Deep learning for images recognition

Codewise
Krakow, Dec 2017

INTRO

fast.ai

Making neural nets uncool again

kaggle

The wonderful and terrifying implications of computers that can learn

Passenger Screening Algorithm Challenge — $1,500,000
Improve the accuracy of the Department of Homeland Security's threat recognition al... — 498 teams
Featured · 9 days to go · terrorism, image, object detection

Statoil/C-CORE Iceberg Classifier Challenge — $50,000
Ship or iceberg, can you decide from space? — 1,963 teams
Featured · 2 months to go · weather, shipping, image, binary classification

AGENDA

- What learning means
- Logistic regression
- Gradient descent
- Neural network and Deep neural network
- Convolutional neural network
- Pretrained models environment (imagenet)
- Kaggle image recognition competitions

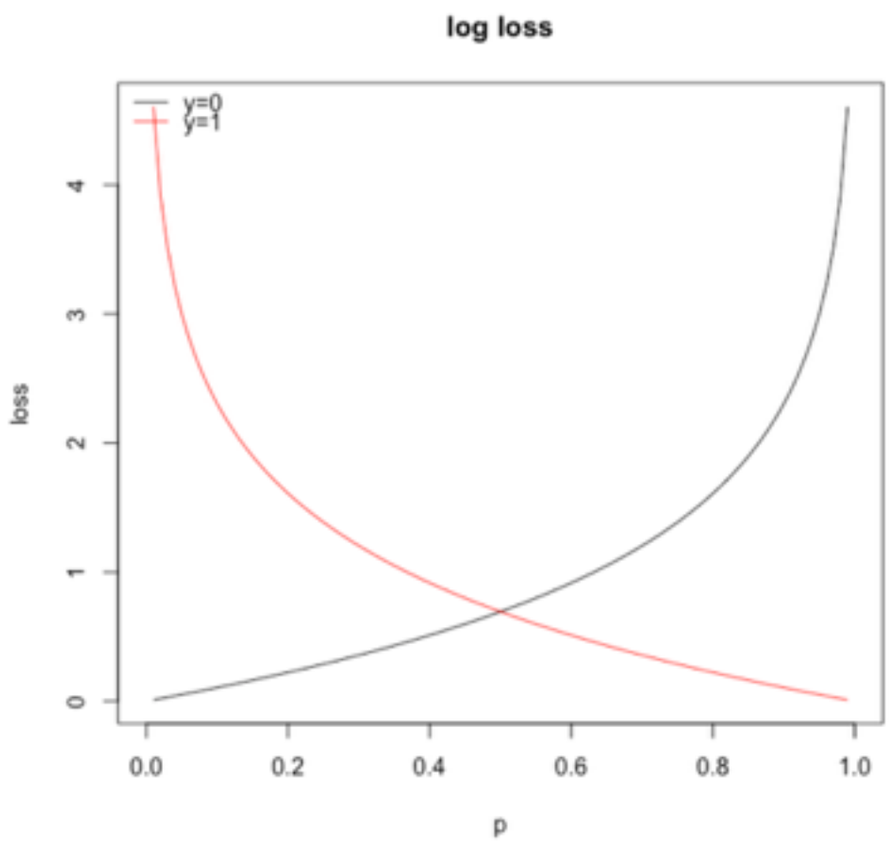
What learning means

Function ⟶ **Correct predictions**

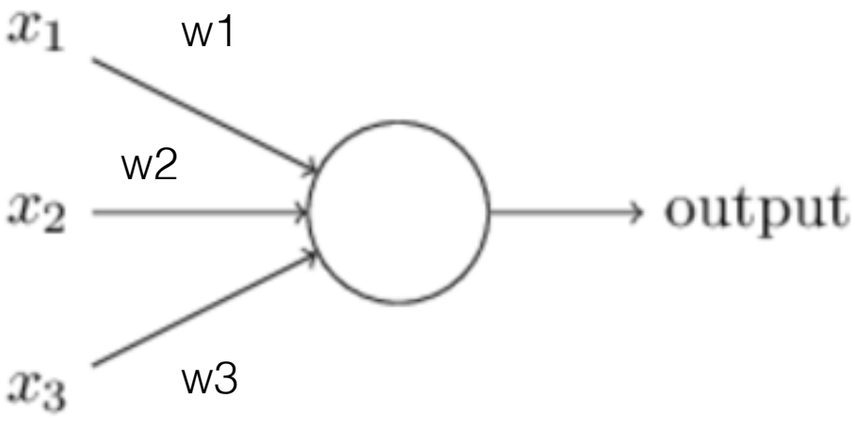
What learning means

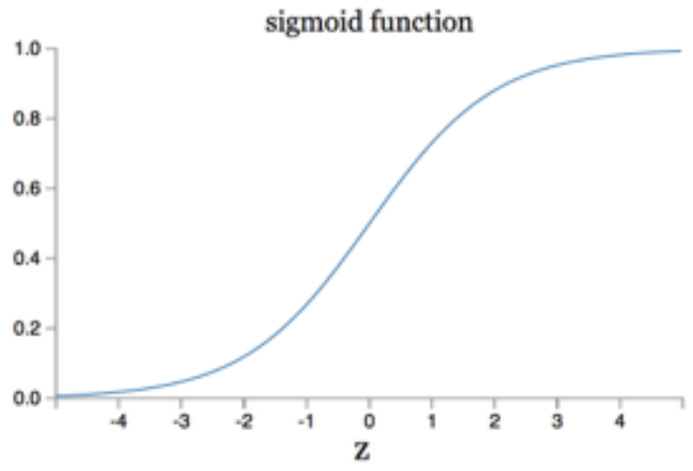
* Metrics of quality for humans:
  * accuracy
  * f1 score
  * Confusion matrix metrics
* Metrics of quality for machine - cost function.
* **The goal of learning is always to minimise cost function.**
* Typical cost function for image recognition task is Logatighmic loss:

$$\text{logloss} = -\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}[\,y_{ij}log(p_{ij}) + (1-y_{ij})\log(1-p_{ij})]$$



