Machine Learning & Predictive Analytics

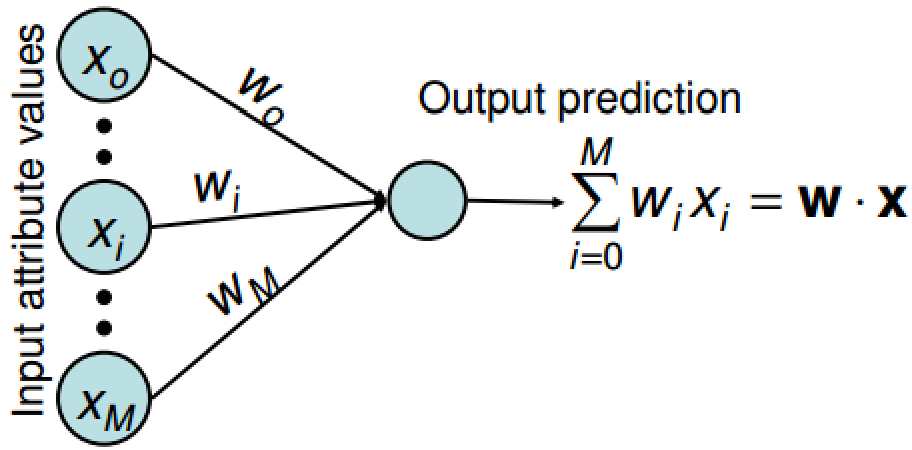
Class 6

Arnab Bose, Ph.D.

MSc Analytics

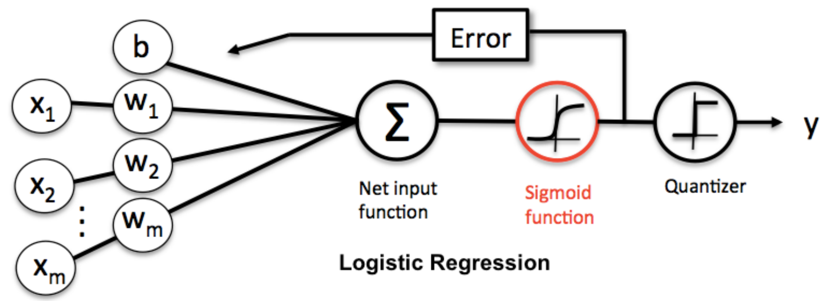
University of Chicago

Linear Regression special case of ANN

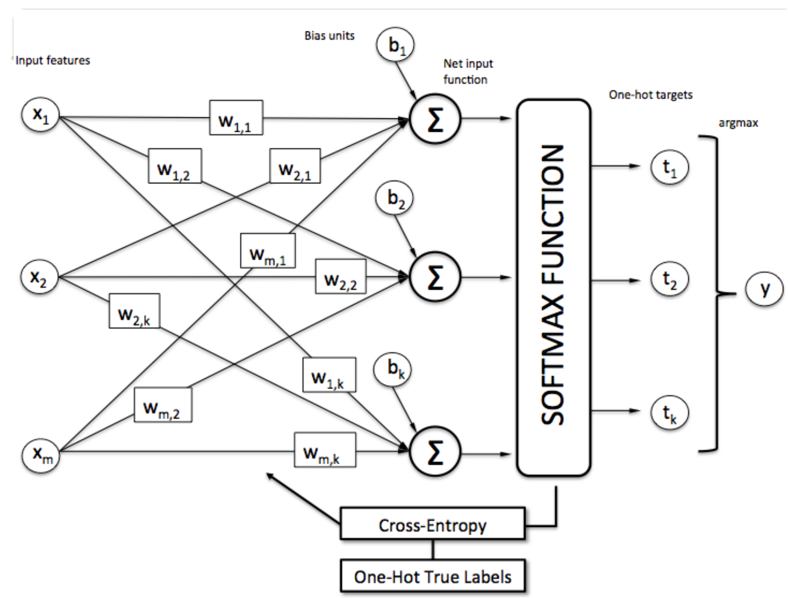


https://stats.stackexchange.com/questions/253337/what-is-the-difference-between-regular-linear-regression-and-deep-learning-lin

