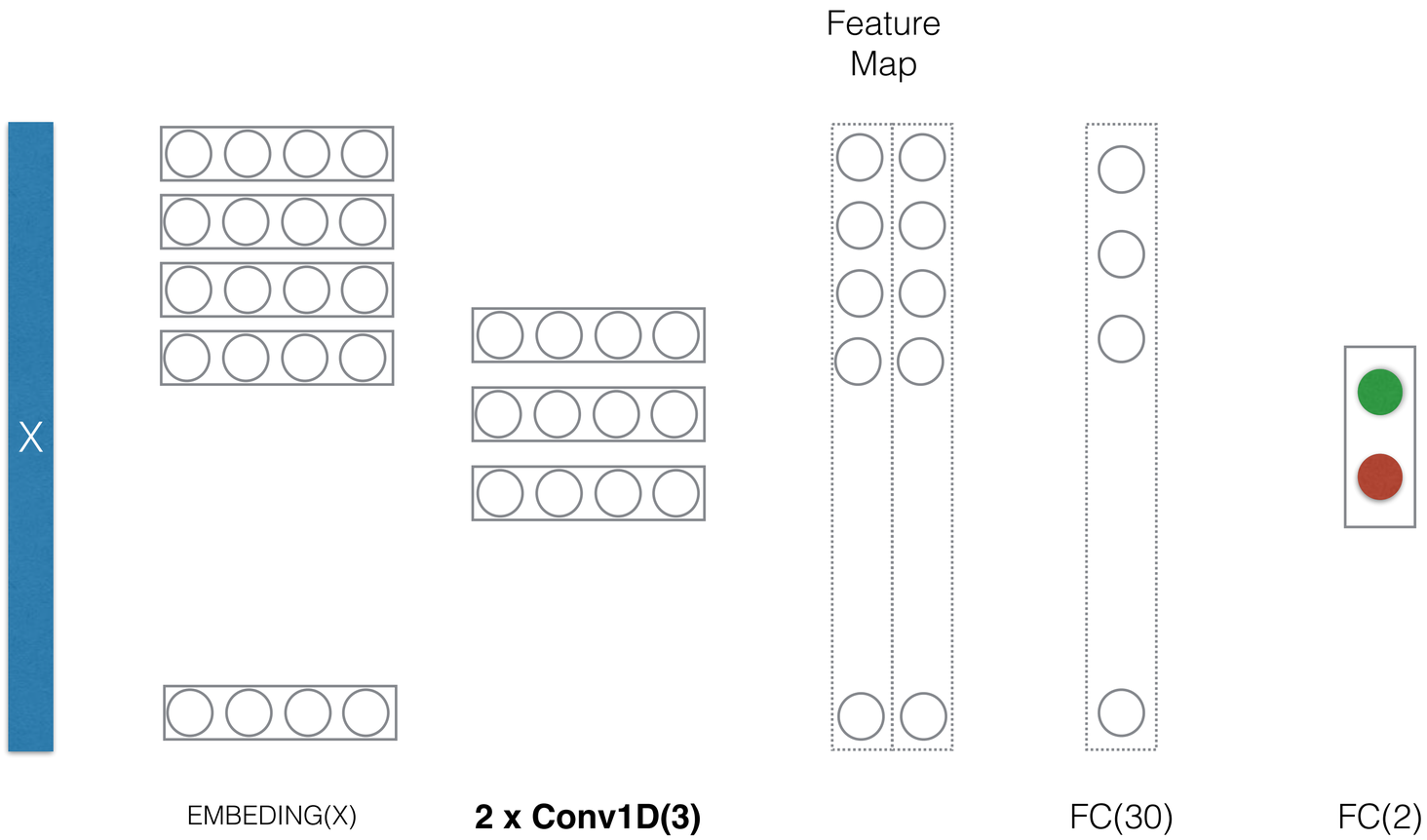
**“not nice”**

# Hands on time!
# Can we use a convoluational layer

# With 1D Convolutions



# With 1D Convolutions



# Recurrent Neural Networks

Classical neural networks, including convolutional ones, suffer from **two severe limitations**:

- **They only accept a fixed-sized vector as input and produce a fixed-sized vector as output.**

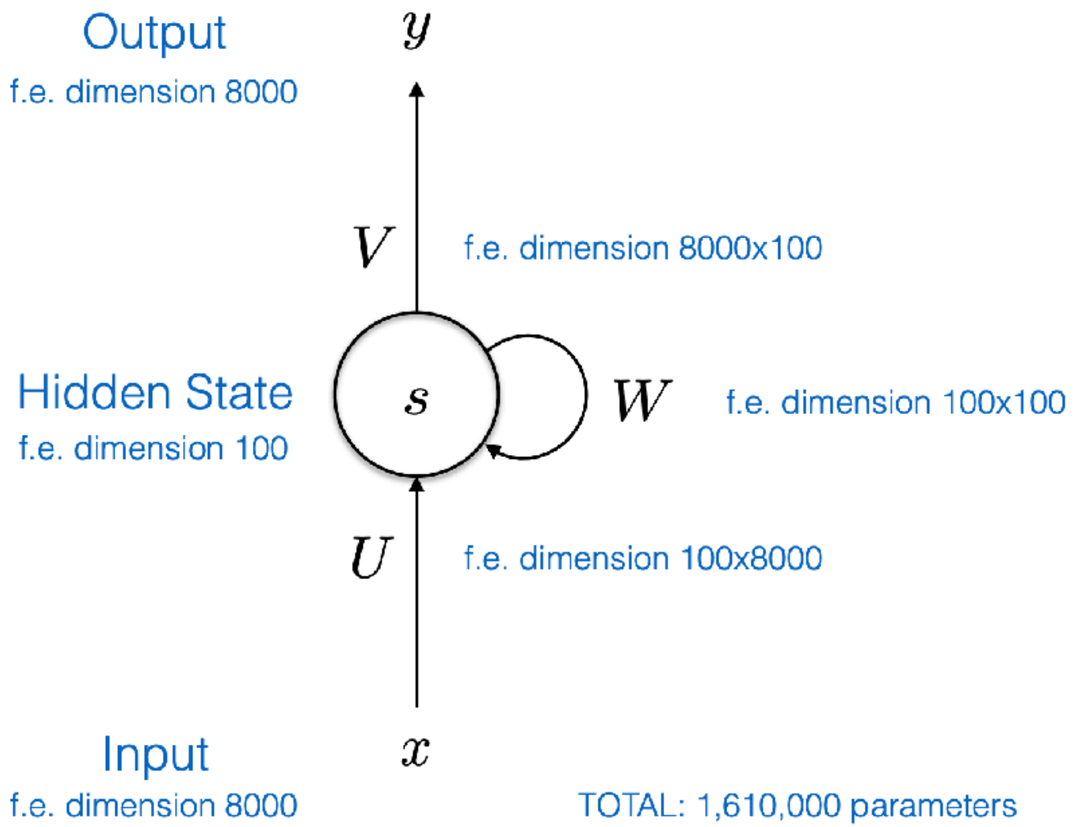
- **They do not consider the sequential nature of some data (language, video frames, time series, etc.)**

**Recurrent neural networks (RNN)** overcome these limitations by allowing to operate over sequences of vectors (in the input, in the output, or both).

**RNN** have shown **success** in:

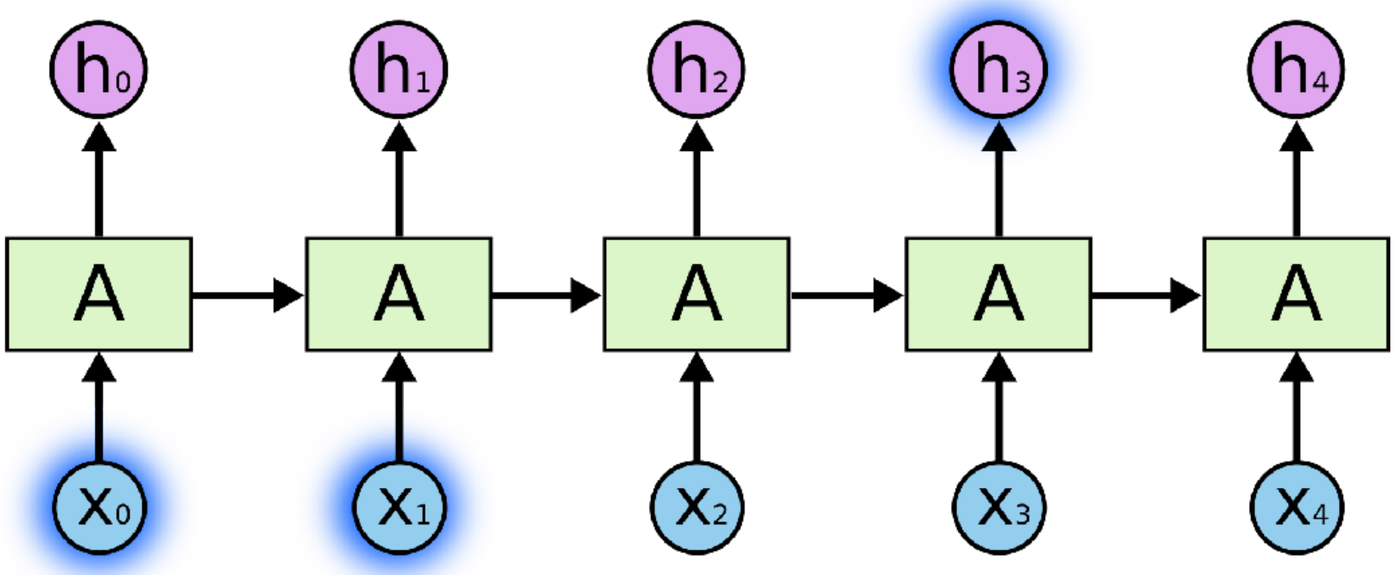
- Language modeling and generation.
- Machine Translation.
- Speech Recognition.
- Image Description.
- Question Answering.
- Etc.