Logistic regression



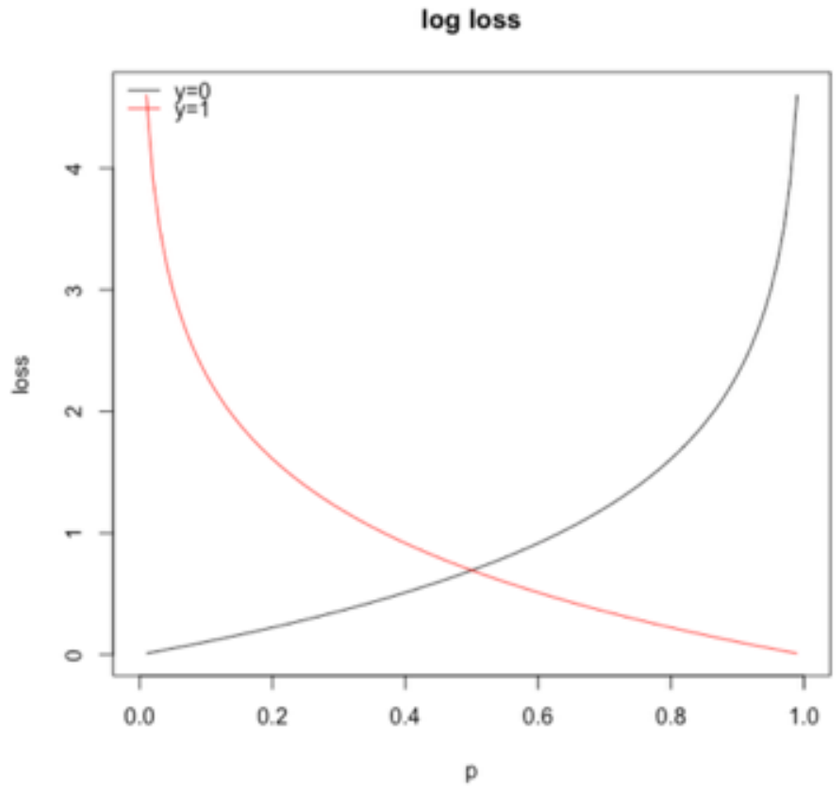


$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

* MNIST picture is 32 x 32 pixels
* After lining it up we got 1024 pixels in a row
* 60K images
* So we have a matrix of size 60000x1024

* **Our task is to find weights *W* (1024 + 1) that minimise cost function (logarithmic loss)**