Logistic Regression special case of ANN



Logistic Regression

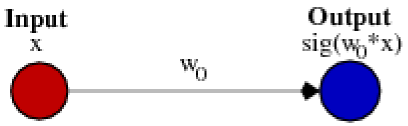


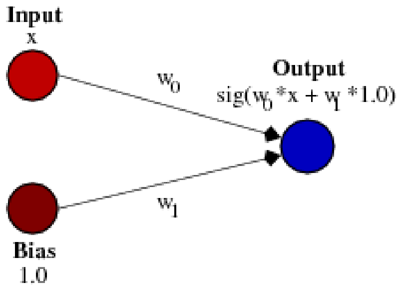
Softmax Regression

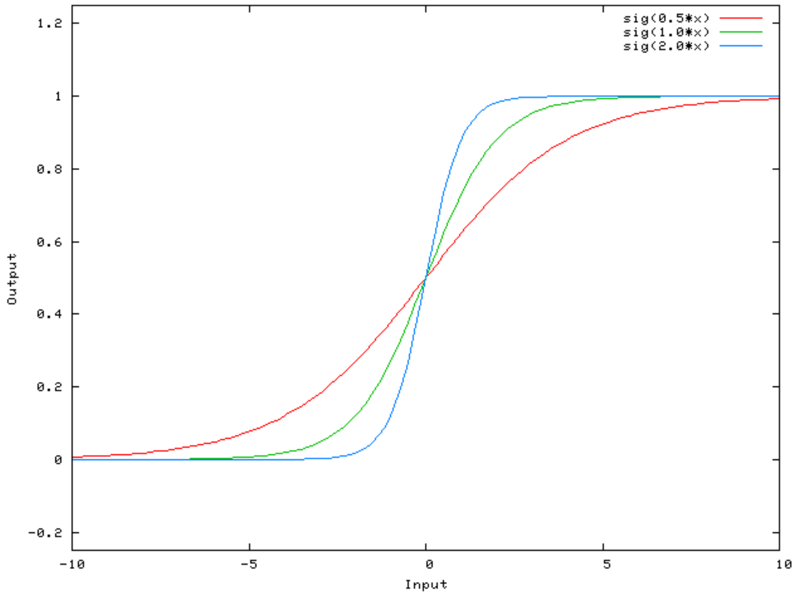
https://stats.stackexchange.com/questions/43538/difference-between-logistic-regression-and-neural-networks/162548#162548

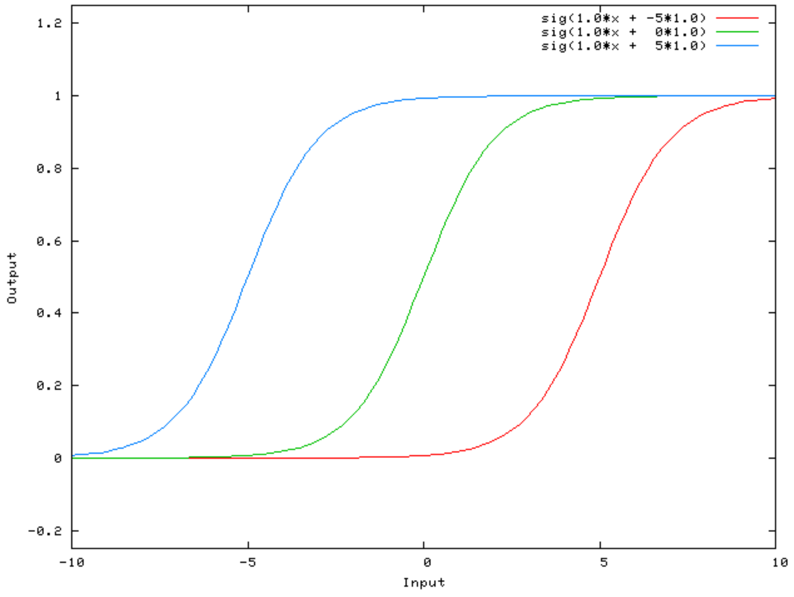
Role of Bias

Changing the weight w0 essentially changes the "steepness" of the sigmoid. Just changing the steepness of the sigmoid won't really work -- you want to be able to shift the entire curve to the right.