# Vanilla Recurrent Neural Network



the clouds are in the ***?***

the clouds are in the ***sky***

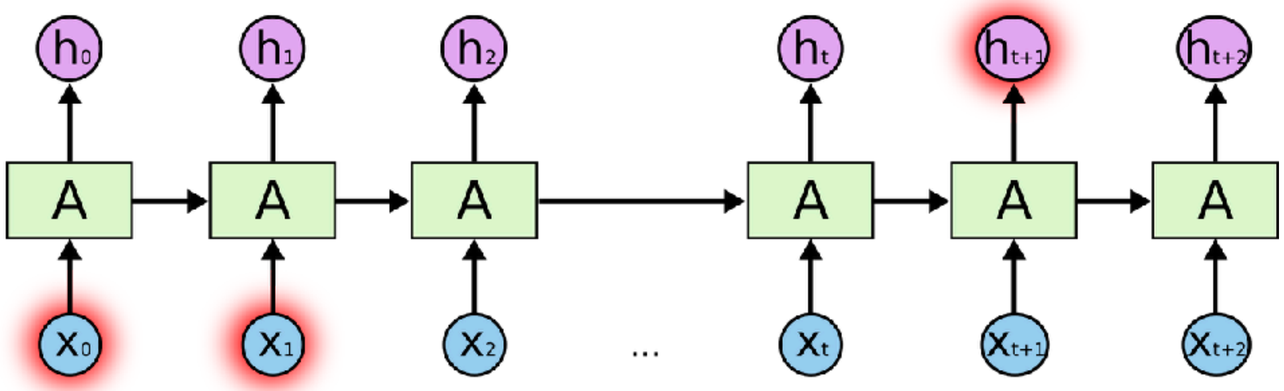


I grew up in France….. … …
… … .. … …
…. … … … I speak fluent ***?***.

I grew up in France….. … …

… … .. … …

…. … … … I speak fluent ***?***.

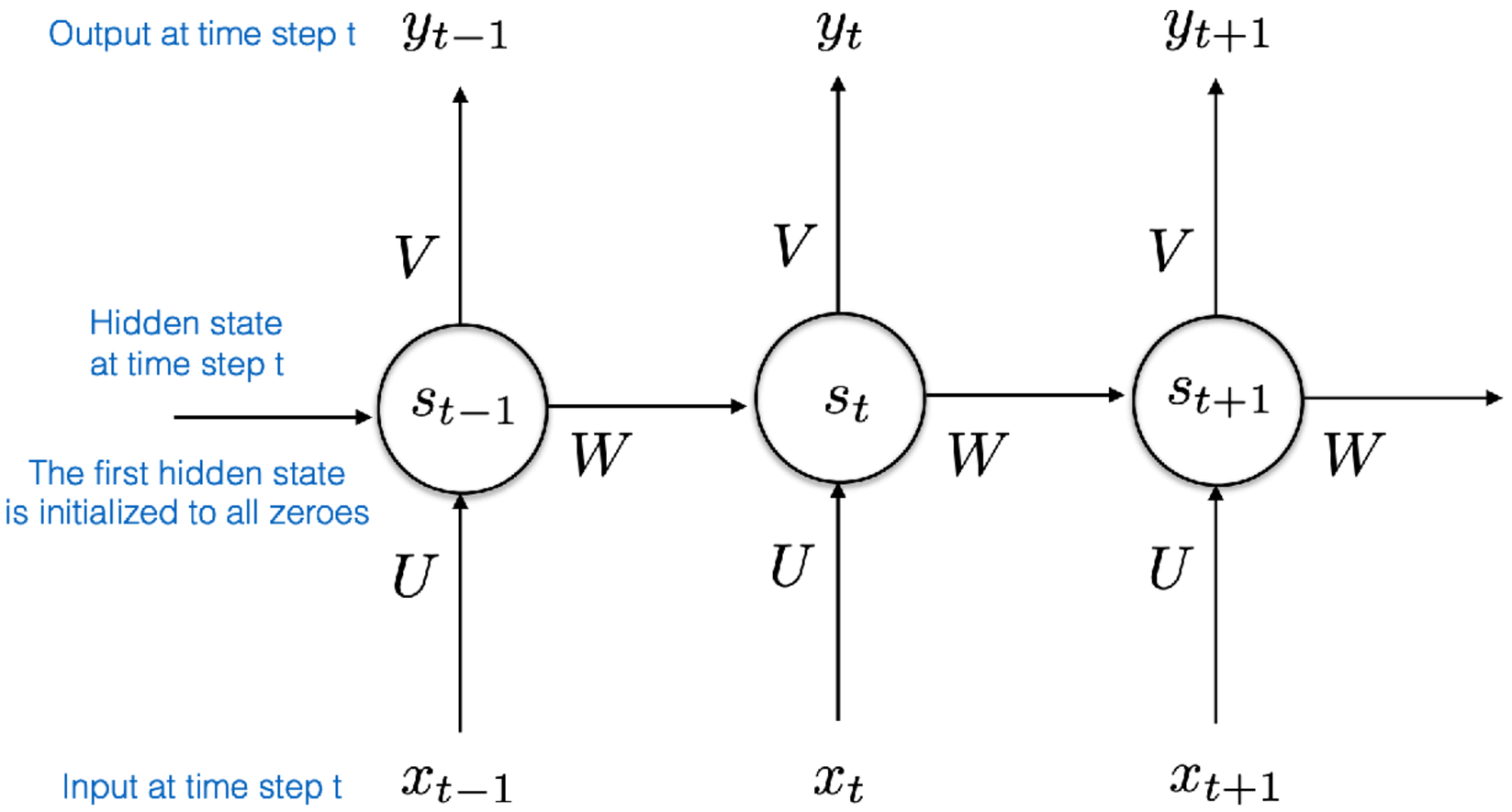


# Unrolling in time of a RNN

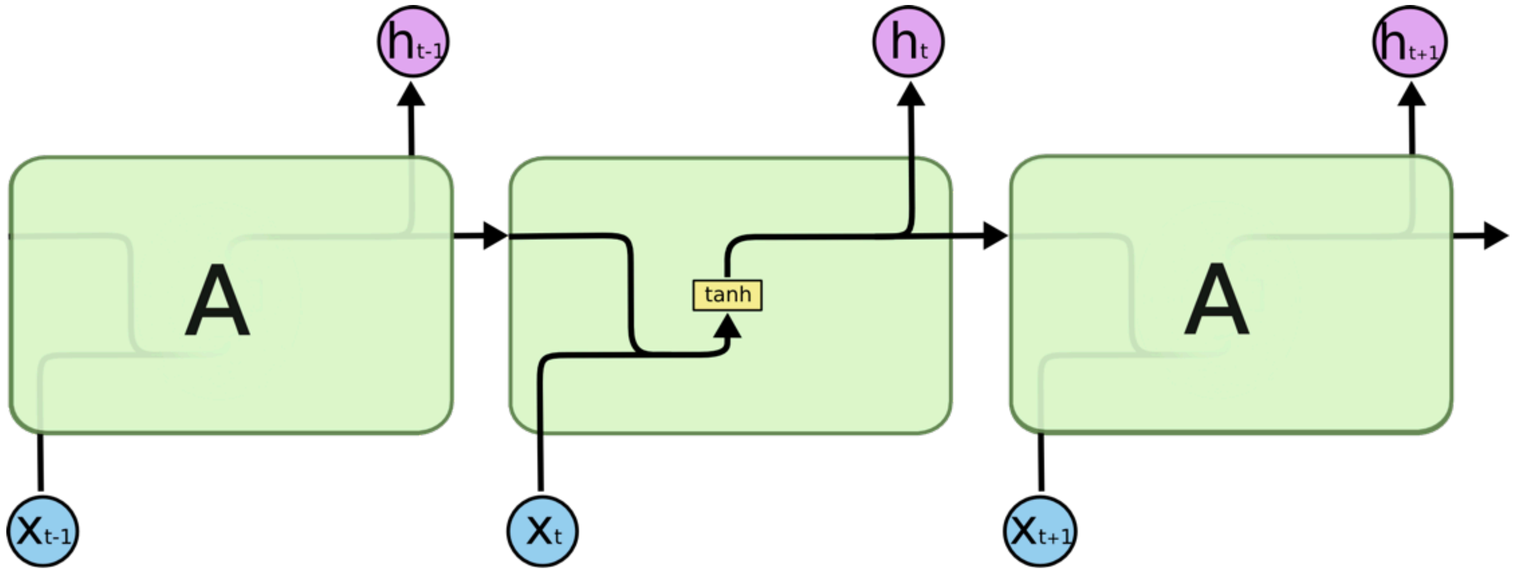
By unrolling we mean that we write out the network for the complete sequence.

