An Introduction to Feedforward Nets and Conv Nets

Dario Garcia-Gasulla
dario.garcia@bsc.es

Deep Learning course
Master of Artificial Intelligence
UPC - BarcelonaTECH

High
Performance
Artificial
Intelligence
group

WHAT'S THE PLAN?

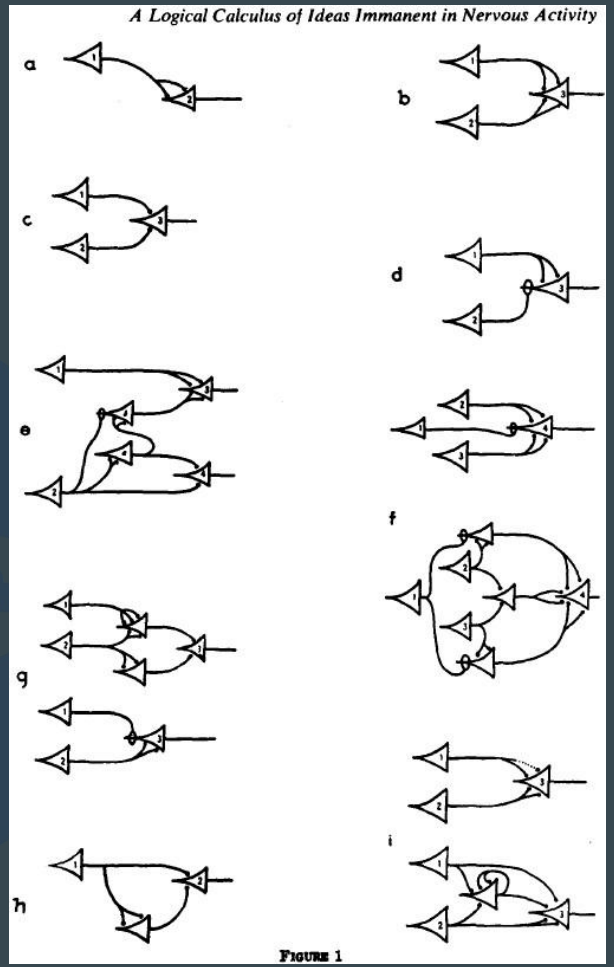
- Content
  - Review of the basic components of Artificial Neural Networks (ANN)
  - Review of the basic components of Convolutional Neural Networks (ANN)
- Goal
  - Basic understanding of the theory, so that you can perform experiments with a minimum coherence
  - To be tested tomorrow
- What is missing from this seminar
  - Gory and mathy details (because who cares why it works if it works)
    - For those interested, references are provided. Beyond those, the Internet is full of info
  - Introduction to machine learning (e.g., evaluation of models, data pre-processing, etc.)
    - Ping us if you need help with this, or try the mailing list *ai@bsc.es*

ANCIENT HISTORY 1/3

McCullock & Pitts, 1943 [1]

- Using neurons as the basic computational unit
- Each neuron has a number of inputs and one output
- Synapses (output-input links) have an associated weight characterizing the **influence**
- The output of a neuron with *n* inputs is:

$$w^1*x^1 + w^2*x^2 + \ldots + w^n*x^n$$



ANCIENT HISTORY 2/3

Hebb, 1949 [48] :

- *“Neurons that wire together fire together”*
- Adaptative (learnable) weights, to match the desired output of the neuron

ANCIENT HISTORY 3/3

Frank Rosenblatt's **Perceptron,** 1958 [3]

- Binary classifier (cat or dog?)
- Real valued weights
- Output of neurons are binary:

$$f(x) = \begin{cases} 1 \ \textit{if} \ w \cdot x + b > 0 \\ 0 \ \textit{otherwise} \end{cases}$$

- b is real scalar constant (bias)

Rosenblatt built the Mark I Perceptron [3,4,9]:

- 400 photosensitive sensors (input)
- 512 stepping motors (neurons)
- 8 response units (output)

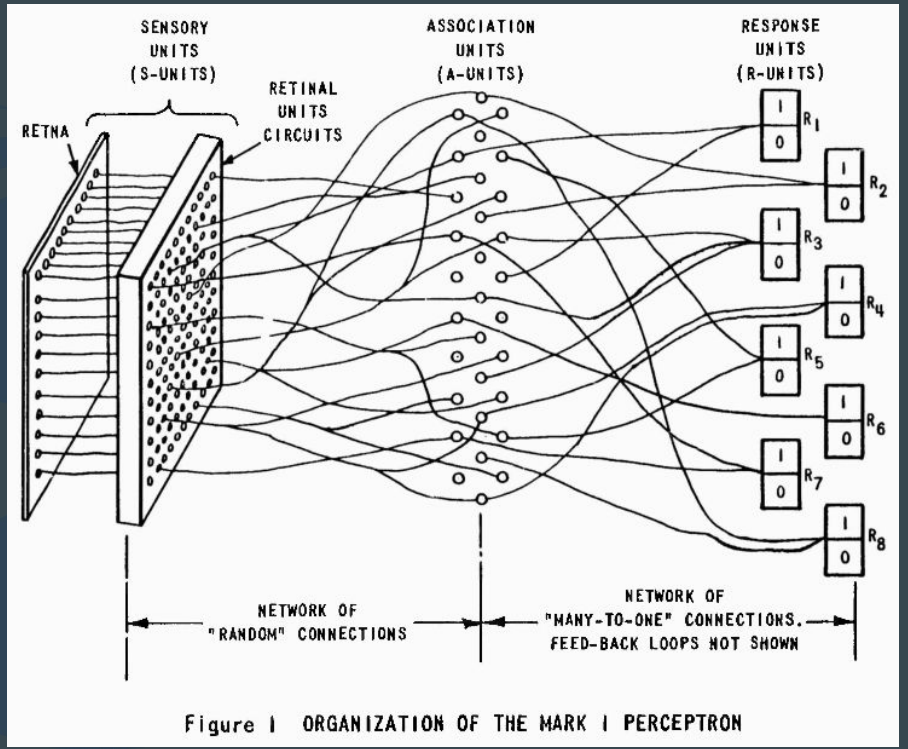
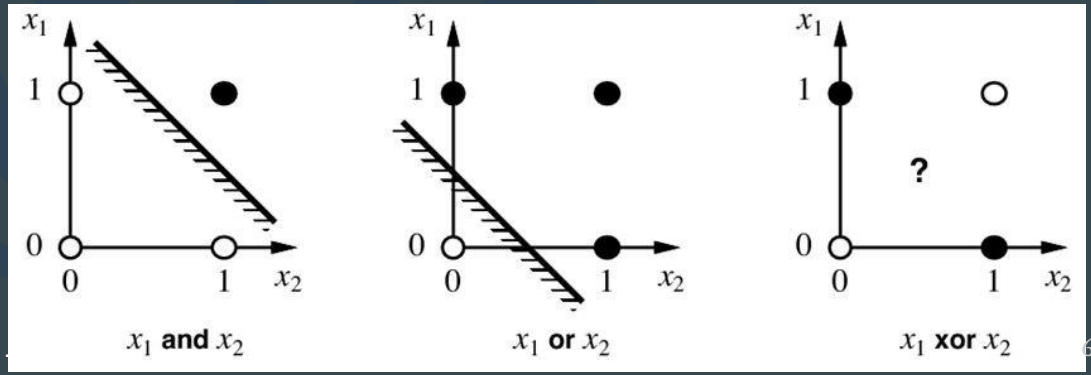


Figure 1 ORGANIZATION OF THE MARK I PERCEPTRON

WINTER CAME

- The Perceptron caused the first hype (not the last one) on neural networks
- Rosenblatt acknowledged certain **limitations**. Those were further explored in Minsky and Papert, *"Perceptrons: an introduction to computational geometry"*, 1969 [6]
- Main limitation:
  - With one layer, we can only solve linear problems. We can't solve the XOR!
  - With more layers, how can we learn the weights of neurons not next to the output?

- This "killed" NN
  - **AI Winter** (not the last one)
  - No funding
  - Little research for years



BACKPROPAGATION 1/2

- The **backpropagation algorithm** is based on the chain rule, to compute the gradients for various layers. It enables training NN with several layers
- Webos, 1974 [7] was the first to consider its application for training NNs. Rediscovered by Rumelhart, Hinton and Williams, 1985 [8], finished the “AI Winter” on ANN.
  - LeCun used it for digit recognition for the US Postal Service in 1989 [11]
  - Included convolution, previously proposed by Fukushima in 1980 [12].

- Given the error made by the output of the network, the backprop algorithm allows to compute the corresponding error caused by every neuron in the network
- Done through the derivative of the loss with respect to each weight (turns out this is a **matrix multiplication**)

BACKPROPAGATION 2/2

To train a neural network with backprop:

- Pass an example through the network, input to output (**forward pass**)
- Compute the error made by the network in its prediction (**loss function**)
- Backpropagate the error, adjust all weights to minimize the error (**backward pass**)

Repeat...

See [10] for a mathematical approach to backprop. [42] for a simpler visual explanation. [43] shows its impact on examples, [47] explains it step by step with the derivatives.

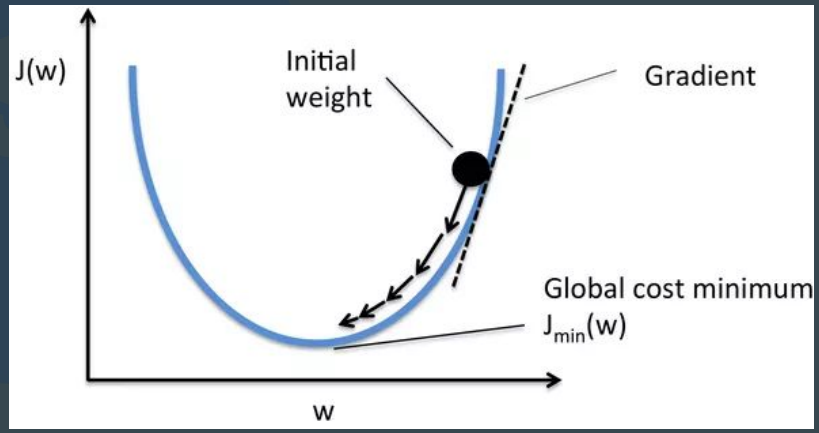
STOCHASTIC GRADIENT DESCENT (SGD)

Backpropagation allows us to determine the responsibility of each neuron with respect to the output error. How do we modify weights according to that responsibility?