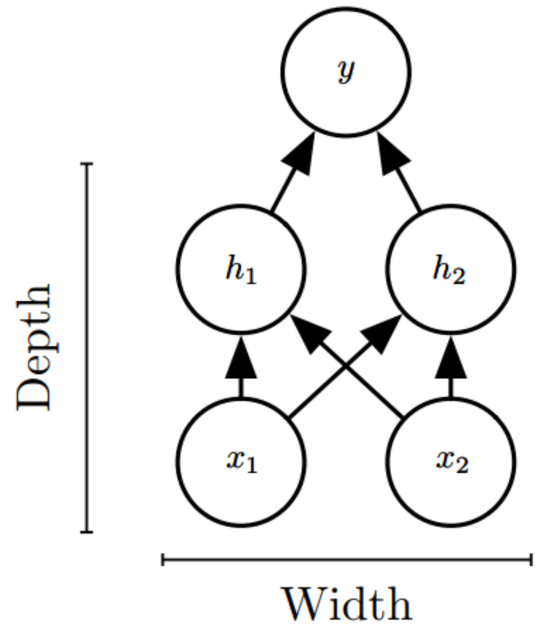






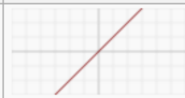
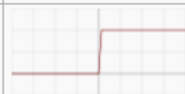
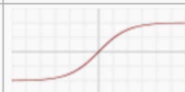
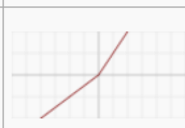
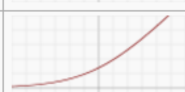
https://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks

ANN Architecture Basics



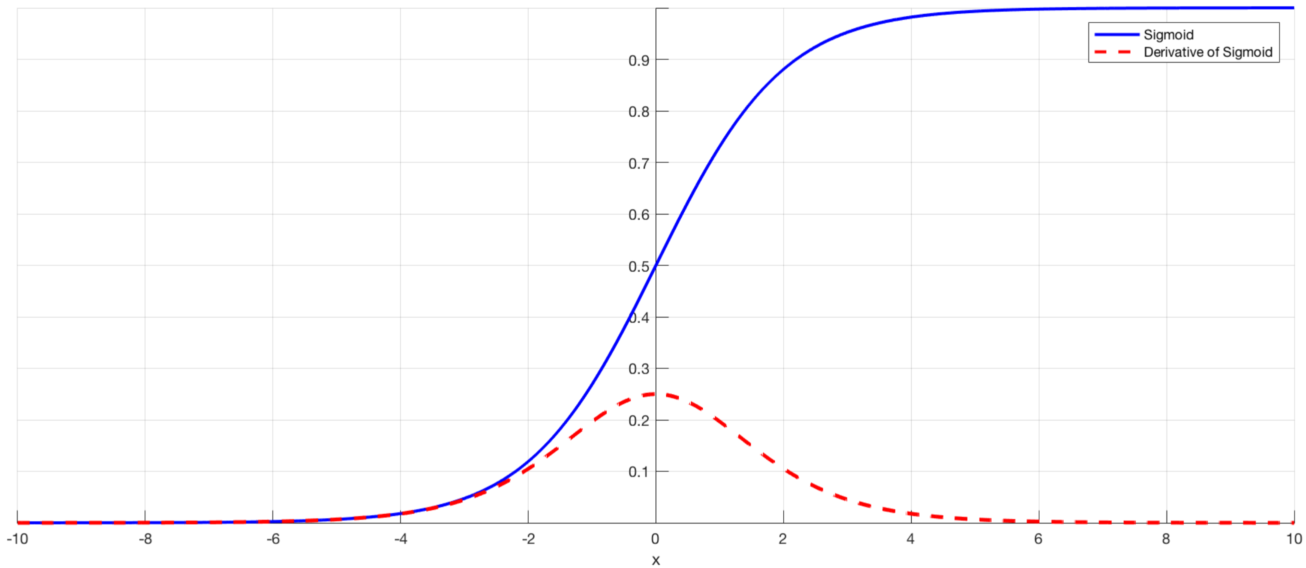
(Goodfellow 2017)