Basic equations of the RNN

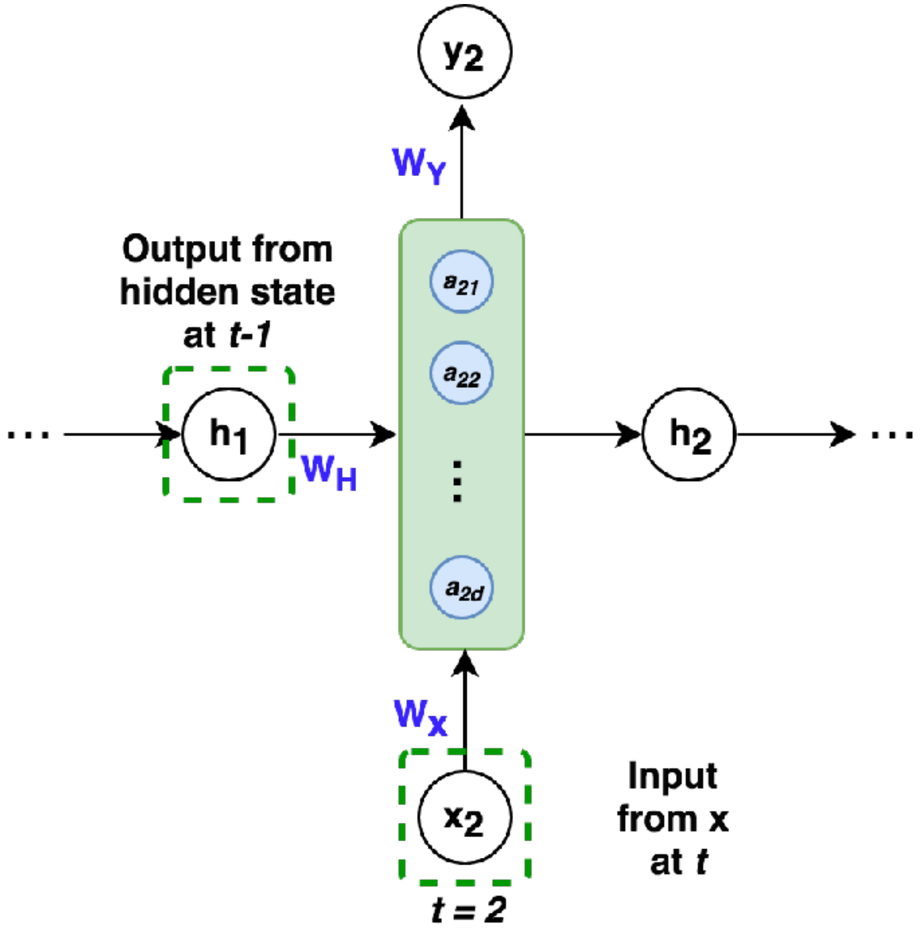
$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$y_t = \mathrm{softmax}(Vs_t)$$



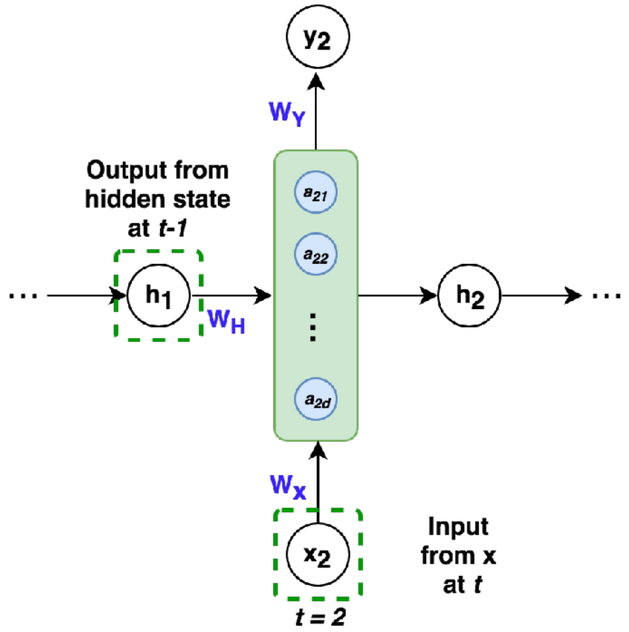
# Vanilla Recurrent Neural Network



# Vanilla Recurrent Neural Network



# Vanilla Recurrent Neural Network




$$d = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad o = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad g = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad s = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

One-Hot Encoding of the word "dogs"

$$a_t = W_H h_{t-1} + W_X X_t$$ Hidden Nodes

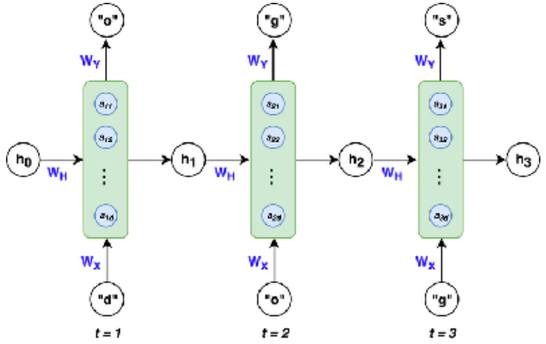
Activation Function

Output From Hidden State → $h_t = tanh(a_t)$

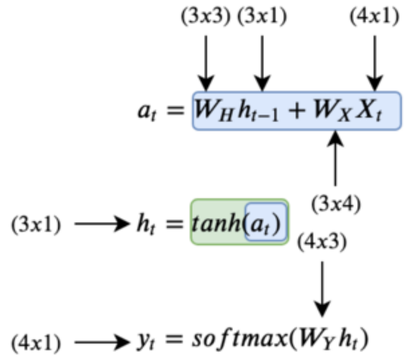
Hidden State

Prediction at time t → $y_t = softmax(W_Y h_t)$

https://medium.com/towards-artificial-intelligence/whirlwind-tour-of-rnns-a11effb7808f




If we use 3 hidden nodes çin our RNN *(d=3)*




At t=1

$$a_1 = \begin{pmatrix} W_{H,11} & W_{H12} & W_{H,13} \\ W_{H,21} & W_{H,22} & W_{H,23} \\ W_{H,31} & W_{H,32} & W_{H,33} \end{pmatrix} \begin{pmatrix} h_{0,1} \\ h_{0,2} \\ h_{0,3} \end{pmatrix} + \begin{pmatrix} W_{X,11} & W_{X,12} & W_{X,13} & W_{X,14} \\ W_{X,21} & W_{X,22} & W_{X,23} & W_{X,24} \\ W_{X,31} & W_{X,32} & W_{X,33} & W_{X,34} \end{pmatrix} \begin{pmatrix} x_{1,1} \\ x_{1,2} \\ x_{1,3} \\ x_{1,4} \end{pmatrix} = \begin{pmatrix} a_{1,1} \\ a_{1,2} \\ a_{1,3} \end{pmatrix}$$