- Compute the direction the weights must go to minimize error (gradient)
- Take a step in that direction (descent)
- Using only part of the input data (stochastic)

Many steps needed to reach a global minimum

Each step may takes us in a different direction



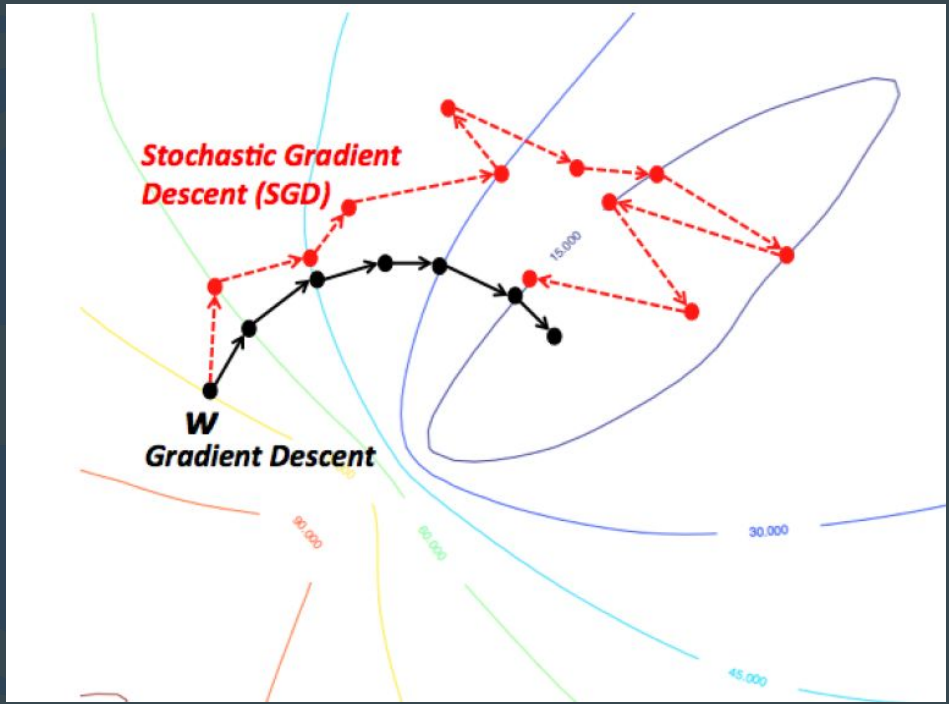
WHY STOCHASTIC?

There is a set of input data instances

We want the NN to learn a function to, given an instance, produce an output.

Why not learn using the dataset as a whole?

- Stochasticity leads to more stable convergence [citation needed]
- Stochasticity is scalable

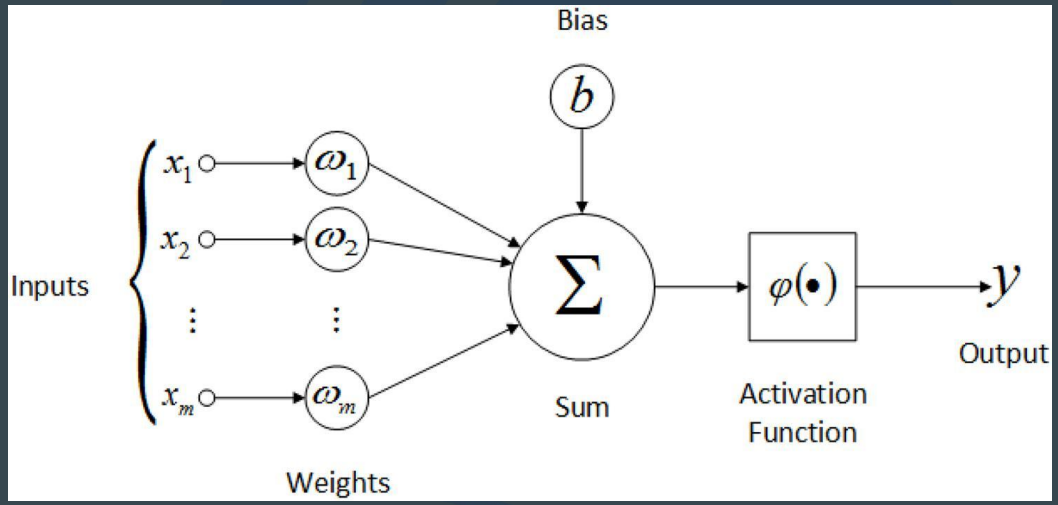


LET'S TAKE A SEC

- So we can train NN of several layers with Backprop. And SGD What about...
  - Neurons
  - Activation functions
  - Loss function
  - Hyperparameters
  - Learning algorithms
  - Overfitting issues

THE (NOT SO) MODERN NEURON

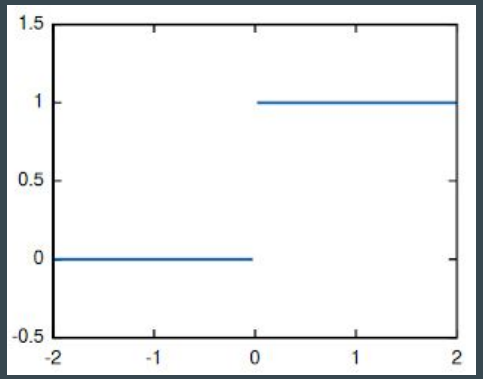
- Inputs are multiplied by the weights (**matrix multiplication!**) and summed.
- The result is passed through a non-linear **activation function**.
- Added **bias**: Provides independent term to the function. Richer learning capability



THE ACTIVATION FUNCTION

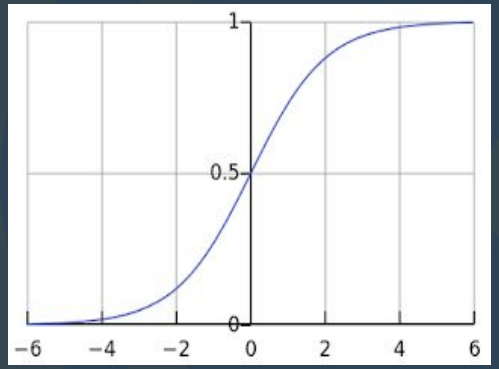
- How should a neuron fire as a response to its input?

Step function



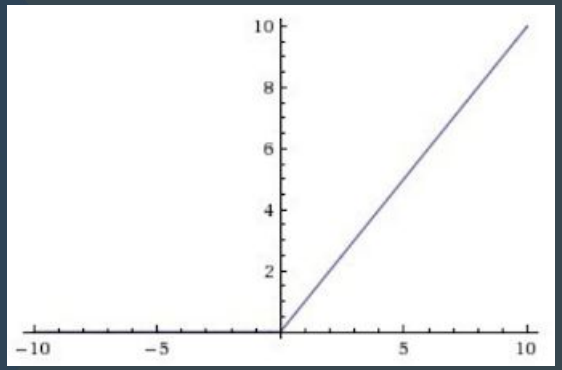
$$f(x) = \begin{cases} 1 \ \ if \ w \cdot x + b > 0 \\ 0 \ otherwise \end{cases}$$

Sigmoid



$$f(x) = \frac{1}{1+e^{-x}}$$

Rectified Linear Unit (ReLU)



$$f(x) = max(0, x)$$

ReLUs avoid the vanishing gradient problem [20]. See [21] for more on activations functions.

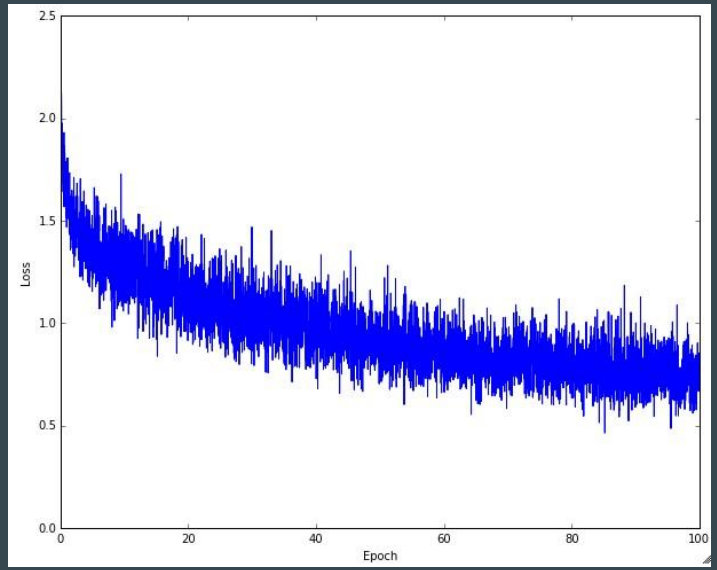
THE LOSS FUNCTION

- How do we compute the error made by the network?
  - Depends on what you are trying to do
- Classification
  - Last layer has as many neurons as classes
  - Add a softmax on top (from values to probabilities)
  - Typical error measure: Categorical cross entropy
- Regression
  - Last layer has a single neuron
  - Typical error measure: Mean squared error
- ...

HYPERPARAMETERS 1/2

- **Number of layers? Number of neurons?**
- **Weight initialization**: Starting with all weights at zero *does not work*
- **Batch size**: How many data instances we use on each training phase
  - Small batch size leads to noisy convergence
  - Large batch train faster but may be less precise [23]
  - Most common: 16, 32 and 64
- **Num. Epochs**: How many times we use the whole dataset for training

See [22] for tips on hyperparameter optimizatIon



General tips:

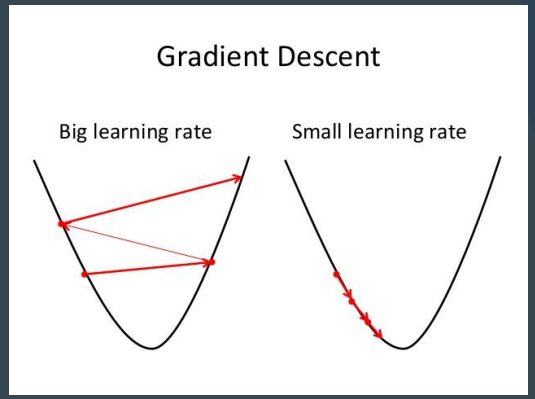
- Begin small in layers and neurons
- Always train for too many epochs