ANN Activation Functions & Derivatives

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

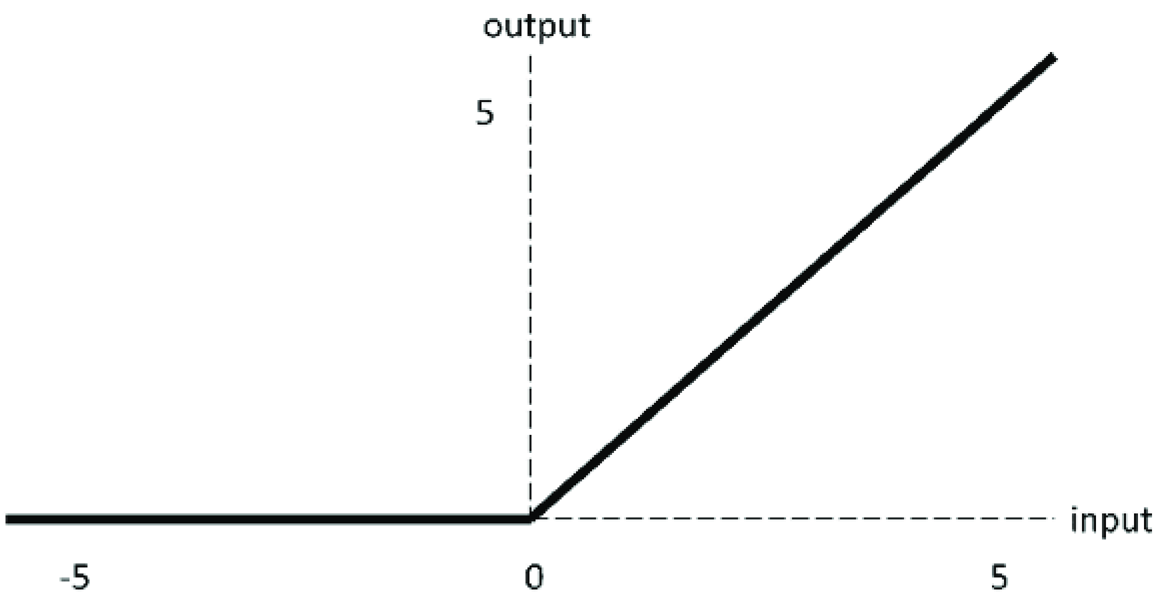
ANN Vanishing Gradient with Sigmoid (and also tanh)