Gradient descent

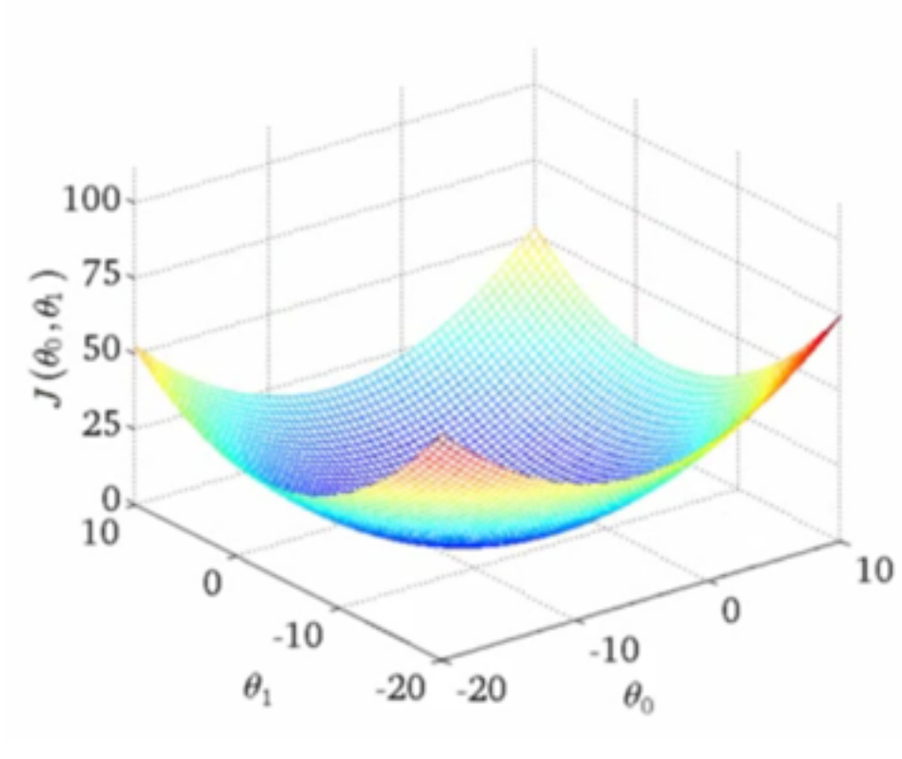


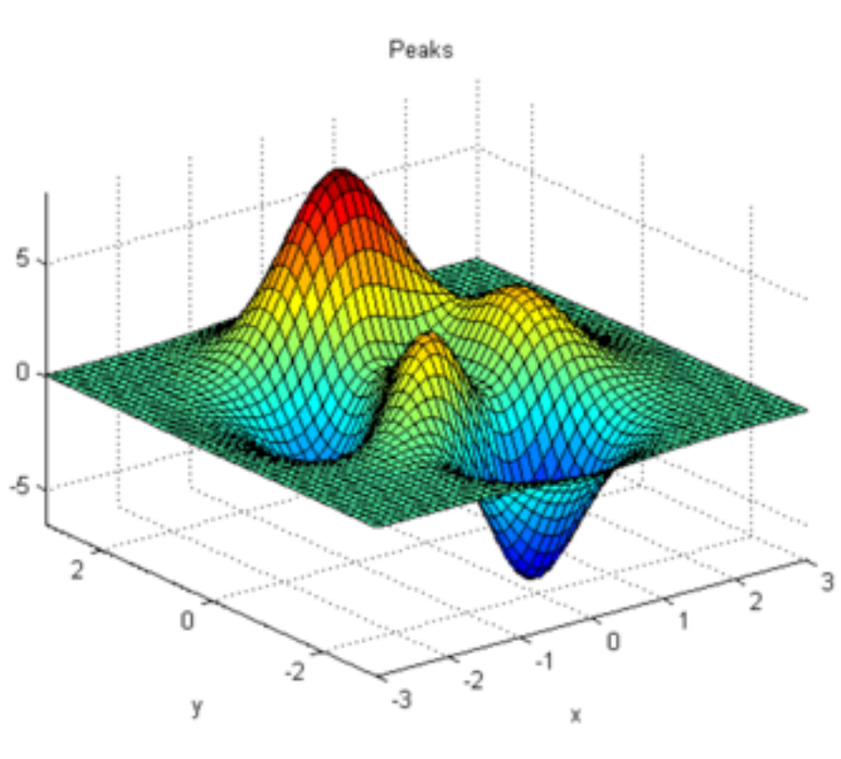
What happens to our cost function if we change ***w1*** by 0.0001 ?

decrease → Increase w1

increase → Decrease w1

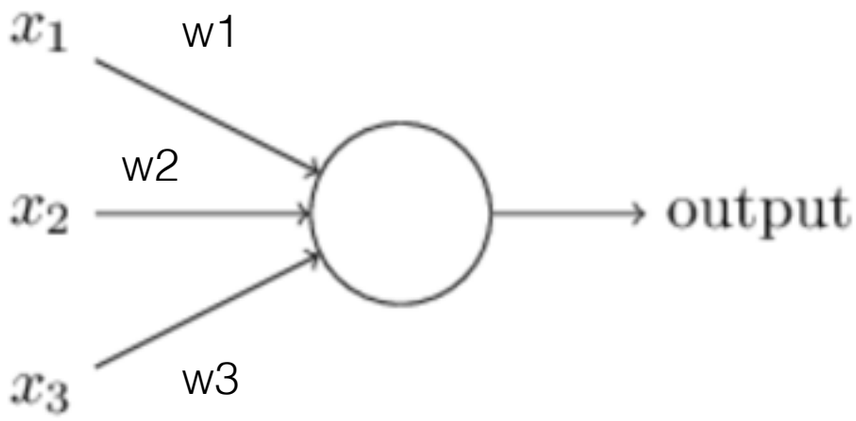
- Randomly initialise weights
- Calculate the influence of each weight (*W*) onto cost function
- Update *W* based on calculated gradients
- Iterate the process till you cant improve loss functions



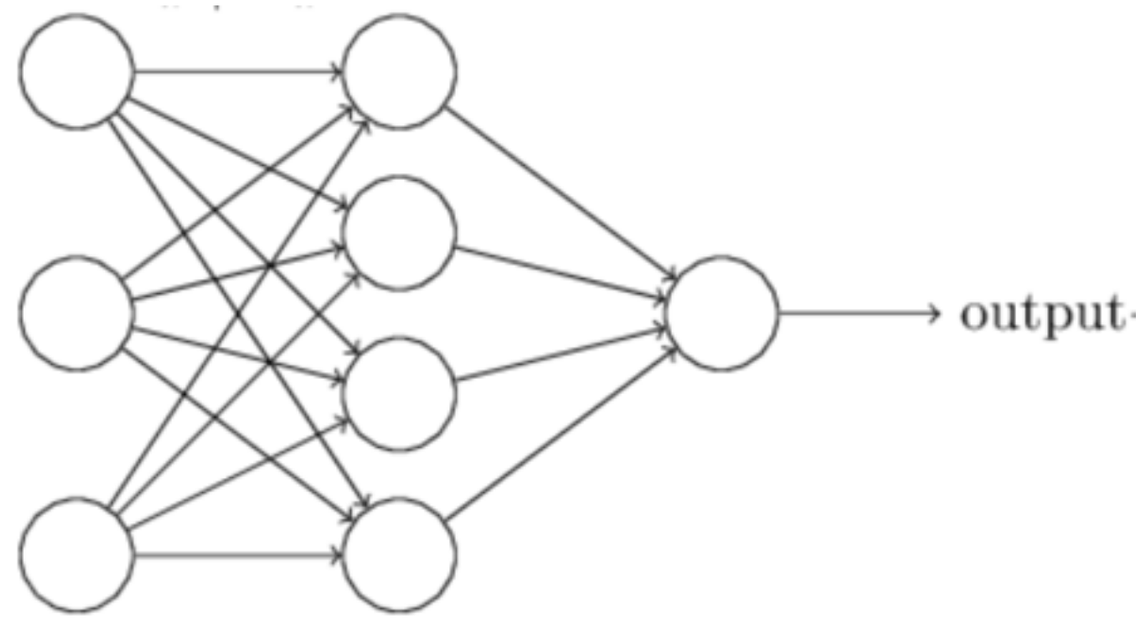


Neural network and Deep neural network

Neural network with single layer and logit activation function is Logistic regression



Neural networks with more than 1 layer (called hidden layer) is called **deep neural network**.



Training deep neural network (finding/optimising its weights) is somewhat similar to training logistic regression.