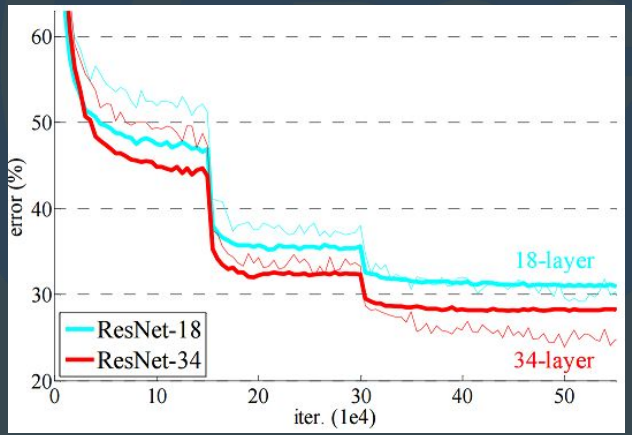
HYPERPARAMETERS 2/2

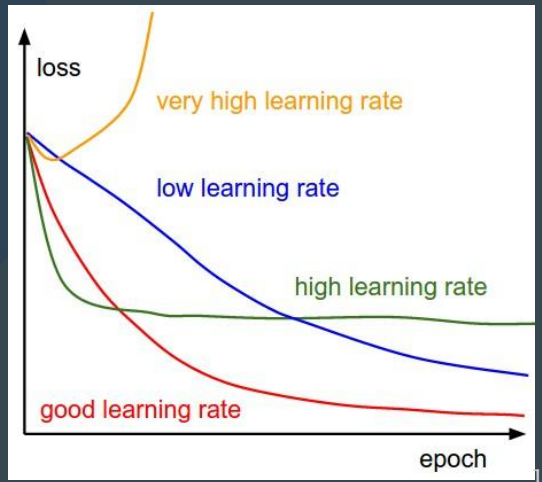
General tips:

- Start with a small LR

- **Learning rate**: How much we advance towards the gradient?
  - Large LR (e.g., full derivative) causes over correction, lack of convergence
  - Small LR takes forever, may not escape local minima
  - Usually best to start large and decrease over time (variable initial LR and decay!)
- **Momentum**: Include the gradient of the previous training phase in the current one

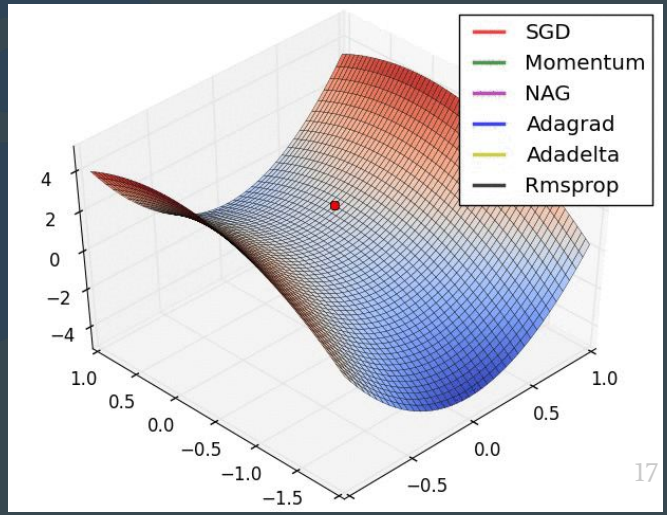






ADAPTATIVE LEARNING ALGORITHMS (Beyond SGD)

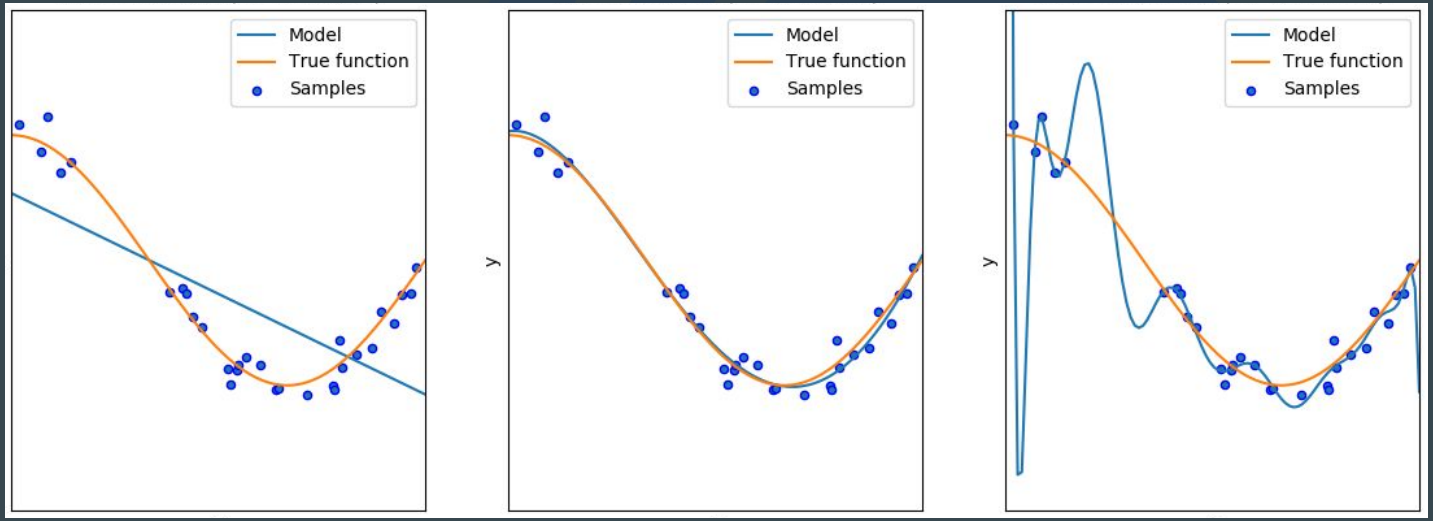
- Lets improve SGD by adding momentum
  - SGD with **Momentum** (add a fraction of the previous gradient to the current one)
  - **Nesterov** (correct momentum based on an approximation of future gradients)
- Lets get rid of the learning rate parameter
  - **Adagrad** (adapt learning rate per parameter, i.e., learnable weight)
  - **Adadelta** (smoothed Adagrad)
  - **RMSprop** (smoothed Adagrad, unpublished)
  - **Adam** (RMSprop with momentum)
- General tips:
  - Don't use vanilla SGD
  - Nesterov and Adam are often among the most competitive



See [41,46] for more details on learning algorithms

DEEP NEURAL NETWORKS: COMPLEXITY vs OVERFIT

- Within a deep NN there are lots parameters
  - 2 consecutive layers of 4,096 neurons -> 16M parameters!
- Instead of learning common patterns, the network can memorize the input: **Overfit**



General tips:

- Try to overfit first. It's a sign of health.

REGULARIZATION

- With enough epochs most NN will eventually overfit. We can mitigate it by...
- Adding noise
  - L1/L2 regularization: Penalize extremely large weight values
  - Dropout [27]: Random connectivity
  - Weight decay: Reduce weight value by a given factor after each training pass
  - Batch Normalization [39]: Normalize input by mean and std. deviation
  - Layer Normalization [44]: Batch normalization adapted for RNNs and FC layers
  - Max Out: Keep only maximum value
- Augmenting our training data
  - Rotate
  - Scale
  - Crop
  - Mirror
  - ...

See [24] for more details on regularization methods

SO FAR...

- Given an input dataset composed by I instances and V variables...
  - Split the data into training, test and validation (if you don't use validation set, don't tell anyone)
- Define a number of layers, and a number of neurons per layer
- Define batch size, epochs, learning alg., learning rate, momentum, regularization, etc...
- Launch training
  - Wait for (hopefully) a few hours
  - Look at loss curve
  - Look at accuracy curve
- Change one hyperparameter. Repeat. Repeat. Repeat.

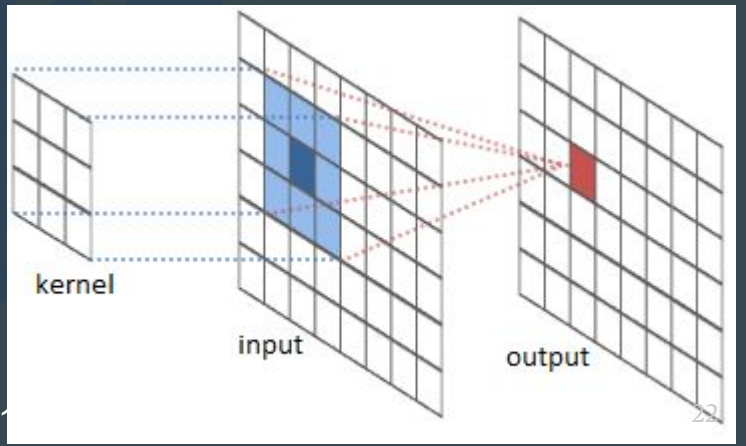
LIMITS OF FC NETWORKS

Fully connected (FC) network aka Multilayer Perceptron (MLP) aka Feedforward Neural Network (FNN)

- All neurons of a layer are connected to all neurons from previous & following layer
  - Large number of parameters
  - Assumes nothing on the input variables
- What if...
  - Input variables define a 2D matrix? E.g., Images (Convolutional Neural Networks)
  - Input variables define a sequence? E.g., Text, sound (Recurrent Neural Networks)
  - We don't have a goal? (Autoencoders)
- FNN may not be efficient...

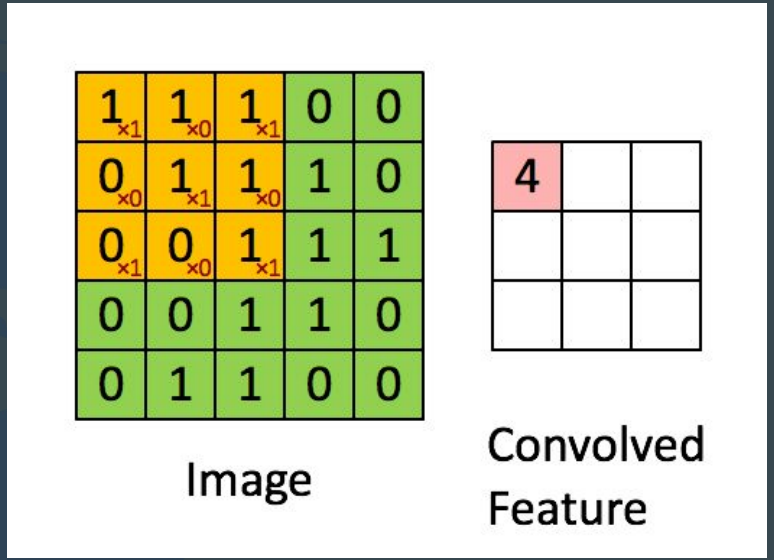
CONVOLUTIONAL NEURAL NETWORKS

- Given a 2D input (e.g., an image), every neuron within the first layer of a FNN has as input every pixel in the image.
- That's not how mammal eyes work. Instead...
  - Low level neurons focus on small and consecutive patches of the input
  - Later neurons aggregate the output of low level neurons to generate larger scale perception
  - Eventually, there is a set of neurons that perceive the whole image
- Convolutional neuron
  - Assumes spatial relation among the inputs
  - Input is a patch (e.g., 3x3)
  - Learn a kernel of 3x3 weights (so few!!)
  - Produce a single output