Gradient decreases exponentially with backpropagation

https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

ReLU Activation Function

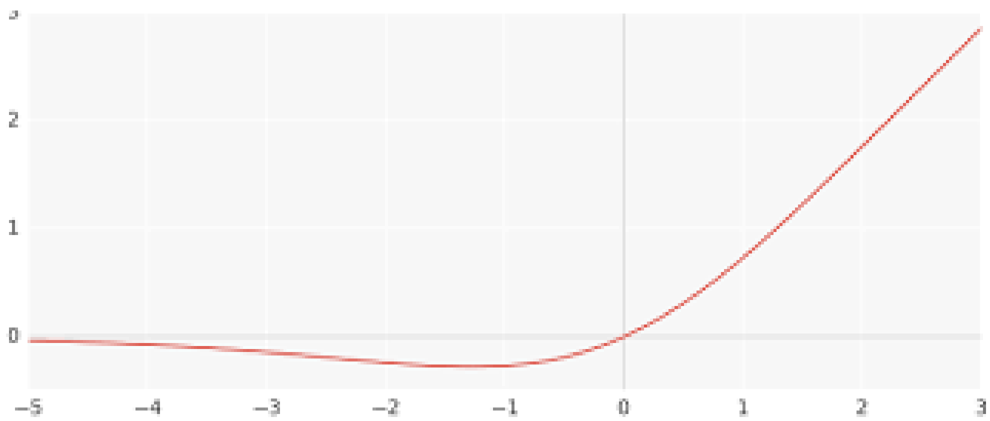


Non-saturating activation function since $\lim_{input \to \infty} f_{ReLU} = \infty$

NOTE – any activation function that is not non-saturating is saturating (intuition – the activation squeezes the input)

Swish Activation Function

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

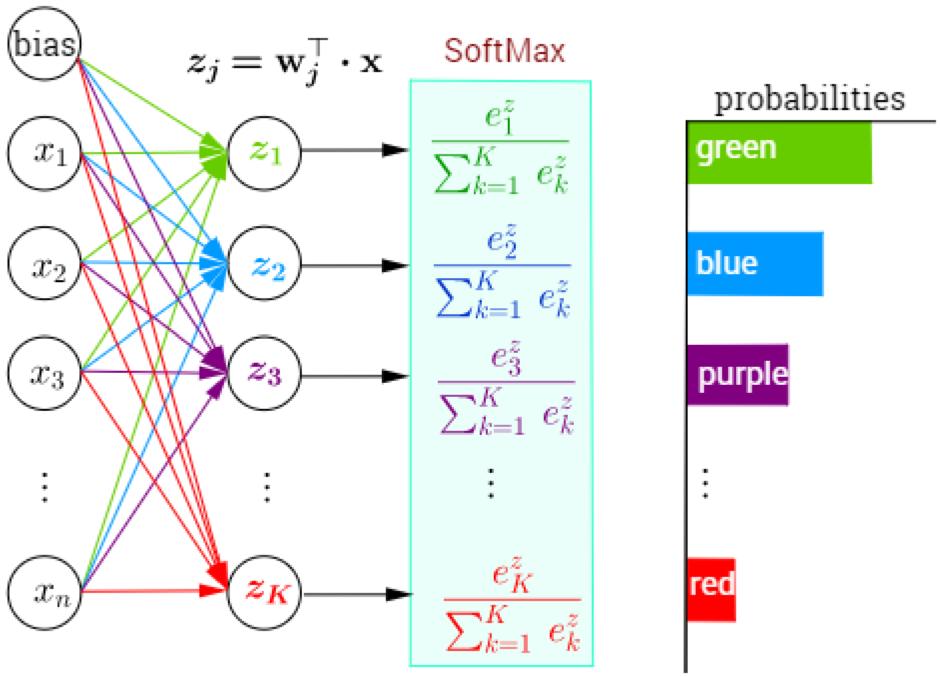


Advantage over ReLU:

1. Smooth function
2. Non-monotonic function

https://medium.com/analytics-vidhya/swish-booting-relu-from-the-activation-function-throne-78f87e5ab6eb

SoftMax Activation Function



https://deepnotes.io/category/cnn-series

ANN Loss / Cost Functions

1. Regression
    1. Mean Squared (Logarithmic) Error
    2. Mean Absolute Error

2. Classification
    1. Cross Entropy
    2. Kullback-Leibler Divergence
    3. Negative Log-Likelihood

3. Embedding
    1. L1 & L2 Hinge
    2. Cosine

https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

https://keras.io/losses/

ANN Recommendations

Output Type	Output Distribution	Output Layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary cross-entropy
Discrete	Multinoulli	Softmax	Discrete cross-entropy
Continuous	Gaussian	Linear	Gaussian cross-entropy (MSE)
Continuous	Mixture of Gaussian	Mixture Density	Cross-entropy
Continuous	Arbitrary	See part III: GAN, VAE, FVBN	Various