- Randomly initialise weights

Forward pass

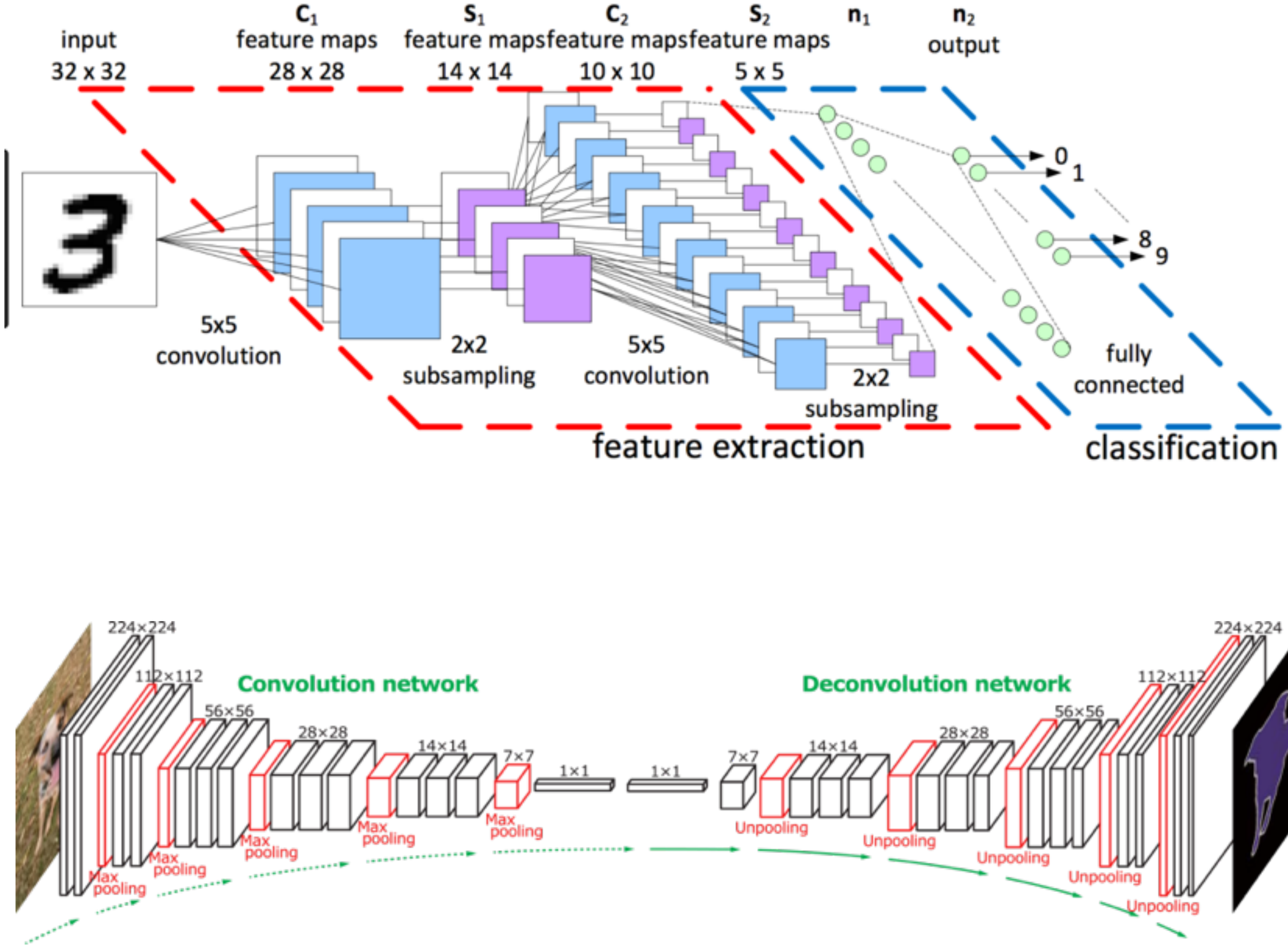
- Calculate values in all nodes
- Calculate cost function

Backward pass

- Calculate the influence of each weight (*W*) onto cost function
- Update *W* based on calculated gradients

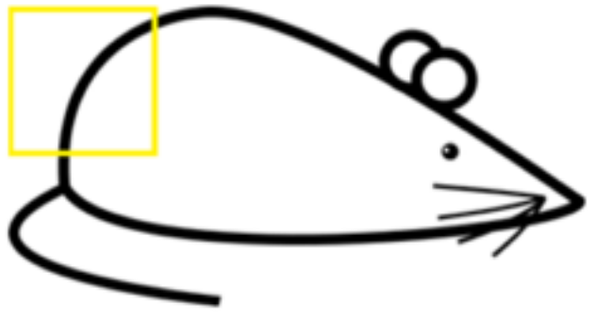
Iterate the process (forward-backward passes) till you cant improve loss functions

Convolutional neural network



Convolutional neural network - Filters

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

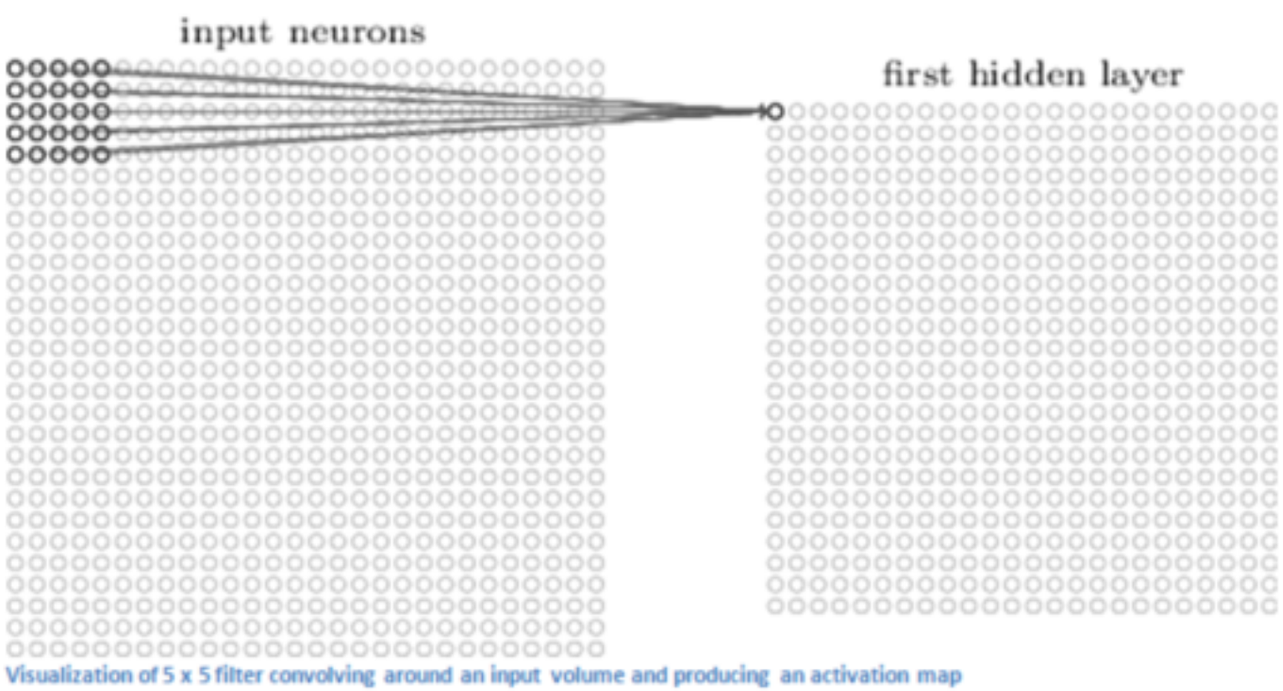
Pixel representation of the receptive field