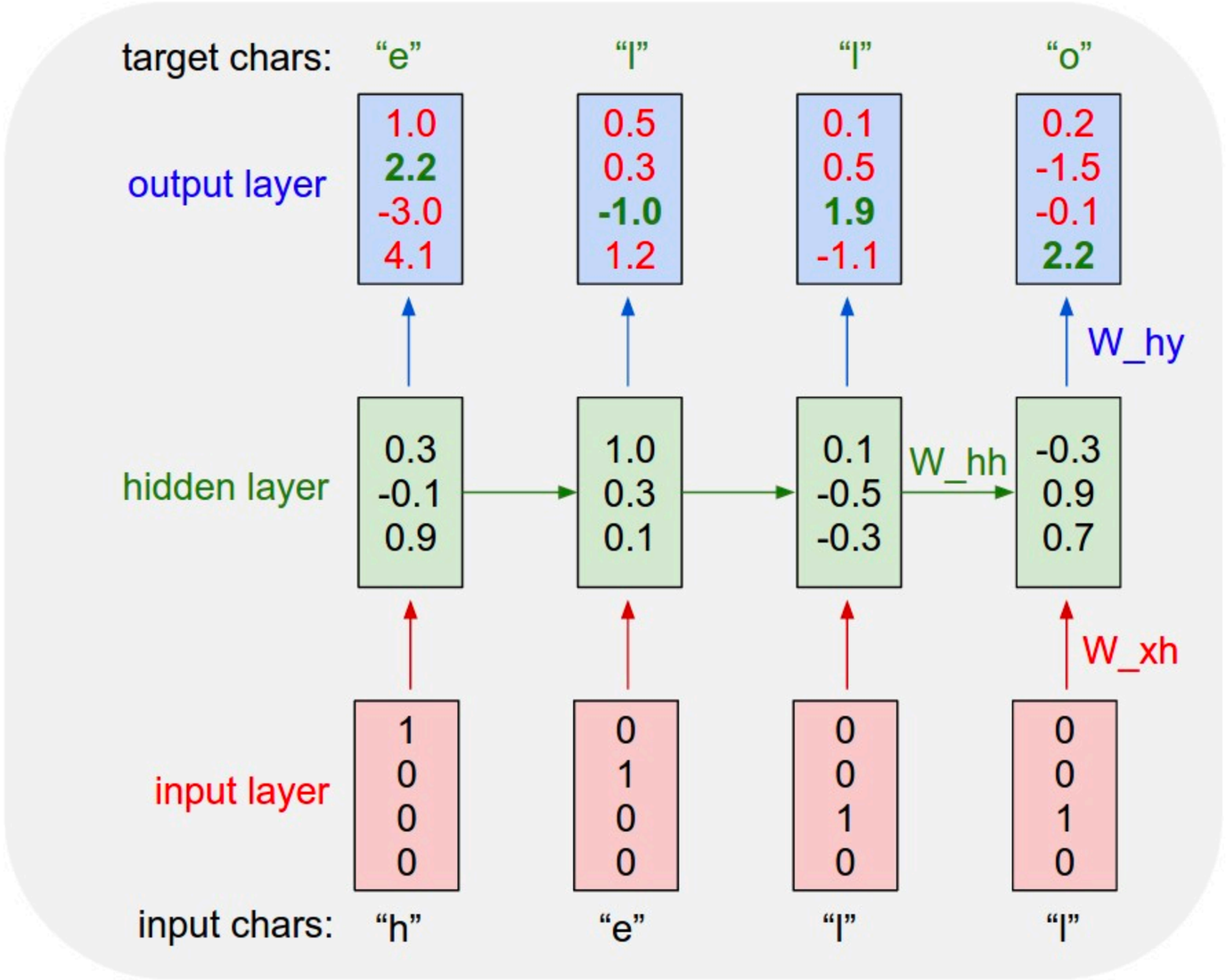
$$a_1 = \begin{pmatrix} 0.1 & 0.5 & 0.1 \\ 0.5 & 0.9 & 0.3 \\ 0.3 & 0.2 & 0.1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0.6 & 0.8 & 0.4 & 0.8 \\ 0.2 & 0.2 & 0.8 & 0.7 \\ 0.9 & 0.8 & 0.1 & 0.2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.6 \\ 0.2 \\ 0.9 \end{pmatrix}$$

$$h_1 = tanh(\begin{pmatrix} a_{1,1} \\ a_{1,2} \\ a_{1,3} \end{pmatrix}) = \begin{pmatrix} h_{1,1} \\ h_{1,2} \\ h_{1,3} \end{pmatrix}$$

$$h_1 = tanh(\begin{pmatrix} 0.6 \\ 0.2 \\ 0.9 \end{pmatrix}) = \begin{pmatrix} 0.54 \\ 0.20 \\ 0.72 \end{pmatrix}$$

$$y_1 = softmax(\begin{pmatrix} W_{Y,11} & W_{Y,12} & W_{Y,13} \\ W_{Y,21} & W_{Y,22} & W_{Y,23} \\ W_{Y,31} & W_{Y,32} & W_{Y,33} \\ W_{Y,41} & W_{Y,42} & W_{Y,43} \end{pmatrix} \begin{pmatrix} h_{1,1} \\ h_{1,2} \\ h_{1,3} \end{pmatrix}) = \begin{pmatrix} y_{1,1} \\ y_{1,2} \\ y_{1,3} \\ y_{1,4} \end{pmatrix}$$

$$y_1 = softmax(\begin{pmatrix} 0.9 & 0.8 & 0.3 \\ 0.2 & 0.3 & 0.4 \\ 0.6 & 0.9 & 0.1 \\ 0.5 & 0.0 & 0.3 \end{pmatrix} \begin{pmatrix} 0.54 \\ 0.20 \\ 0.72 \end{pmatrix}) = \begin{pmatrix} 0.32 \\ 0.21 \\ 0.24 \\ 0.22 \end{pmatrix}$$

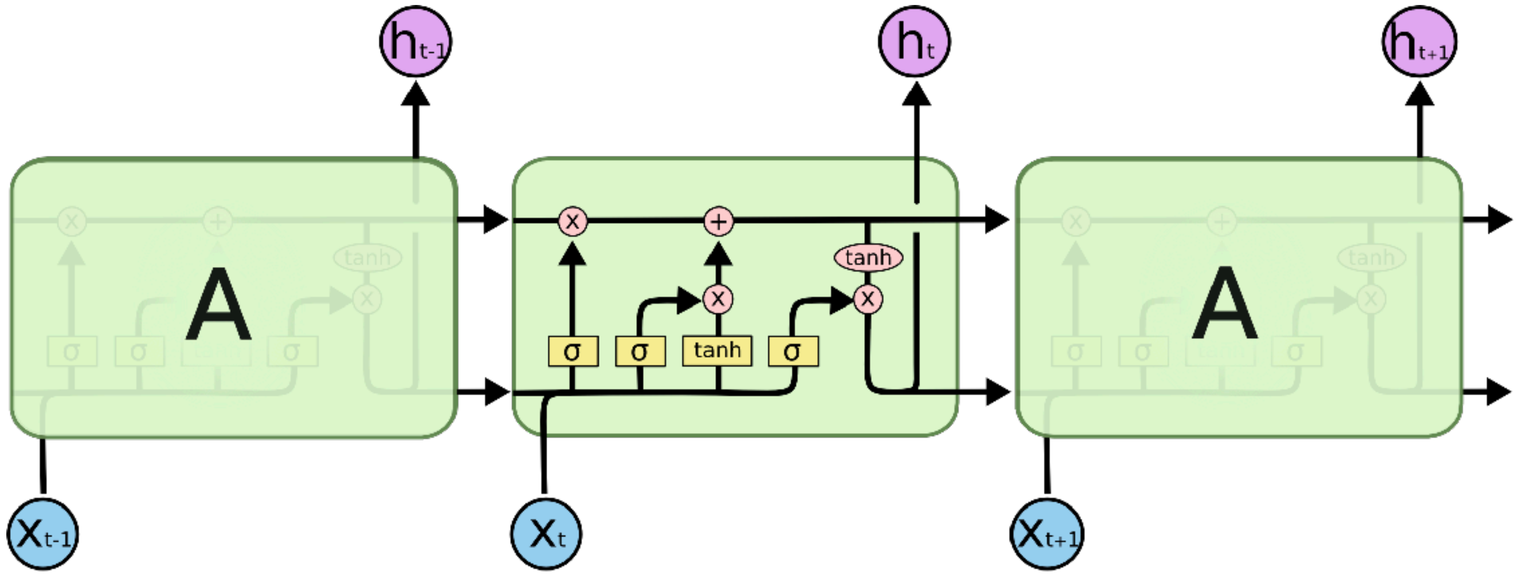

target chars:
“e”
“l”
“l”
“o”
output layer
1.0
2.2
-3.0
4.1
0.5
0.3
-1.0
1.2
0.1
0.5
1.9
-1.1
0.2
-1.5
-0.1
2.2
W_hy
hidden layer
0.3
-0.1
0.9
1.0
0.3
0.1
0.1
-0.5
-0.3
W_hh
-0.3
0.9
0.7
W_xh
input layer
1
0
0
0
0
1
0
0
0
0
1
0
0
0
1
0
input chars:
“h”
“e”
“l”
“l”

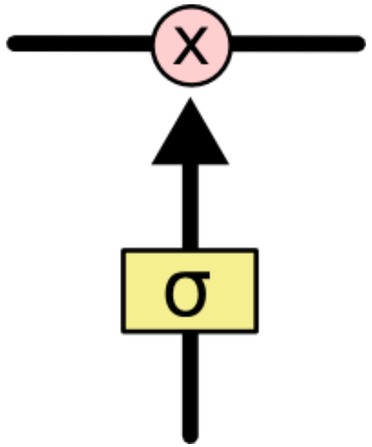
**vanishing gradient problem**. Vanishing gradients make it difficult for the model to learn long-term dependencies.
For example, if an RNN was given this sentence:

**The brown and black dog, which was playing with the cat, was a german shepherd.**

$(x_2)$ $(x_4)$ $(x_5)$ $(x_{14})$ $(x_{15})$

# LSTM

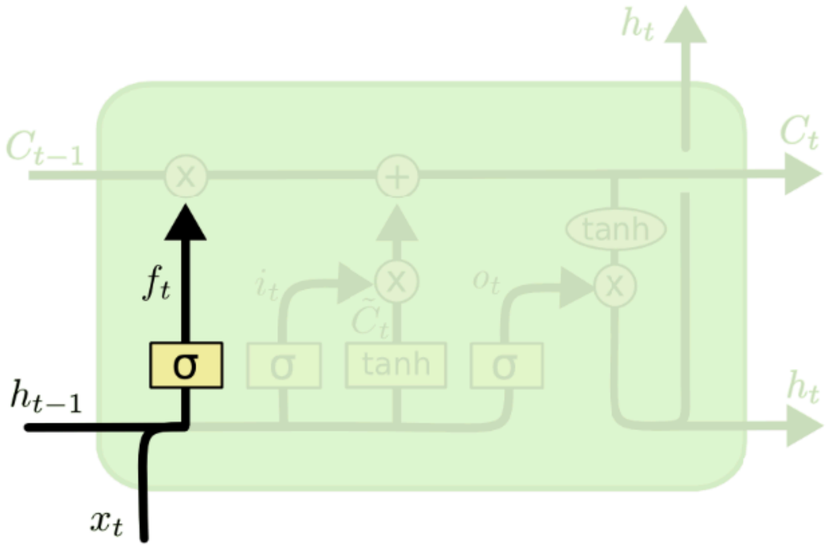





LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called **gates**.