(Goodfellow 2017)

Universal Approximator Theorem

- One hidden layer is enough to *represent* (not *learn*) an approximation of any function to an arbitrary degree of accuracy
- So why deeper?
  - Shallow net may need (exponentially) more width
  - Shallow net may overfit more

(Goodfellow 2017)

Better Generalization with Greater Depth

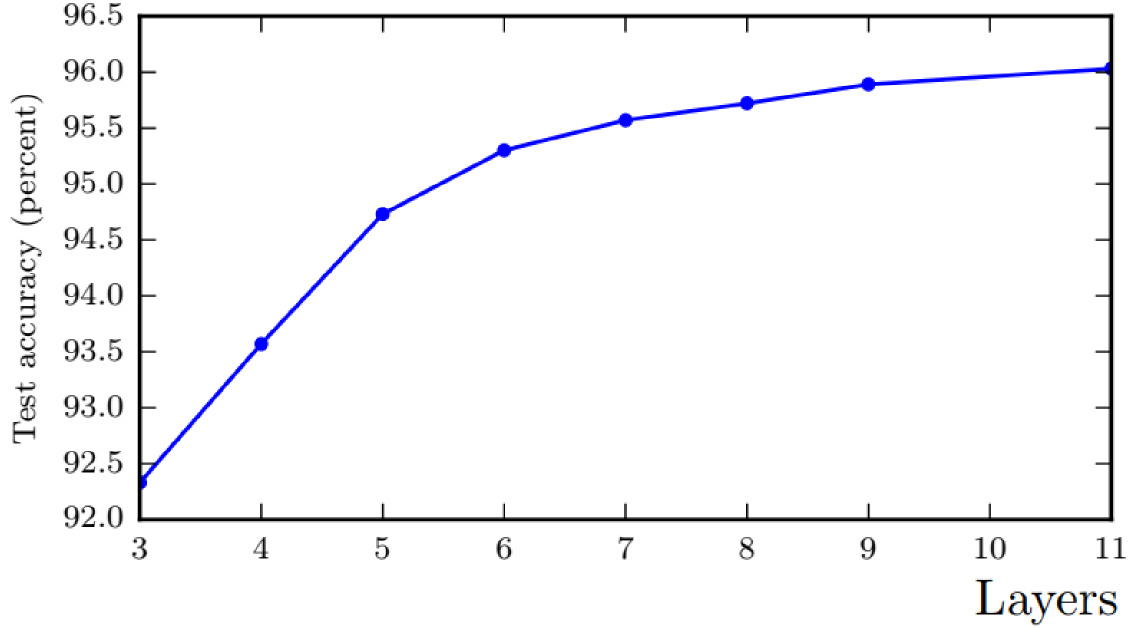
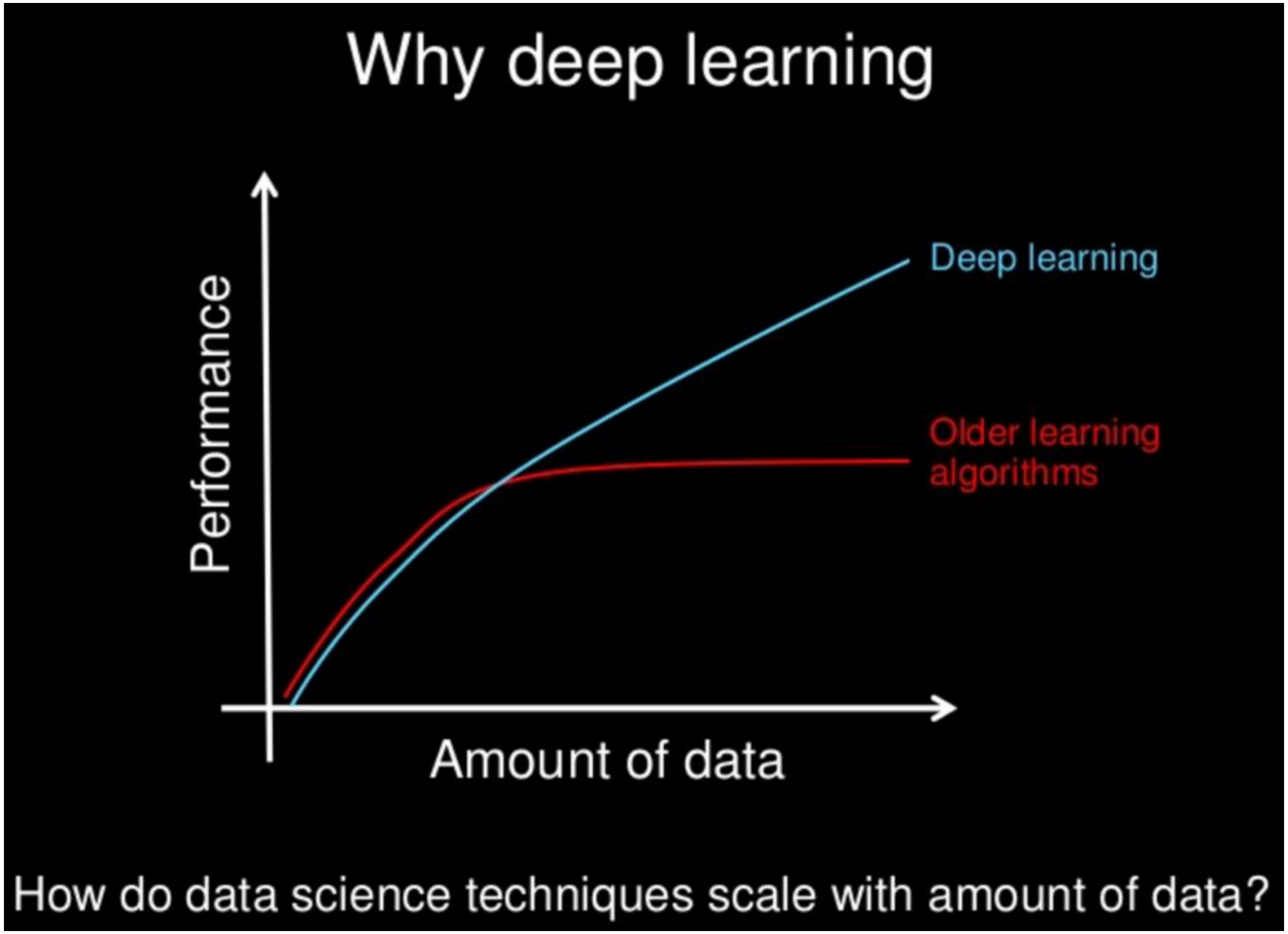