*

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

Pixel representation of filter

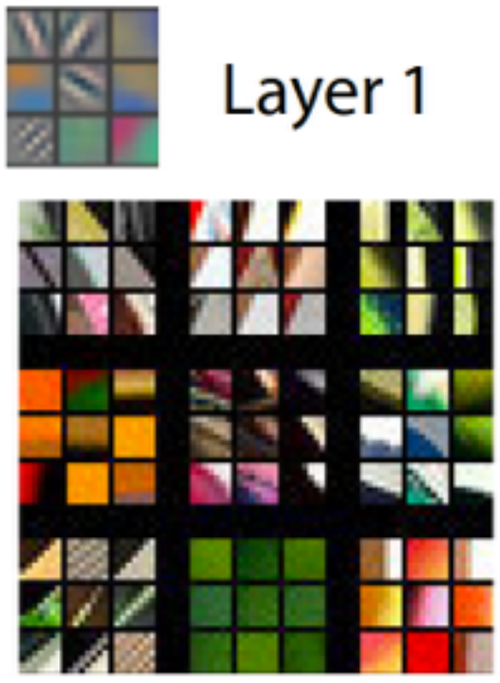
Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

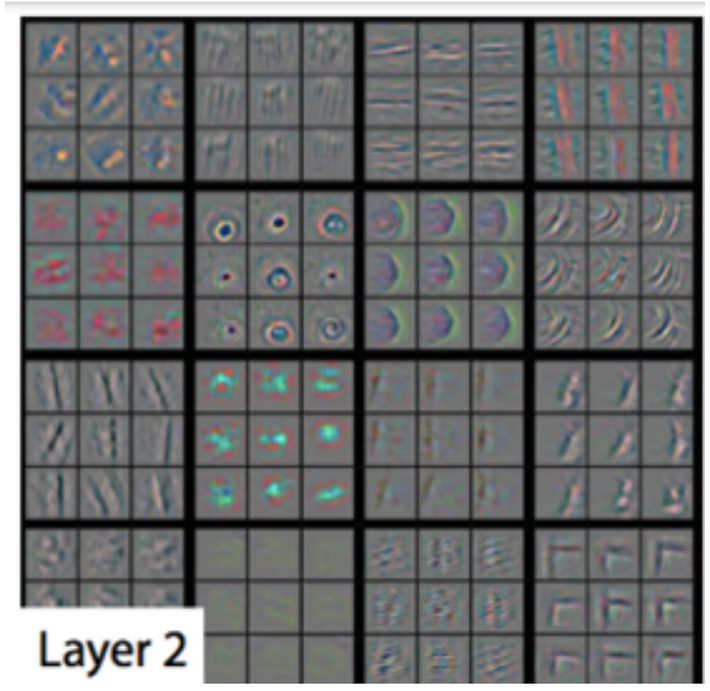


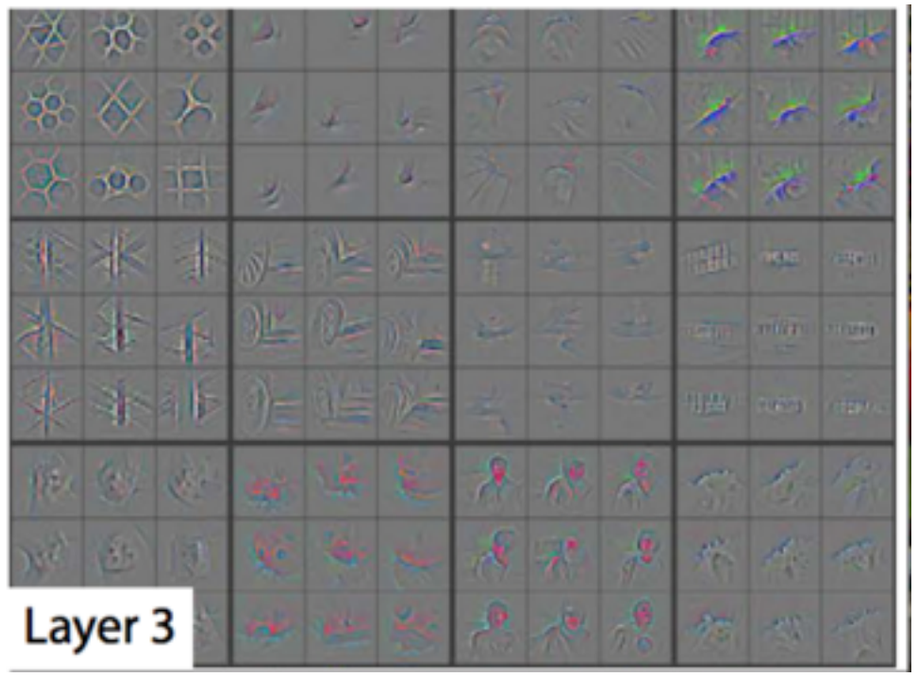
Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