CONVOLUTIONAL NEURAL NETWORKS

- The filter learnt by a neuron for the patch (0,0) (3,3) of the input, is relevant for all (x,y) (x+3,y+3) patches
  - Weight sharing!
  - Train a single filter, reuse it (i.e., convolve it) throughout the input
- Convolutional neurons drastically reduce the number of parameters w.r.t. Fully connected neurons because:
  - Limited input size
  - Weight sharing

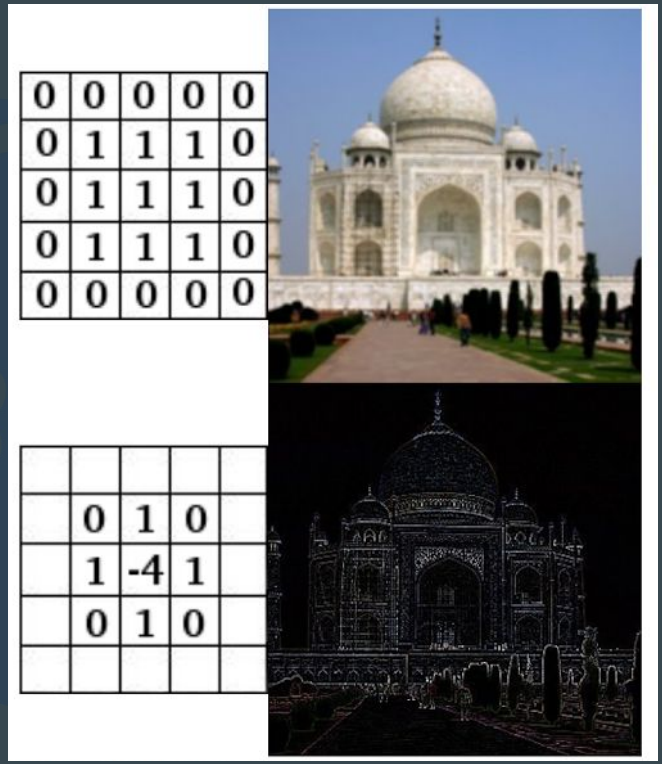


EFFECT OF FILTERS

- Different filters have different effects

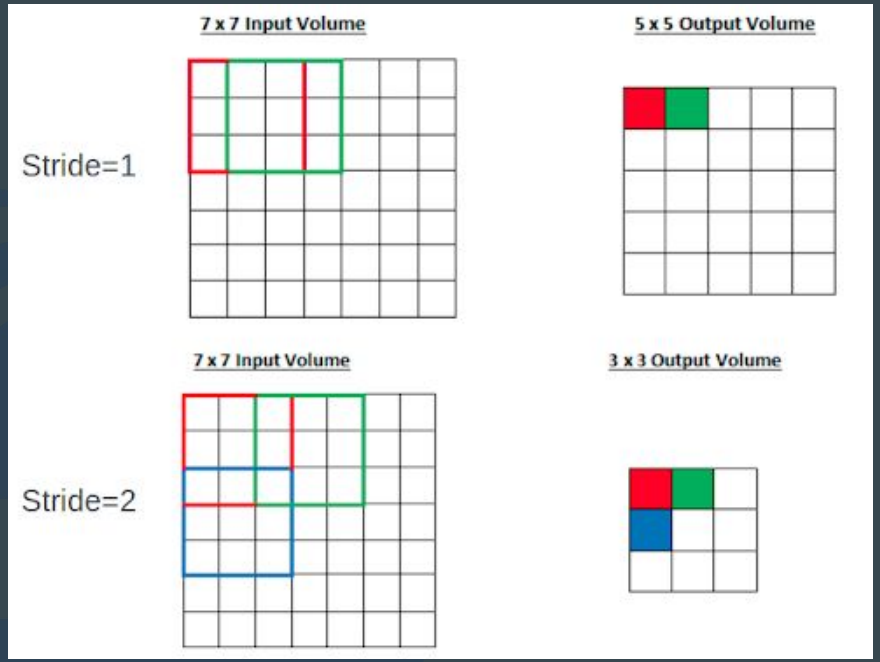
  Blurring an image

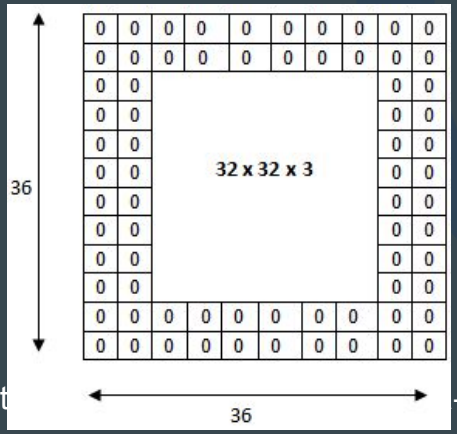
  Finding the edges



CNN PARAMETERS

- Kernel size (3x3? 5x5? 7x7?)
- Stride: Number of steps for convolving
- Padding: What to do with borders
  - Zero padding
  - Necessary to make the input and output volume equal in size



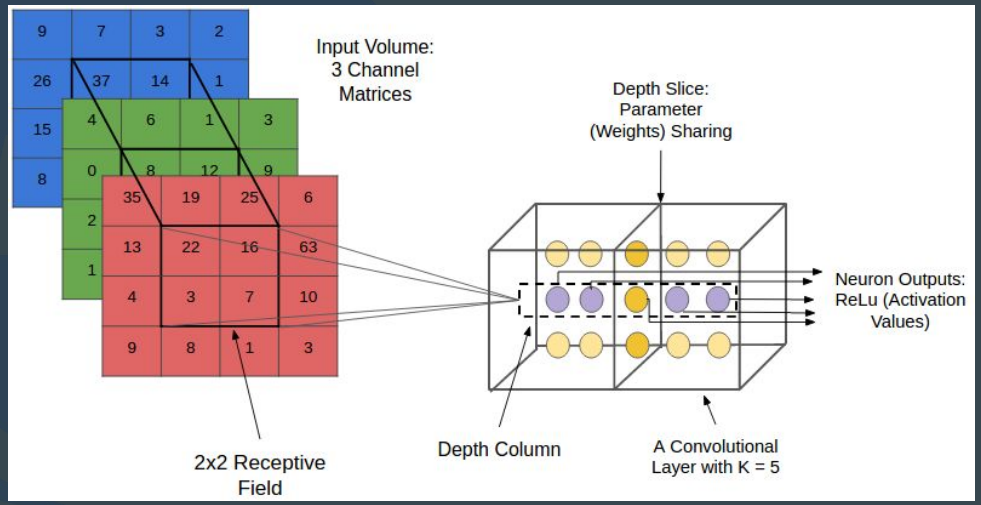


$$OutputSize = \frac{InputSize - KernelSize + 2 * Padding}{Stride} + 1$$

CNN VOLUMES

- It may be that the 2D input is in fact a 2D volume with several channels (e.g., RGB)
- Although technically it's a 3D input, we want to process it as a 2D, where each (x,y) position has 3 values

- Convolutional neurons work full depth
  - Filters are 3x3x3, or 5x5x3 or 7x7x3
- Each convolved filter produces a 2D output
- The output of a layer is a 3D volume
  - A channel per convolving neuron
  - Each channel showing the existence of a different learnable feature



CNN VOLUMES

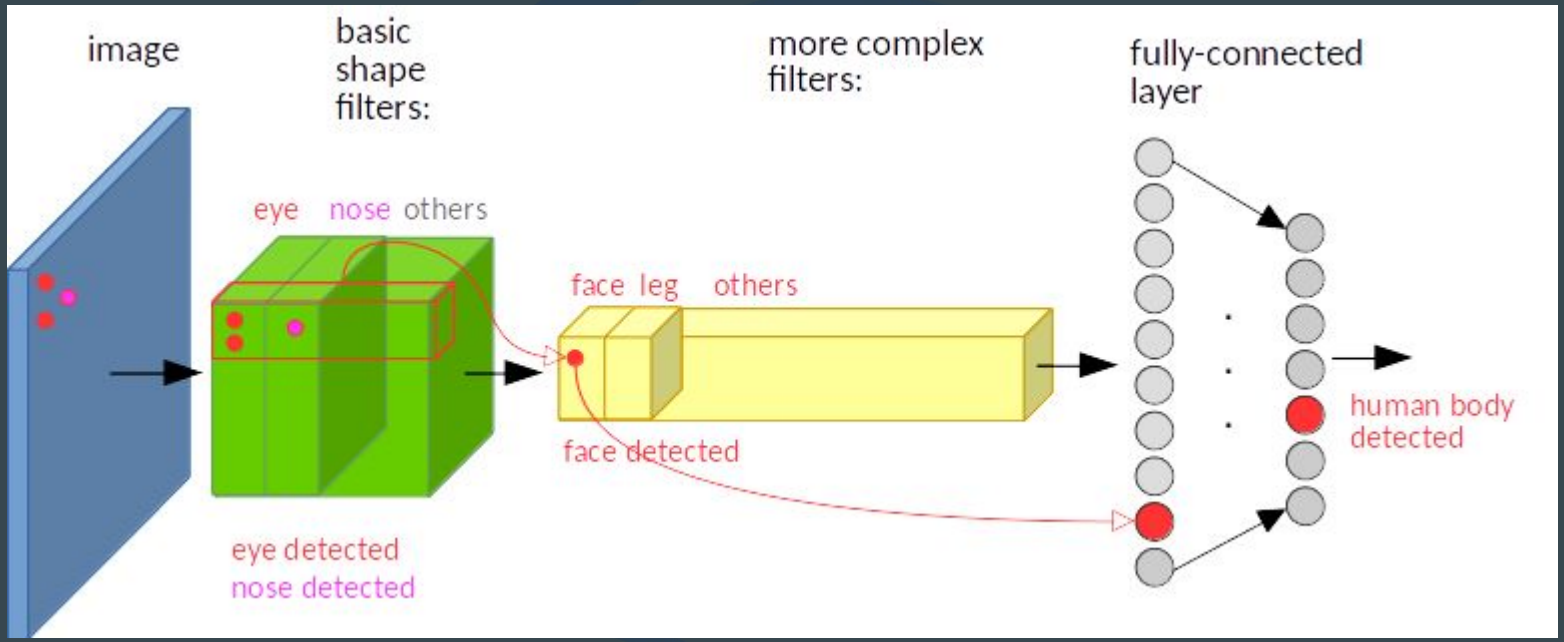
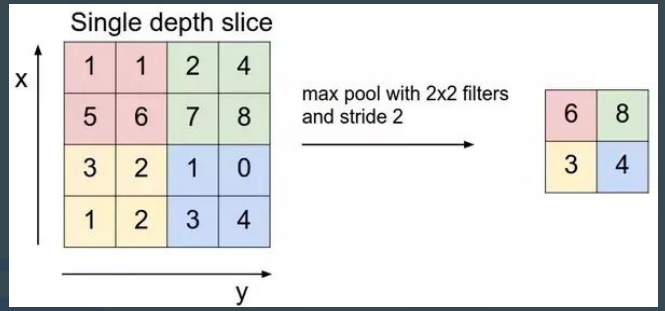
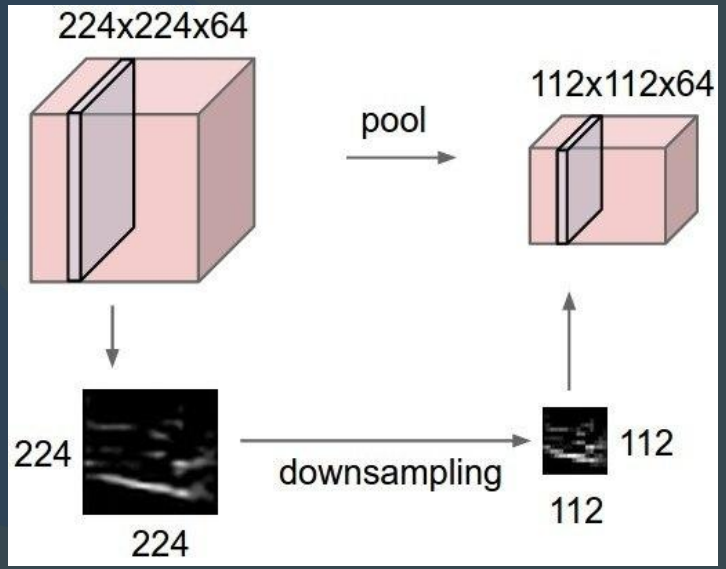