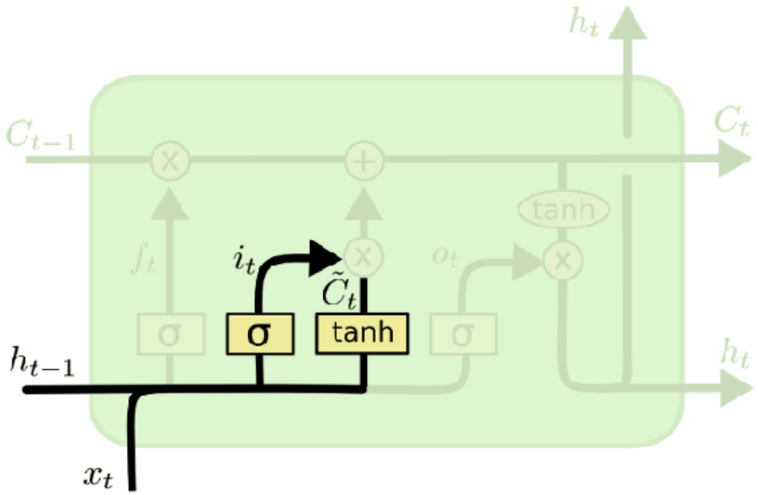
Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

# Forget Layer




$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

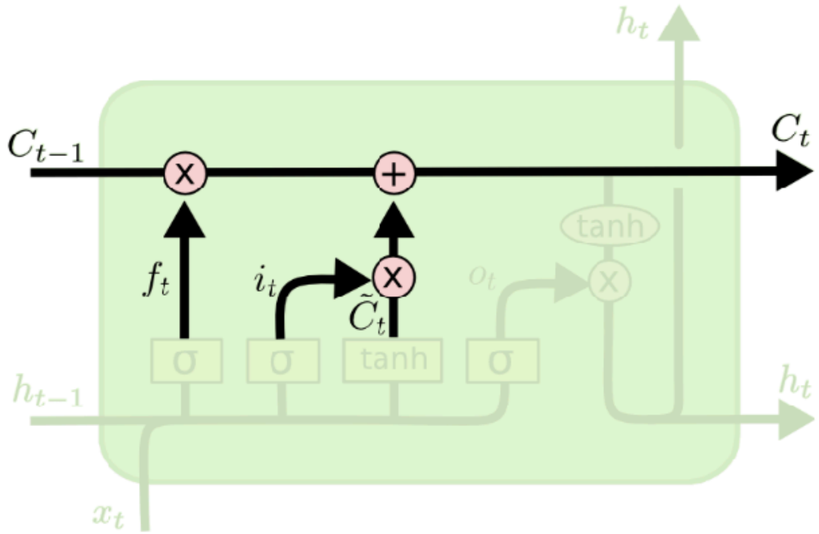
# Input gate layer




$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

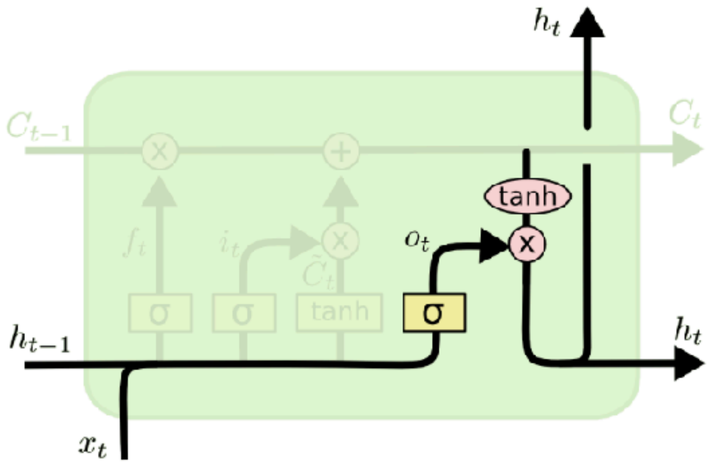
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

# ssdfd Layer




$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

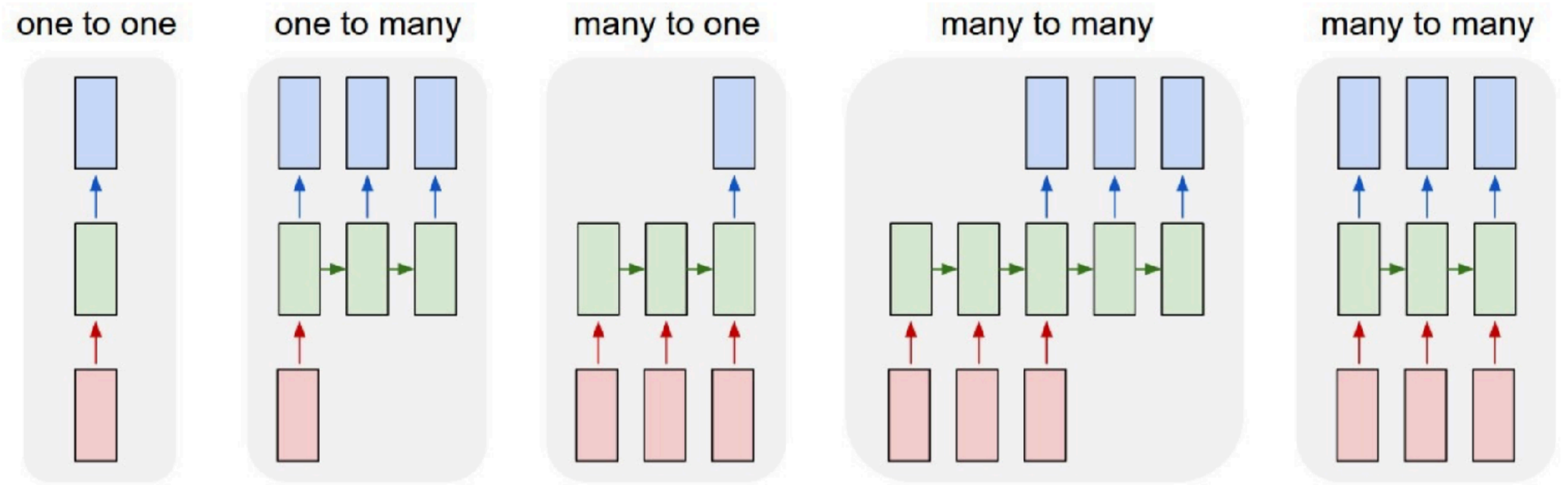
# output Layer




$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through  tanh (to push the values to be between -1 **and** 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.




Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

# The power of RNN

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

http://cs.stanford.edu/people/karpathy/deepimagesent/

# Hands on time!

# Unsupervised learning

# Layout

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

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.

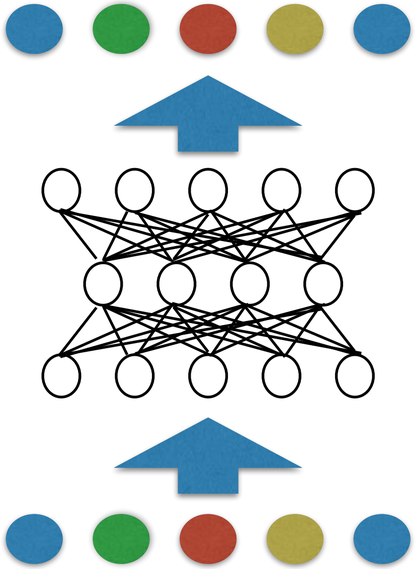
- Manifold learning - Understanding how data is distributed in the space, parameterising a manifold.