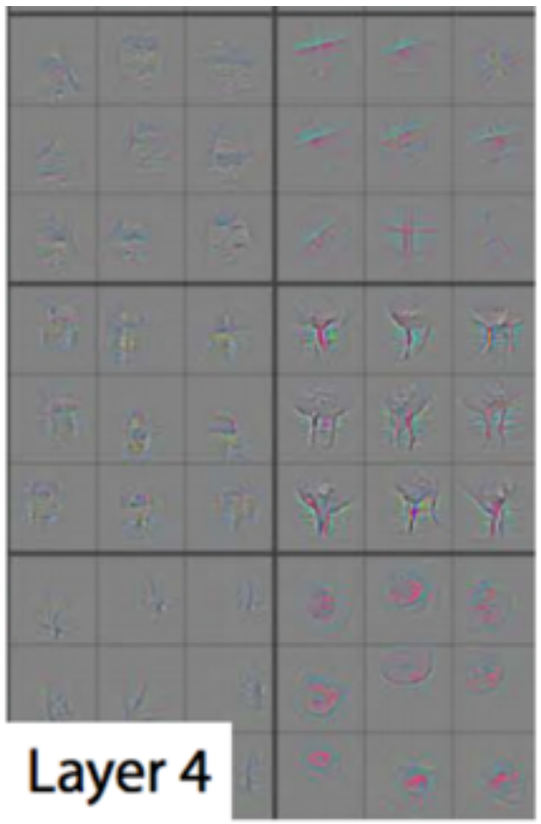
- Small matrixes (3x3, 5x5) which we use to extract features from images
- Similar to weights in Logistic regression, filters are parameters we can learn with Gradient descent