image
basic shape filters:
more complex filters:
fully-connected layer
eye
nose
others
face
leg
others
face detected
eye detected
nose detected
human body detected

POOLING

- A certain degree of spatial invariance is desired *sometimes*
- Pooling layers provide that...
  - Without parameters to learn
  - Apply a simple function
    - Max or Avg
  - While losing spatial accuracy
  - Not applied full depth (channel wise)
  - Also reduce the <x,y> dimensionality of the volume





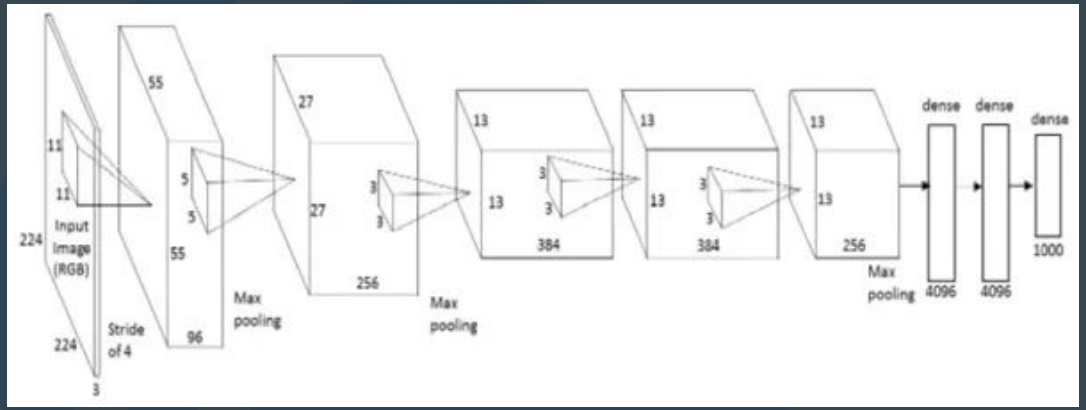
CNN ARCHITECTURES

General tip:

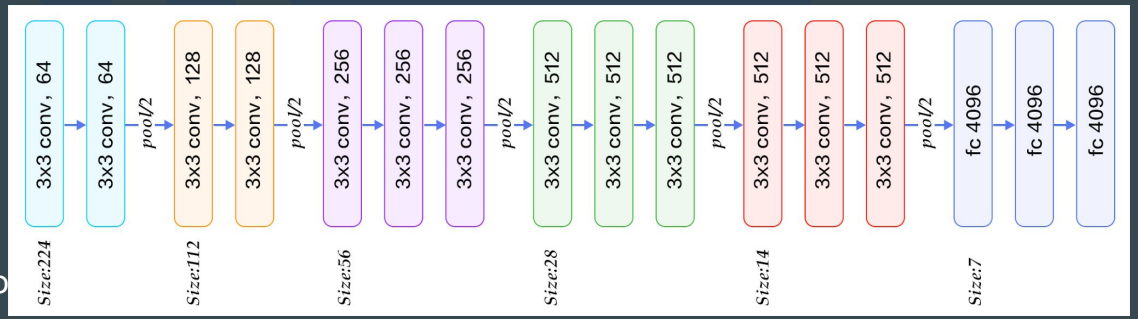
- Reduce the number of conv filters with depth

- Most popular follow the pattern: **conv-pool-conv-pool...-fc-fc**

AlexNet (2012) [26]

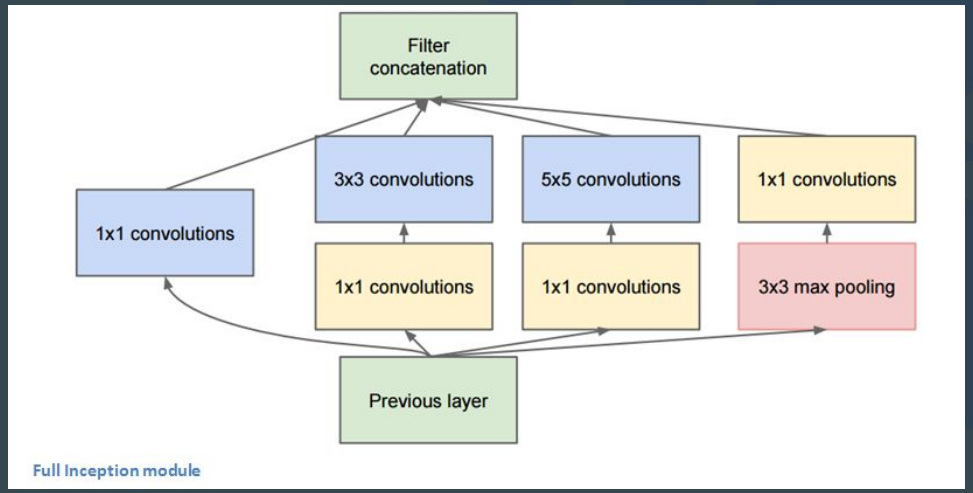


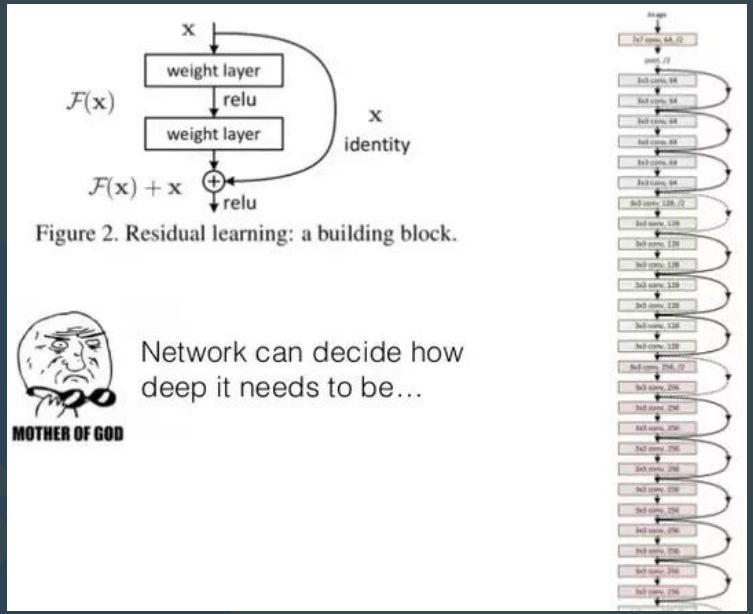
VGG16 (2014) [36]



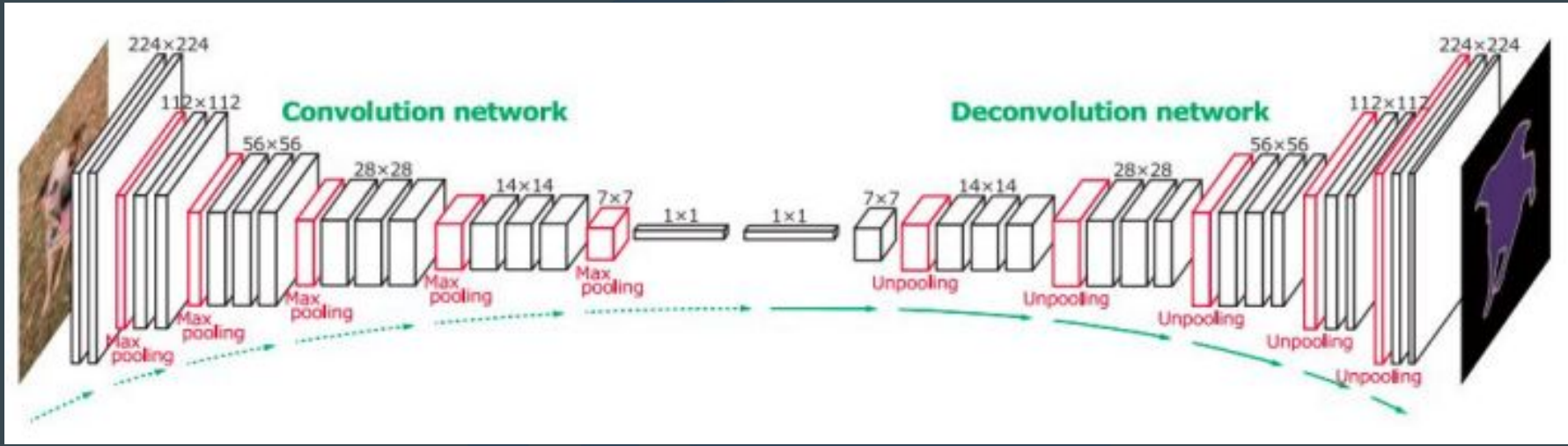
OTHER ARCHITECTURES

- ResNet (Microsoft) [38]: A CNN with 1K layers
- Inception (Google) [37 and following work]: Inception modules