# 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 $\tilde{x}$

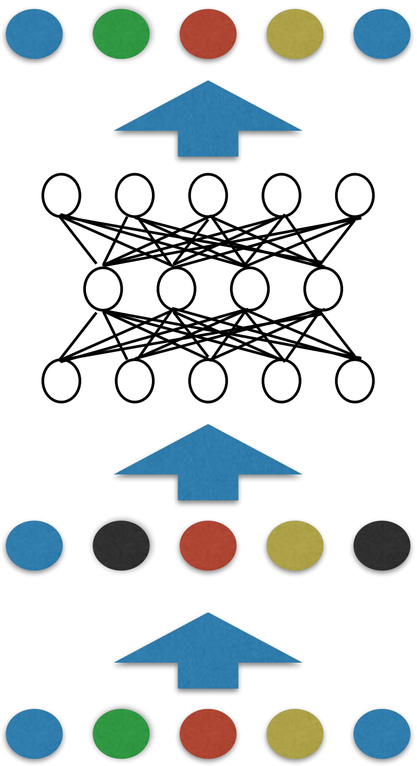
Input $x$

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

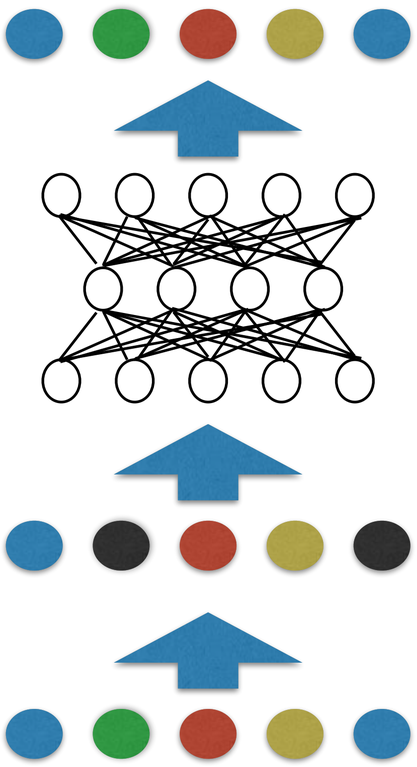
Noise: dropout input values

Input

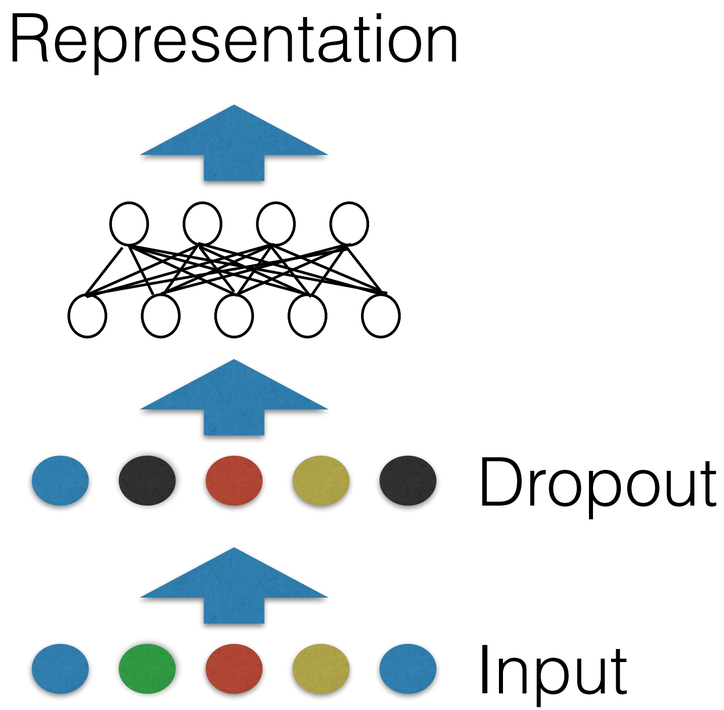
# Learning representations

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

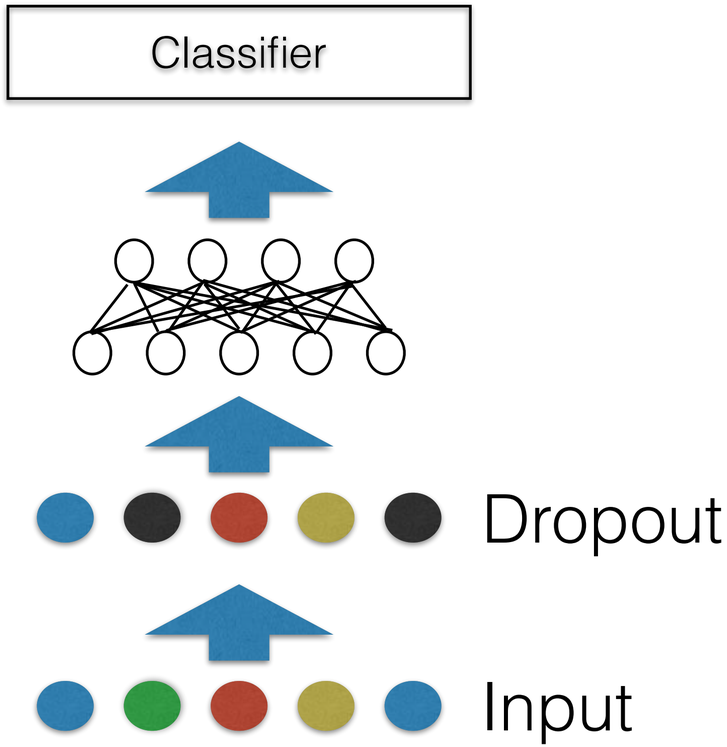
# Pretraining and transfer



# Pretraining and transfer

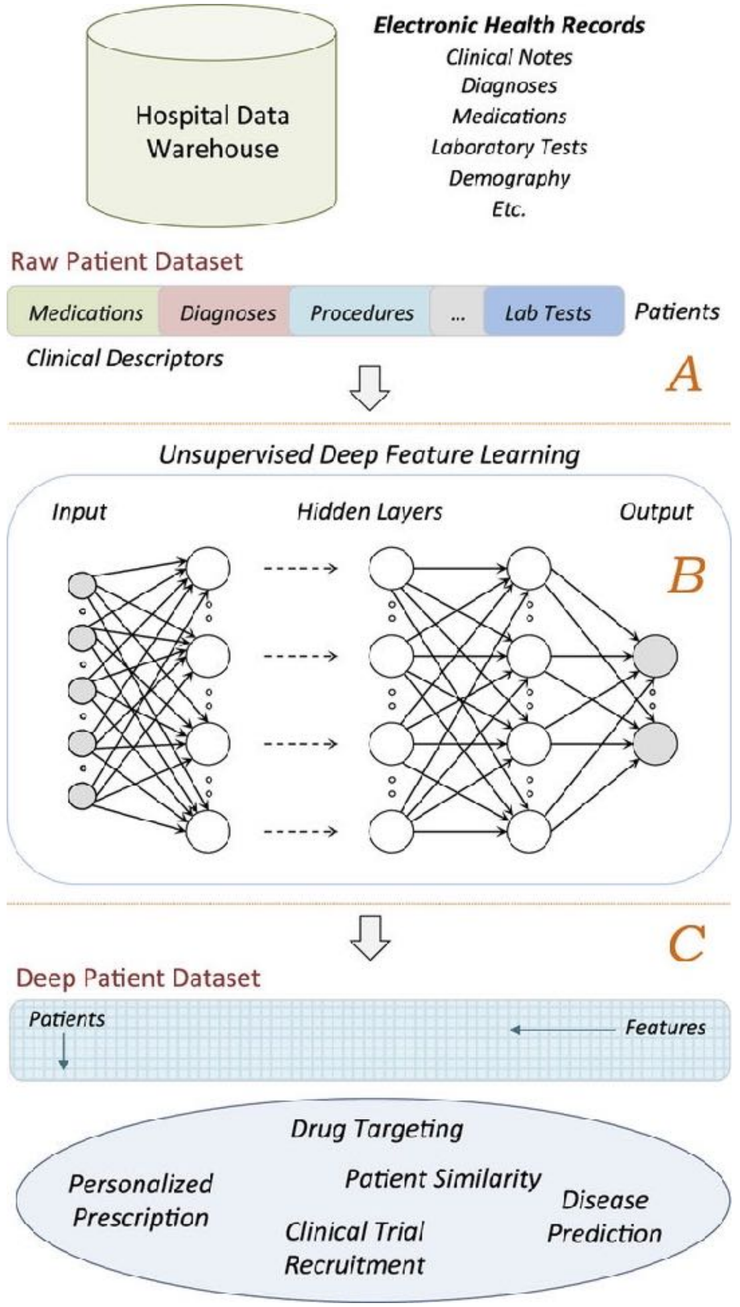


# Pretraining and transfer



# 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

# Wireless Capsule Endoscopy Diagnosis By AI

# Early detection of diseases

in this project

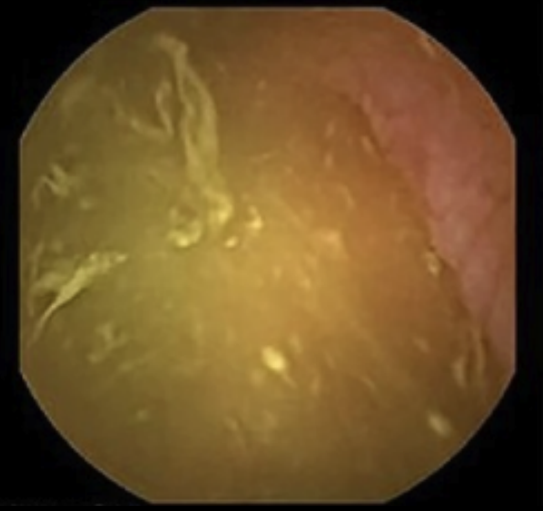
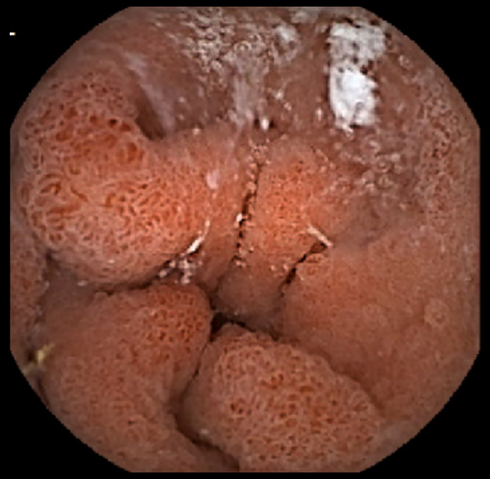
the **AI goal** is to **reduce**