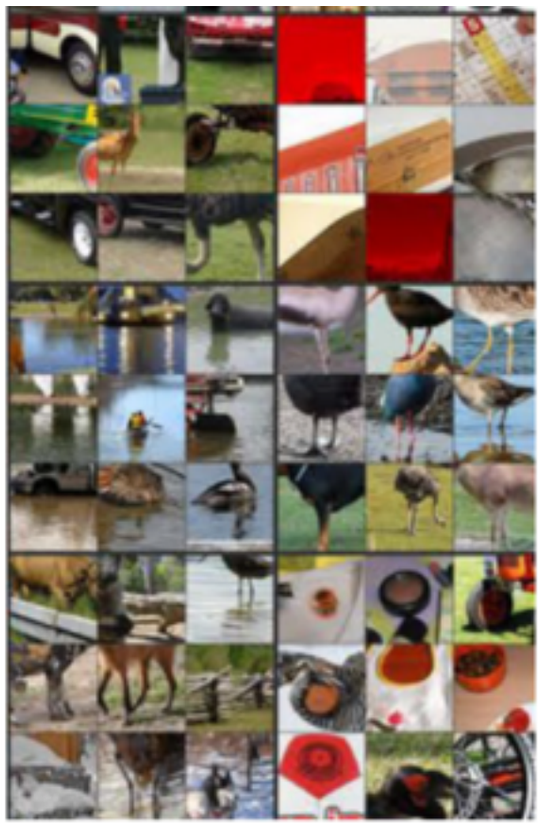
Convolutional neural network - Filters

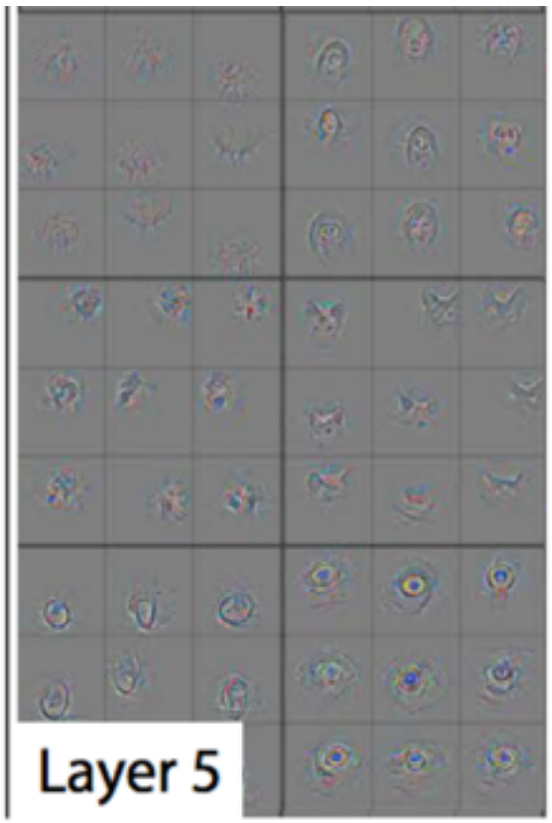




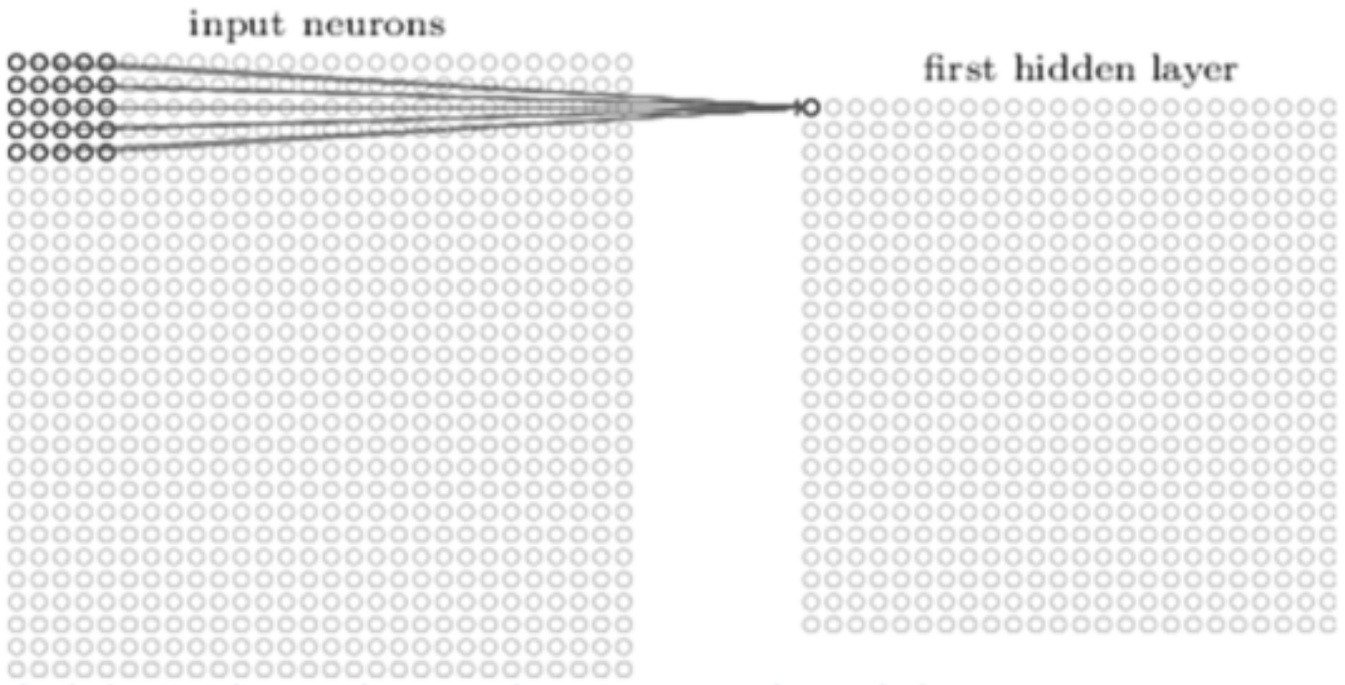






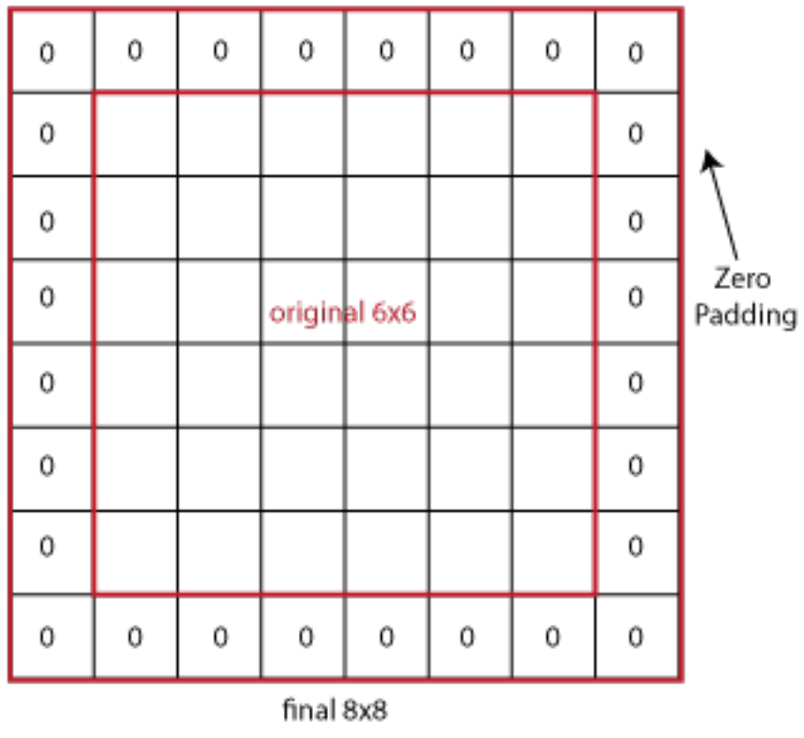


Convolutional neural network - Padding



Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

Multiplying image matrix by filter reduce the size of the original image

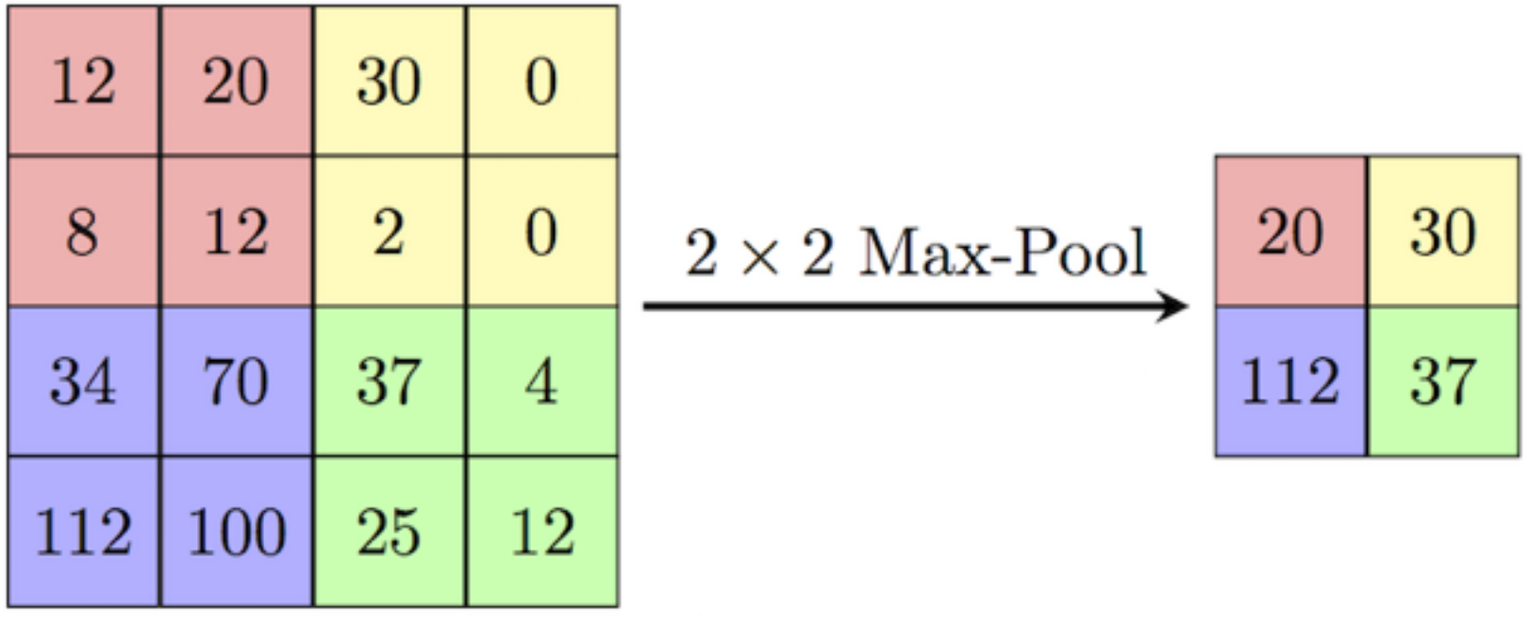


Padding prevents size decrease by adding 0 on the edges of the image.

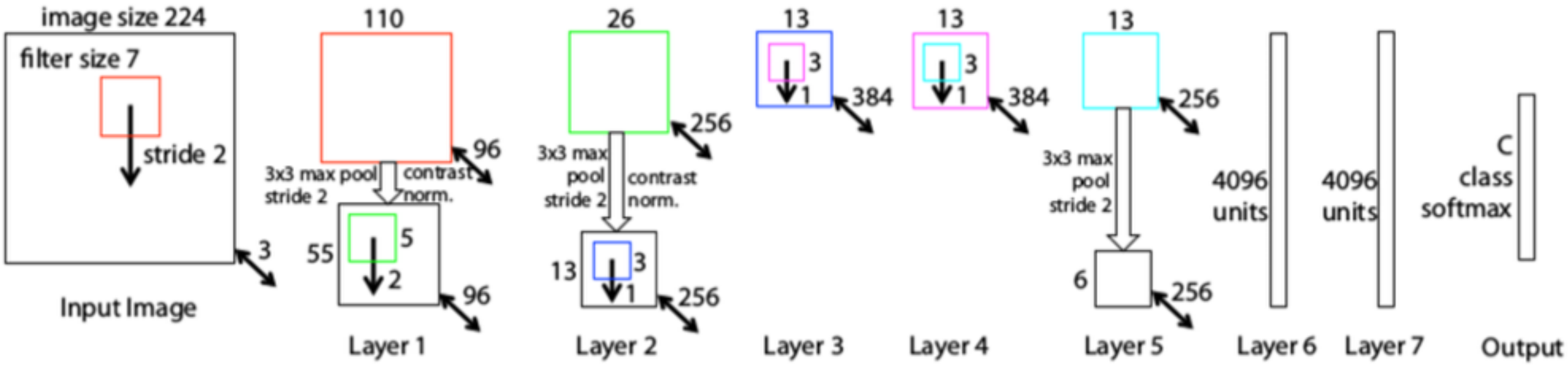
Convolutional neural network - Pooling

Reduce the size of the image by selecting max/mean values from a specified area (2x2, 3x3, etc).

Stride - is a hyperparameter that defines how many pixels to the right or down to make after each calculation. On the picure below, stride 2 is shown.



Convolutional neural network - Puting it all together



ZF Net Architecture

Image size 224x224x3 (3 colours)

96 filters of size 7x7x3 are learned

Stride 2 and filter size 7 reduce image size to 110x110xdepth

Depth is equal to number of filters in the layer (96).

Its not an image anymore. Its a 96 depth representation of filters activationgs.

After 3x3 max pooling we got 55x55x96 “image”.

256 filters of size 5x5x96 are learned

Resulting image is 26x26x256

After 3x3 max pooling we got 13x13x256

Learning 384 filters of size 3x3x256

Because resulting size is the same, that means 0 padding was applied.

Learning 256 filters of size 3x3x384

Because resulting size is the same, that means 0 padding was applied.

Fully connected layers (FC)

Just plain neural network after flattening activations from convolutional layers.