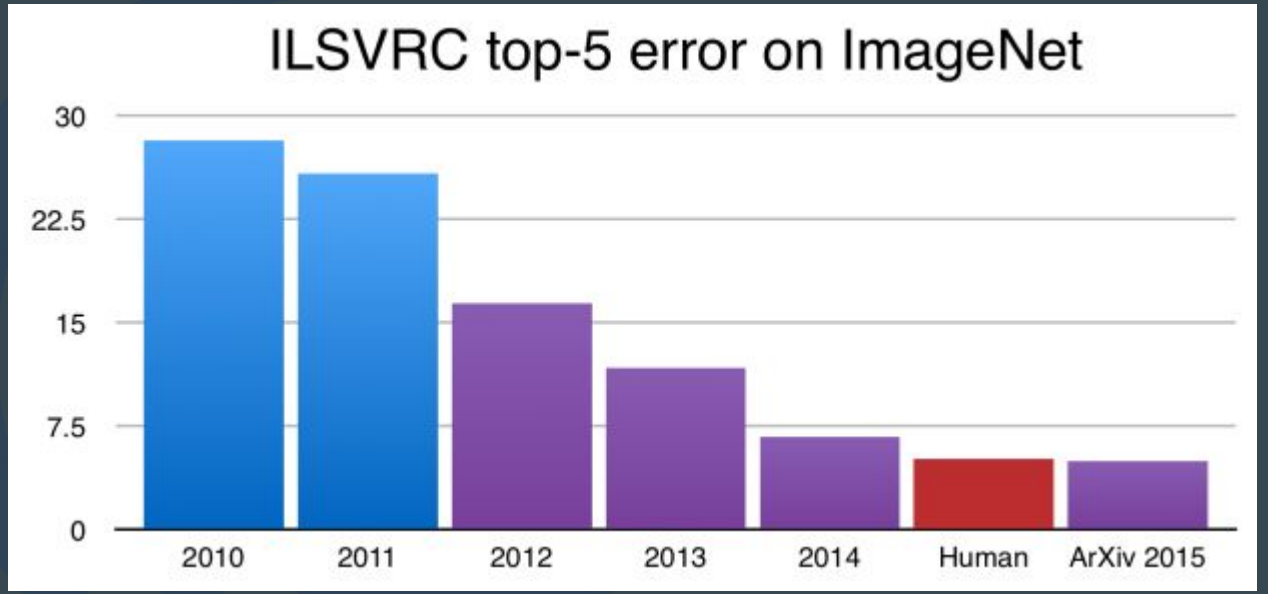
OTHER ARCHITECTURES



Convolution + Deconvolution
Provides pixel level classification
See [45] for example

IMAGENET

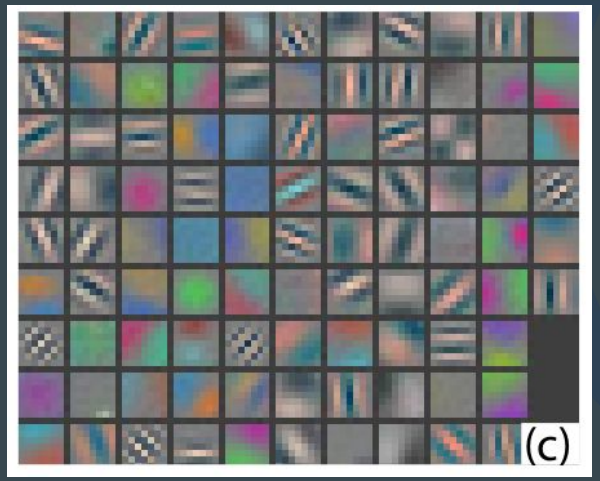
- Image recognition challenge
  - 2012: “Its impossible”
  - 2015: “Its solved”
- 1K categories
  - 120 breeds of dogs
- 1.2M labeled images for training