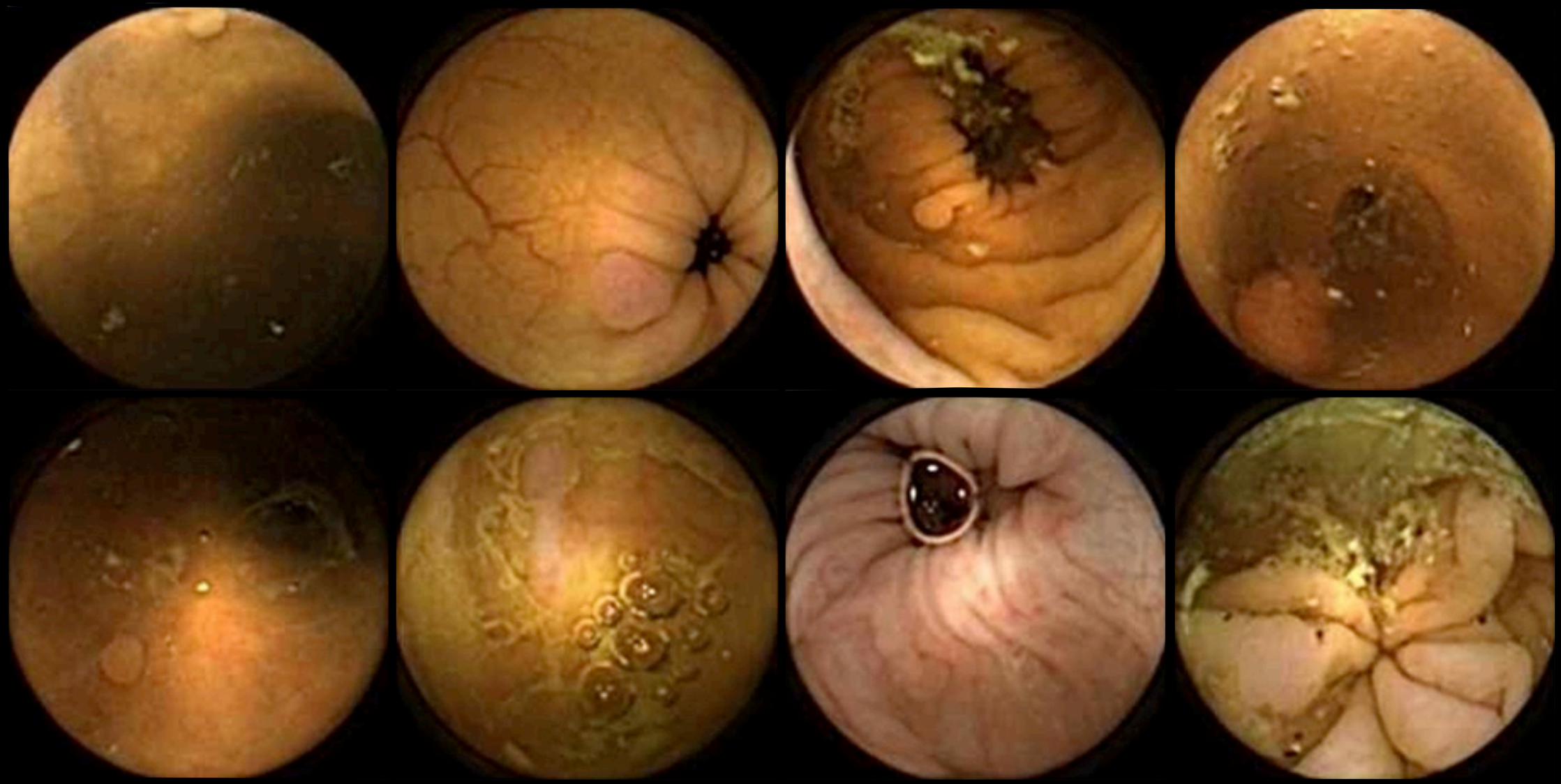
the **diagnostic time**

needed by the physicians

# AI system to detect polyps, bleeding, inflammatory lesions and intestinal content



**Deep Learning** is

an amazing tool but diagnostic of

**Medical Images** is really **challenging**

**data sets** used to be **really poor**

# Research Challenges

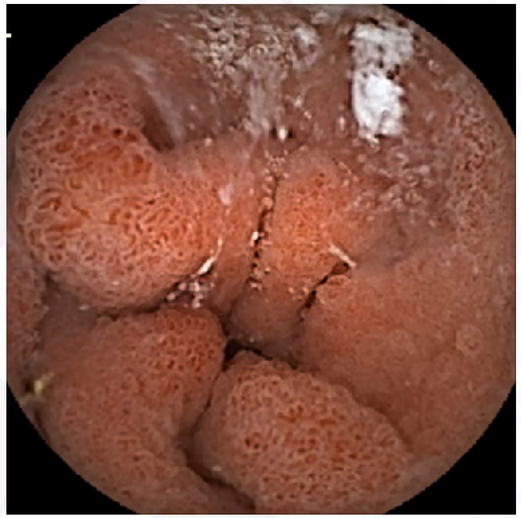
DL with **small number** of images

DL models with **uncertainty**

**Explainable** DL models

**Domain-Adaptation** method

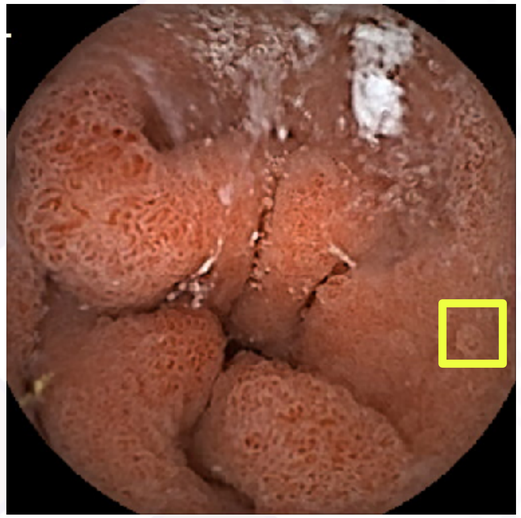
# DL models with **uncertainty**



Polyp Detection

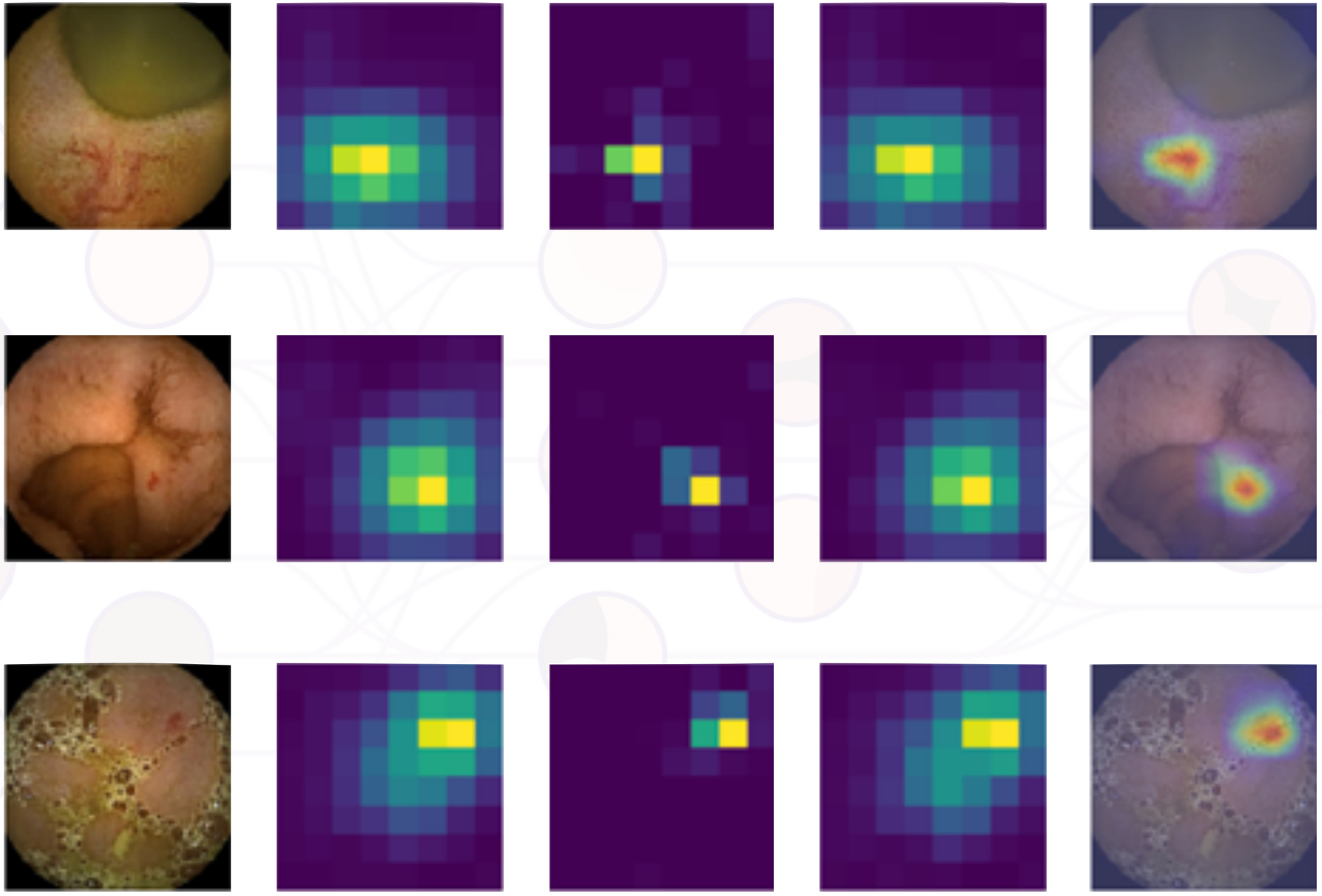
But, how confident about it?