And whole this construction is still trained using gradient descent

Pretrained models environment

Imagenet competition - most prestigious competition in image recognition typically won by Google/Microsoft/Facebok or research institutes. The competition is about predicting 1000 different categories. Winners typically publish papers about their models, disclose arhitectures and sometimes even upload weights. Typically model is trained during 2-3 weeks.

Accuracy on validation set (single model)

Model	Version	Acc@1	Acc@5	
NASNet-A-Large	Tensorflow	82.693	96.163	Google
NASNet-A-Large	Our porting	82.566	96.086	Google
InceptionResNetV2	Tensorflow	80.4	95.3	Google + Microsoft
InceptionV4	Tensorflow	80.2	95.3	Google
InceptionResNetV2	Our porting	80.170	95.234	
InceptionV4	Our porting	80.062	94.926	
ResNeXt101_64x4d	Torch7	79.6	94.7	UCSD, Facebook
ResNeXt101_64x4d	Our porting	78.956	94.252	
ResNeXt101_32x4d	Torch7	78.8	94.4	
ResNet152	Pytorch	78.428	94.110	Microsoft
ResNeXt101_32x4d	Our porting	78.188	93.886	
FBResNet152	Torch7	77.84	93.84	
DenseNet161	Pytorch	77.560	93.798	Cornell and Tsinghua Univ, Facebook
FBResNet152	Our porting	77.386	93.594	
InceptionV3	Pytorch	77.294	93.454	
DenseNet201	Pytorch	77.152	93.548	

We do not need to train a model from scratch, we can take already pretrained a model and just finetune its weights.

If we need to classify similar images to IMAGENET categories, than we might need to finetune only Fully connected layers.

If we have somewhat different images, we need to finetune last convolutional layers.

If we have completely different images we might need to finetune all layers but still it is much quicker than training a model from scratch.