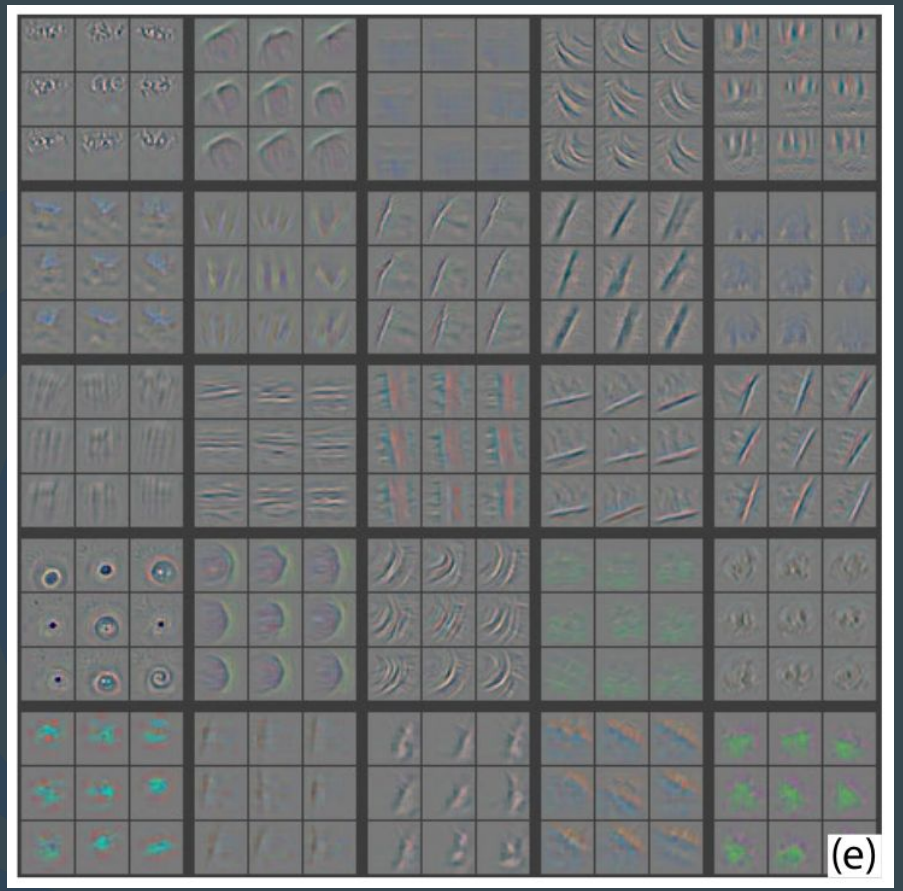


CNN FROM THE INSIDE

Not-so-black box

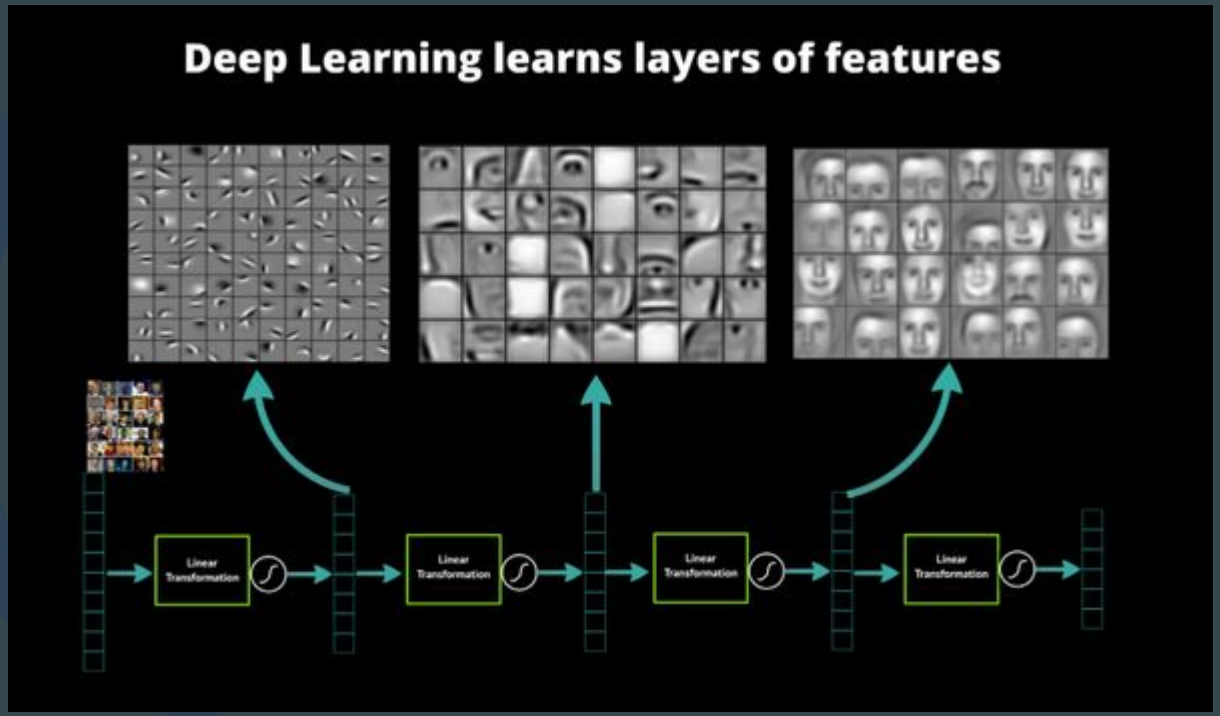
Early conv layers learn simple filters





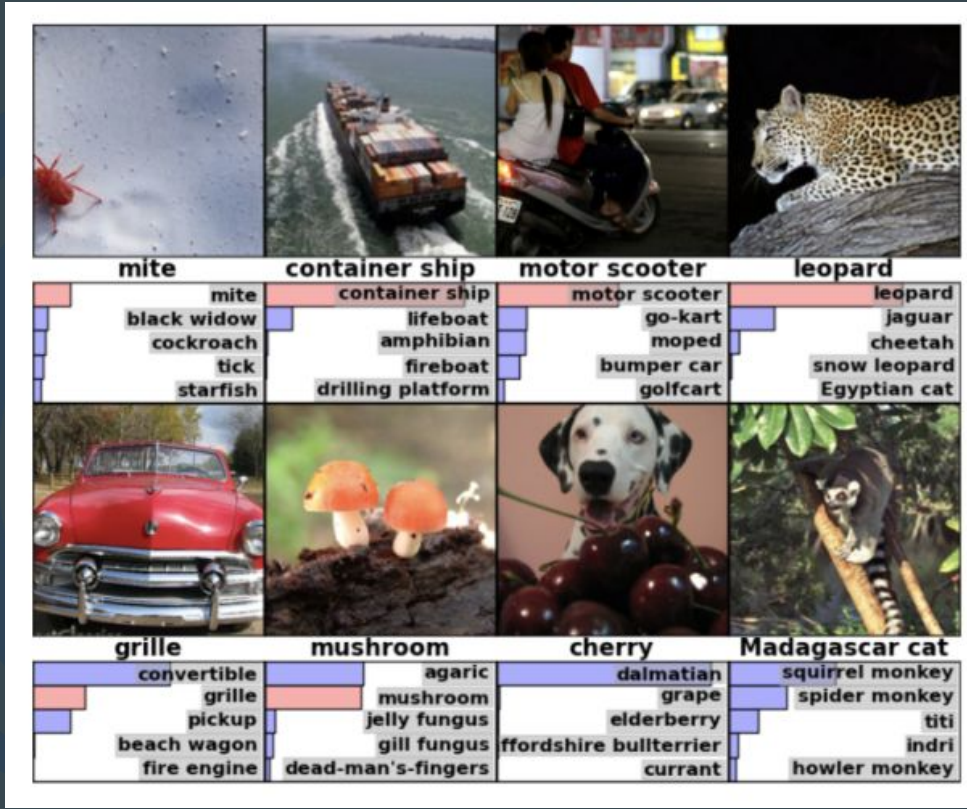
CNN FROM THE INSIDE 2

Later conv layers learn increasingly complex patterns



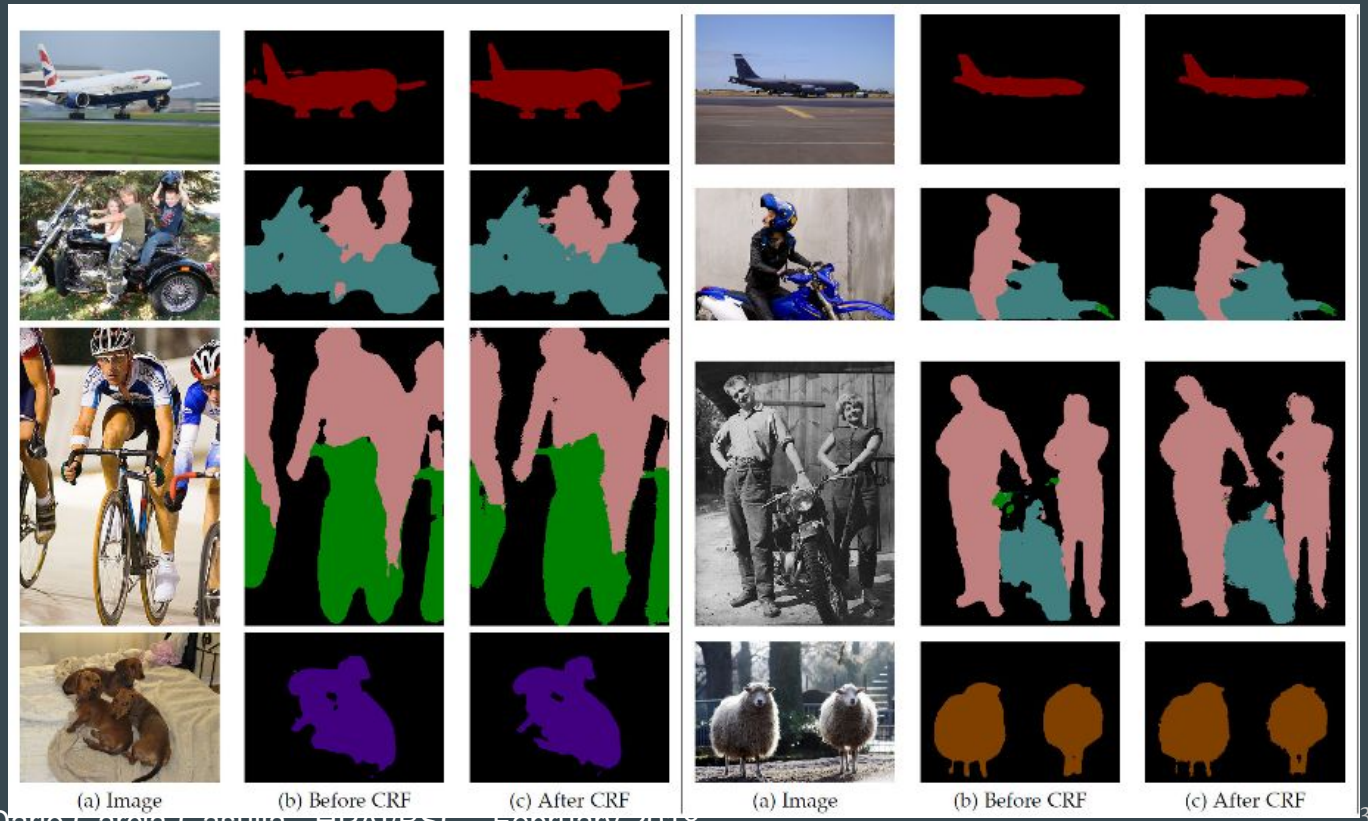
CNN APPLICATIONS 1/5

Image Classification



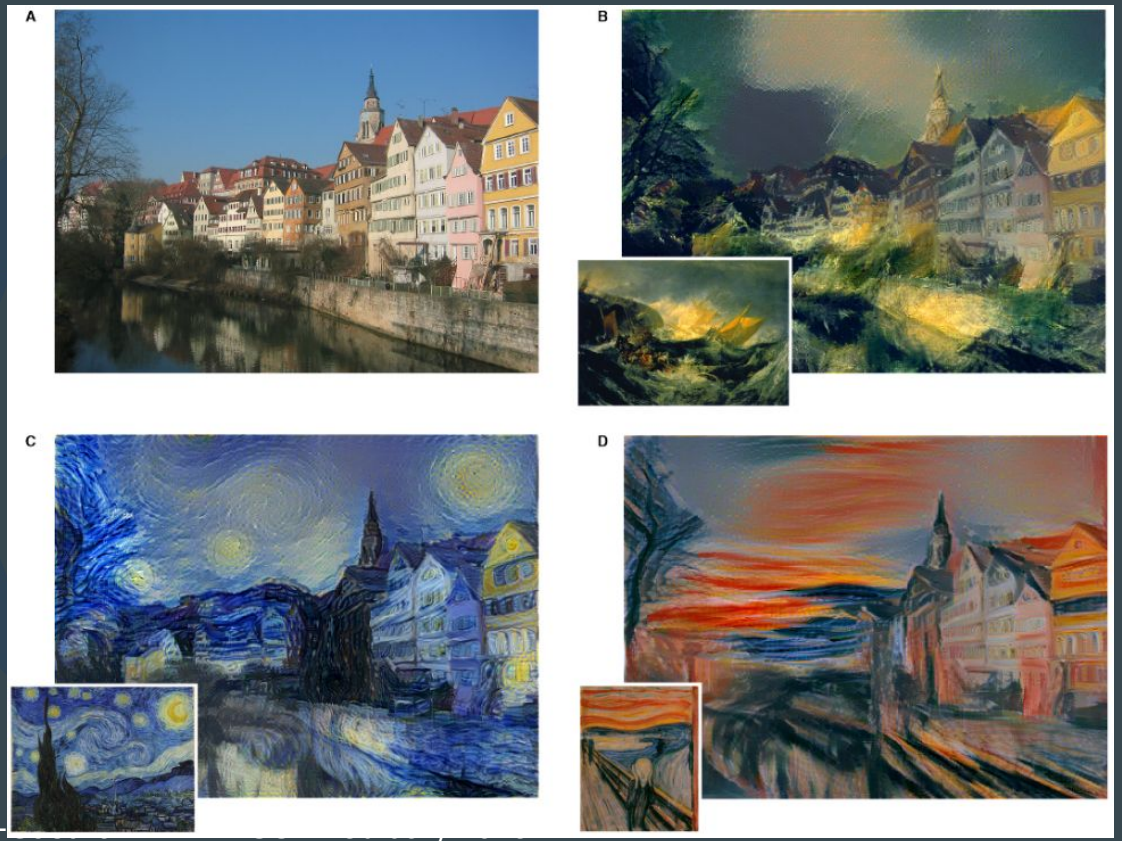
CNN APPLICATIONS 2/5

Image Segmentation



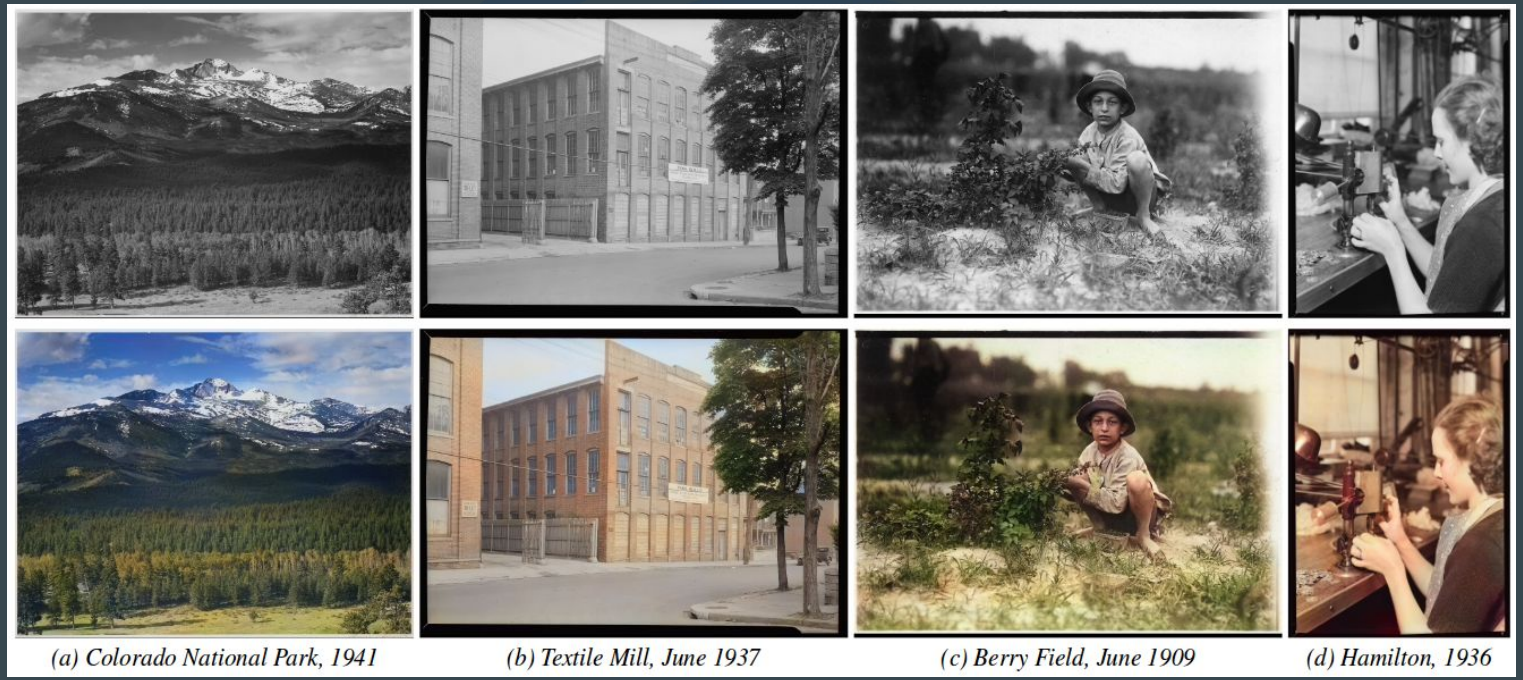
CNN APPLICATIONS 3/5

Style Transfer



CNN APPLICATIONS 4/5

Image Colorization

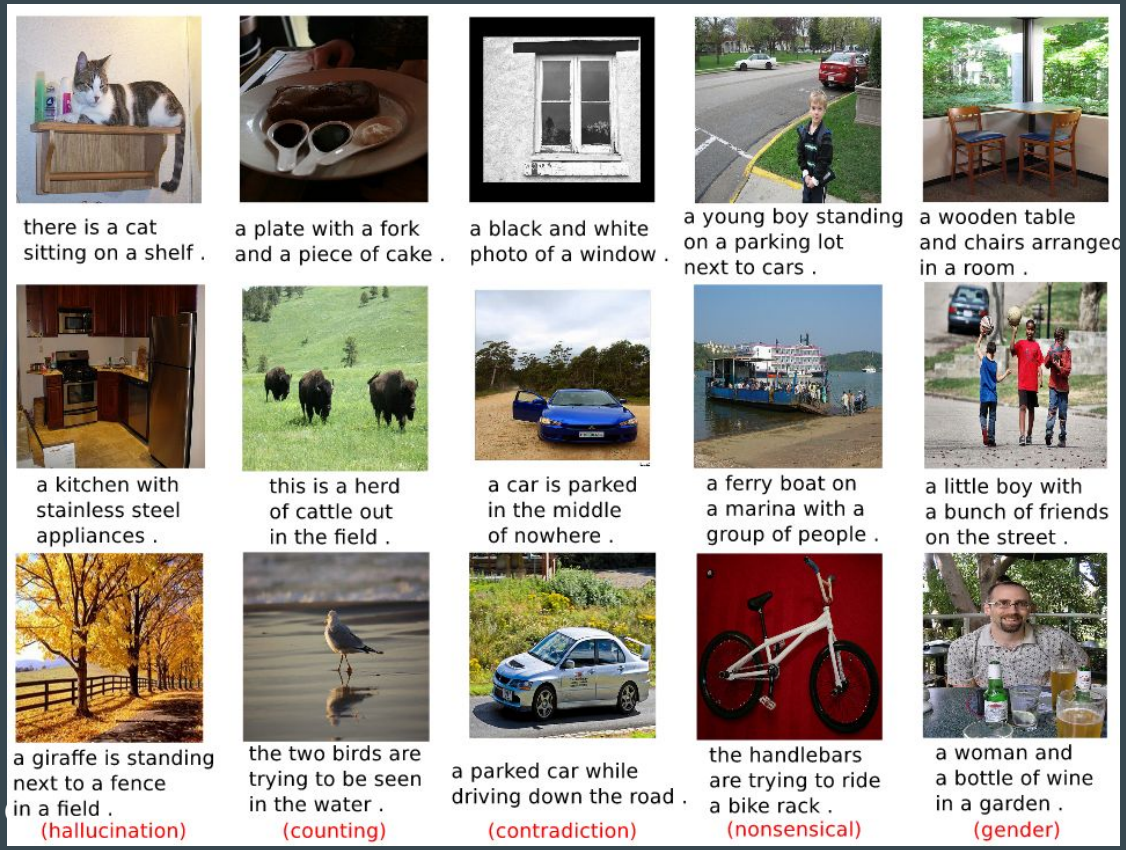


(a) Colorado National Park, 1941 (b) Textile Mill, June 1937 (c) Berry Field, June 1909 (d) Hamilton, 1936

CNN APPLICATIONS 5/5

Image Captioning

Image Retrieval



FRAMEWORKS

All open source. Bold ones available in BSC clusters

- **Caffe** (Berkeley) - C++ with Python interface. Primarily CNNs
- Theano (U. Montreal) - Python. End of development announced in 2017
- Torch - Lua and C
- **TensorFlow** (Google) - C++ with Python interface.
- **Keras** - Python. High level interface to other frameworks
- And many others: CNTK (Microsoft), PyTorch (Facebook), Caffe2 (Facebook), PaddlePaddle (Baidu), DeepLearning4j (in Java), ...

TOMORROW - HANDS ON!

- Exercises on training FNNs and CNNs using Keras
  - Keras is usable from minute 1.
- Other possible seminars
  - TensorFlow: More flexible and powerful, but has a steep learning curve
  - Recurrent Neural Nets: For processing sequences
  - Embedding spaces: Reusing learnt features and multimodal spaces
  - High Performance aspects of DL

REFERENCES 1/8

[1] McCulloch, Warren S., and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity.” The bulletin of mathematical biophysics 5.4 (1943): 115-133.

[2] Rosenblatt, Frank. “The perceptron: A probabilistic model for information storage and organization in the brain.” Psychological review 65.6 (1958): 386

[3] Mark I Perceptron Operators’ Manual

[4] Kurzweil, Ray. How to create a mind: The secret of human thought revealed. Penguin, 2013.

[5] Misky, M., and S. Papert. “Perceptrons: an introduction to computational geometry.” (1969).

[6] https://en.wikipedia.org/wiki/Perceptrons_(book)

[7] Werbos, Paul John. “Beyond regression: New tools for prediction and analysis in the behavioral sciences.” Doctoral Dissertation, Applied Mathematics, Harvard University (1974)

REFERENCES 2/8

[8] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. No. ICS-8506. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[9] http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

[10] http://neuralnetworksanddeeplearning.com/chap2.html

[11] LeCun, Yann, et al. “Backpropagation applied to handwritten zip code recognition.” Neural computation 1.4 (1989): 541-551.

[12] Fukushima, Kunihiko. “Neocognitron: A hierarchical neural network capable of visual pattern recognition.” Neural networks 1.2 (1988): 119-130.

[13] http://intellabs.github.io/RiverTrail/tutorial/

[14] http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

REFERENCES 3/8

[15] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[16] https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

[17] http://xrds.acm.org/blog/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/

[18] https://www.quora.com/How-is-a-convolutional-neural-network-able-to-learn-invariant-features

[19] https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

[20] https://www.quora.com/What-is-the-vanishing-gradient-problem

[21] http://cs231n.github.io/neural-networks-1/#actfun

[22] http://cs231n.github.io/neural-networks-3/

REFERENCES 4/8

[23] https://github.com/fchollet/keras/issues/68

[24] hthttps://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdftp://cs231n.github.io/neural-networks-2/

[25] http://cs231n.github.io/convolutional-networks/

[26] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. “Imagenet classification with deep convolutional neural networks.” Advances in neural information processing systems. 2012.

[27] Srivastava Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting.” Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[28] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. “A neural algorithm of artistic style.” arXiv preprint arXiv:1508.06576 (2015).http://cs231n.github.io/understanding-cnn/