# **Explainable** DL models



**Why** the system think that there is a **polyp**?

Which **regions** of the image makes the system believe it?



# WP2b - Polyp Detection Model

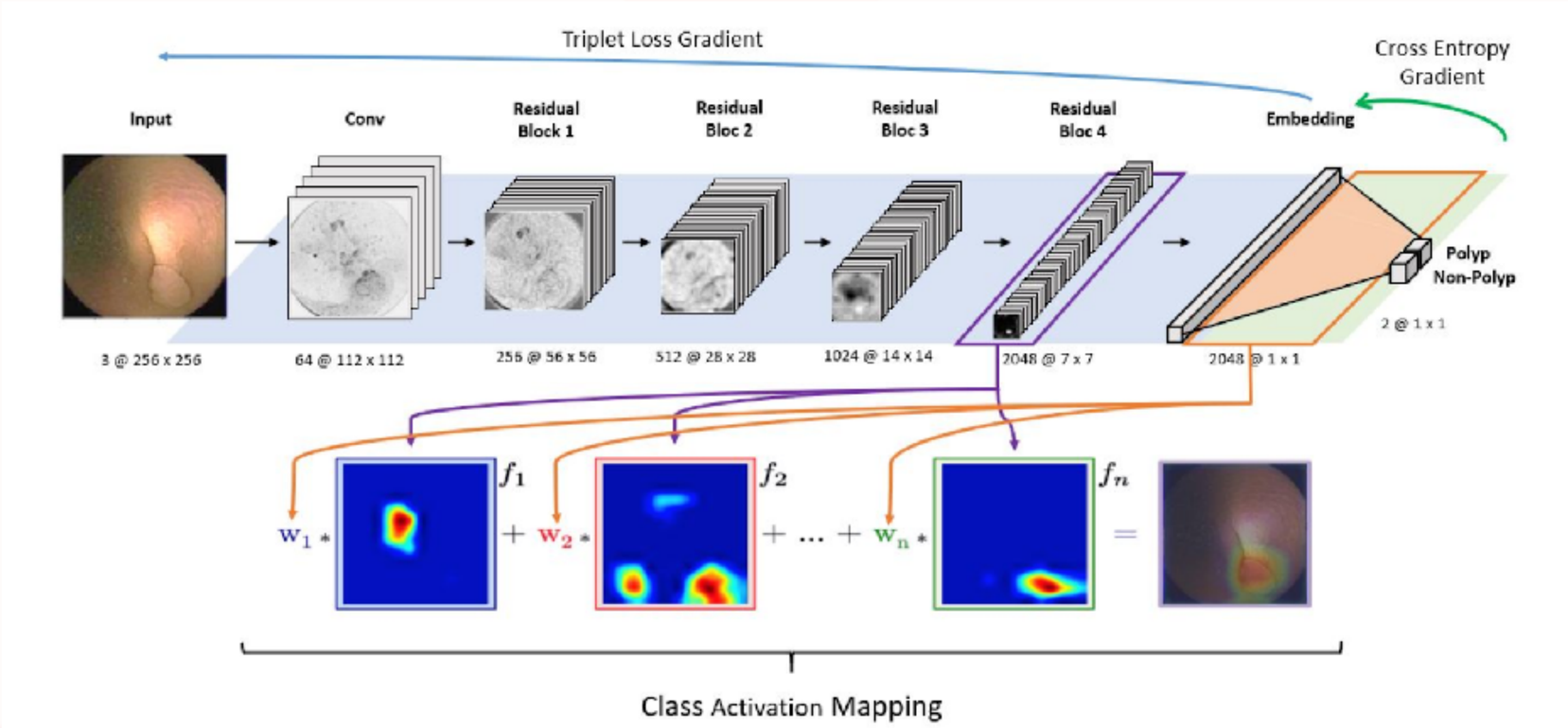
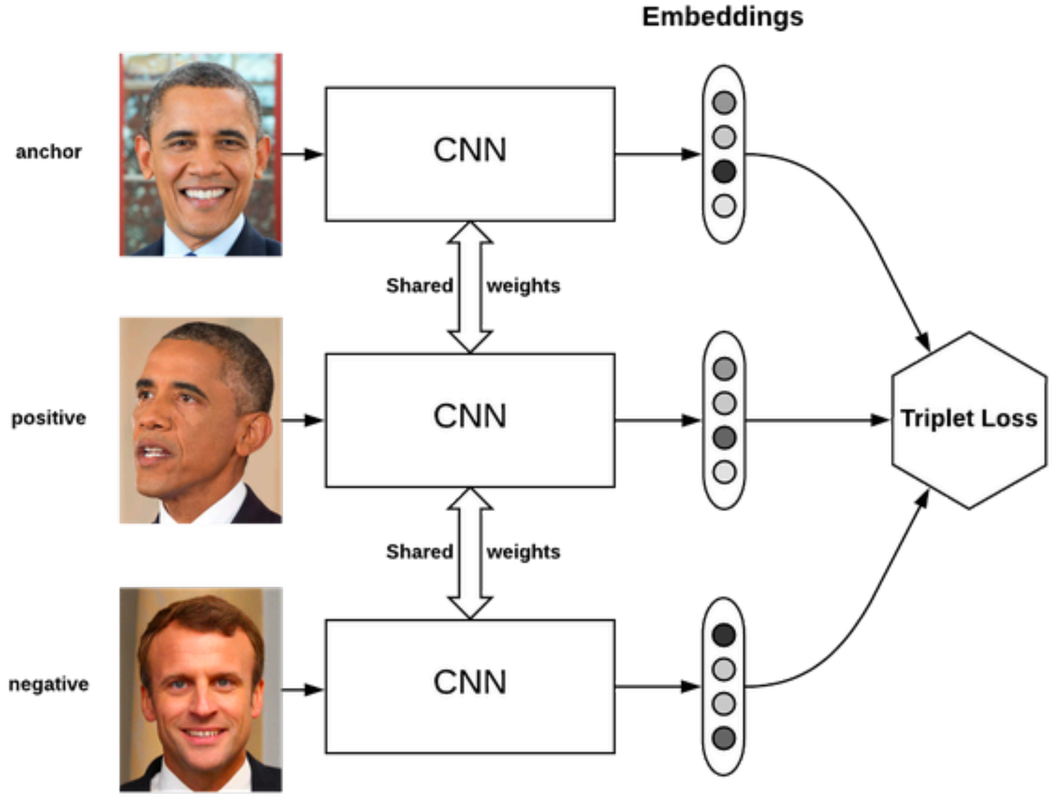



Fig. 3: Overview of the proposed CNN structure. The upper part of the scheme appear the ResNet architecture with our methodology applied in it. The background colour reflect the layers affected by each one of the gradient generated by the main losses. The lower part of the figure show how the class activation mapping is build.

# General Pipeline - 2018: Triplet loss




$$Loss = \sum_{i=1}^{N} \left[ \| f_i^a - f_i^p \|_2^2 - \| f_i^a - f_i^n \|_2^2 + \alpha \right]_+$$

# **Domain-Adaptation** method

| Capsule Endoscopes: Small Bowel, Colon, and Patency Capsule | | | |
|---|---|---|---|
| a   | b   | c  | d   |
| PillCam SB | OMOM | CapsoCam SV1 | PillCam Patency |
| e   | f   | g   | |
| MiroCam | Endocapsule | PillCam Colon | |