[29] Zeiler, Matthew D., and Rob Fergus. “Visualizing and understanding convolu tional networks.” European conference on computer vision. Springer, Cham, 201 4.

REFERENCES 5/8

[30] Chen, Liang-Chieh, et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.” arXiv preprint arXiv:1606.00915 (2016).

[31] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. “A neural algorithm of artistic style.” arXiv preprint arXiv:1508.06576 (2015).

[32] Zhang, Richard, Phillip Isola, and Alexei A. Efros. “Colorful image colorization.” European Conference on Computer Vision. Springer International Publishing, 2016.

[33] Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. “Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification.” ACM Transactions on Graphics (TOG) 35.4 (2016): 110.

[34] http://www.tensorflowexamples.com/2017/01/transposed-convnets-or-deconvolution.html

[35] Kiros, Ryan, Ruslan Salakhutdinov, and Richard S. Zemel. “Unifying visual-semantic embeddings with multimodal neural language models.” arXiv preprint arXiv:1411.2539 (2014).

REFERENCES 6/8

[36] Simonyan, Karen, and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv:1409.1556 (2014).

[37] Szegedy, Christian, et al. “Going deeper with convolutions.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[38] He, Kaiming, et al. “Deep residual learning for image recognition.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[39] Ioffe, Sergey, and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” International Conference on Machine Learning. 2015.

[40] https://www.linkedin.com/pulse/gradient-descent-simple-words-parth-jha

[41] http://ruder.io/optimizing-gradient-descent/

[42] https://becominghuman.ai/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c

REFERENCES 7/8

[43] https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b

[44] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. “Layer normalization.” arXiv preprint arXiv:1607.06450 (2016).

[45] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation.” International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham, 2015.

[46] http://blog.mrtanke.com/2016/10/24/An-overview-of-gradient-descent-optimization-algorithms/

[47] https://medium.com/@erikhallstrm/backpropagation-from-the-beginning-77356edf427d

[48] Hebb, Donald. *0.(1949) The Organization of Behavior*.

REFERENCES 8/8

Other uncited sources:

https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

http://iamaaditya.github.io/2016/03/one-by-one-convolution/

http://www.robots.ox.ac.uk/~vgg/practicals/cnn/

IMAGE SOURCES 1/4

Slide 3: [1]

Slide 5: [3]

Slide 6: https://towardsdatascience.com/radial-basis-functions-neural-networks-all-we-need-to-know-9a88cc053448

Slide 9: https://www.quora.com/What-is-Stochastic-Gradient-Descent

Slide 10: http://www.bogotobogo.com/python/scikit-learn/images/Batch-vs-Stochastic-Gradient-Descent/

Slide 12: https://www.researchgate.net/publication/320270458_A_System_Based_on_Artificial_Neural_Networks_for_Automatic_Classification_of_Hydro-generator_Stator_Windings_Partial_Discharges/figures?lo=1&utm_source=google&utm_medium=organic

Slide 13: [19]

Slide 15: [22]

IMAGE SOURCES 2/4

Slide 16: [40], https://stats.stackexchange.com/questions/247781/what-causes-the-elbow-shape-of-the-loss-curve, [22]

Slide 17: [41]

Slide 18: http://scikit-learn.org/stable/modules/learning_curve.html

Slide 22: [13]

Slide 23: [14]

Slide 24: [15]

Slide 25: [16], [16]

Slide 26: [17]

Slide 27: [18]

IMAGE SOURCES 3/4

Slide 28: [25], [25]

Slide 29: https://www.quora.com/What-are-the-differences-among-AlexNet-GoogleNet-and-VGG-in-the-context-of-convolutional-neural-networks , https://stackoverflow.com/questions/47128782/when-to-insert-pooling-layer-between-convolution-layers

Slide 30: https://www.quora.com/What-should-I-do-when-adding-more-hidden-layers-to-a-deep-neural-network-causes-performance-to-decrease , https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

Slide 32: [29]

Slide 33: Quora

Slide 34: [26]

Slide 35: [30]

IMAGE SOURCES 4/4

Slide 36: [31]

Slide 38: [35]