Figure 6.6

(Goodfellow 2017)

Deep Learning with Dataset Size



Why Deep Learning? Slide by [Andrew Ng](),

Large Shallow Model Overfit More

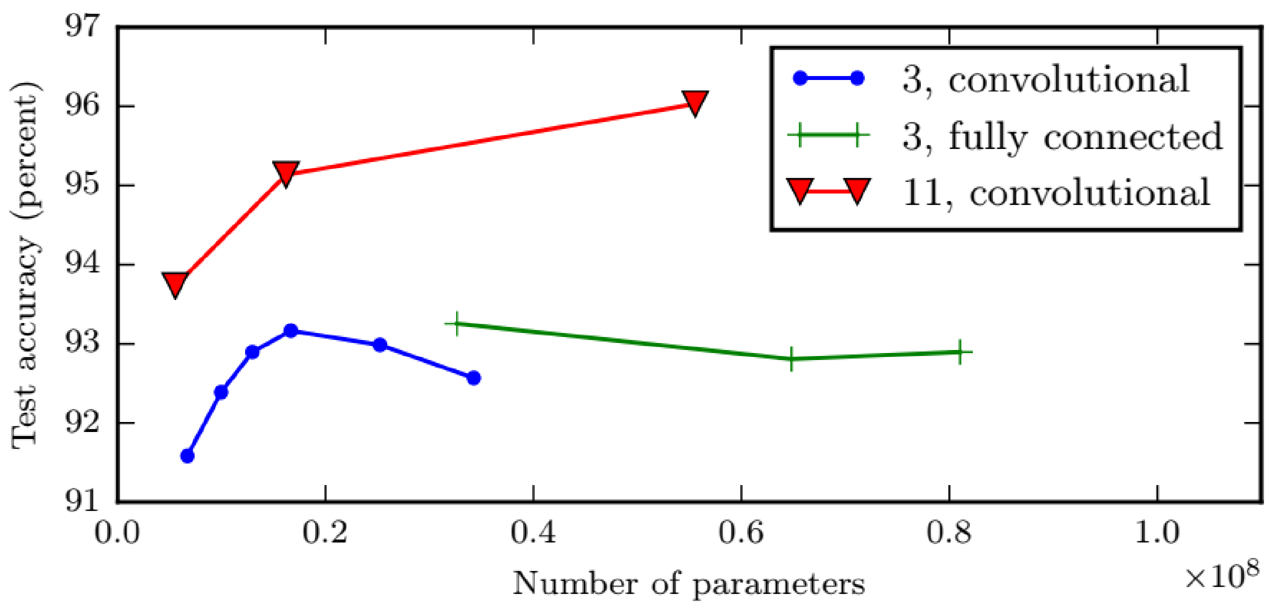


Figure 6.7

(Goodfellow 2017)

Convolution Neural Network (CNN)

- Convolutional Neural Networks (CNN) are biologically-inspired variants of MLPs.
- Emerge from the study of human brain visual cortex.
- Efficiency – CNN O($k \times n$) vs Dense O($m \times n$)

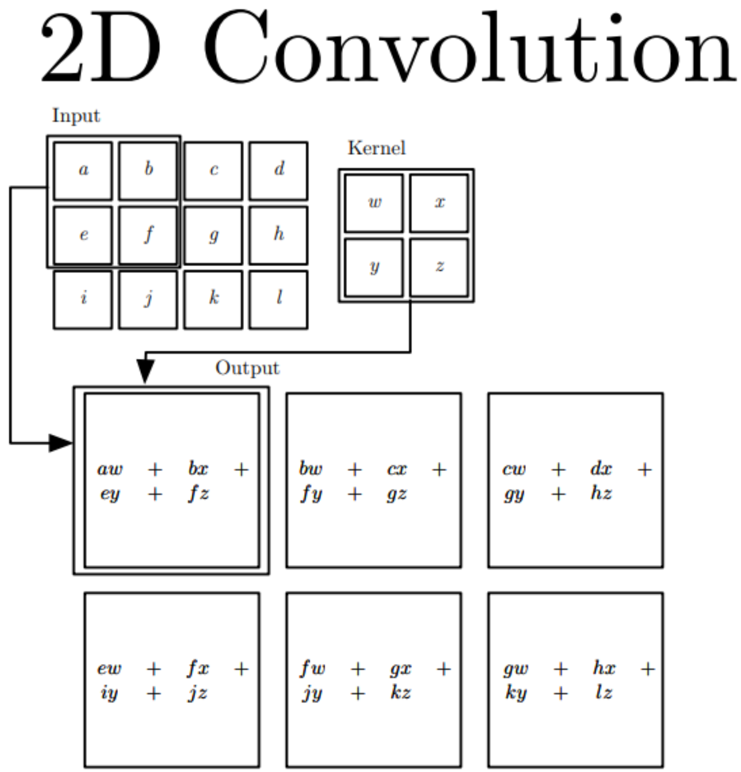


Figure 9.1

(Goodfellow 2016)

CNN – Terminologies

Receptive field – region of the input space CNN is working with / looking at

Zero padding – add zeros around the input to have a desired number of output

Stride – distance between 2 consecutive receptive fields

Filters – aka convolution kernels define the set of weights in the receptive field

Input Feature map – input to the CNN

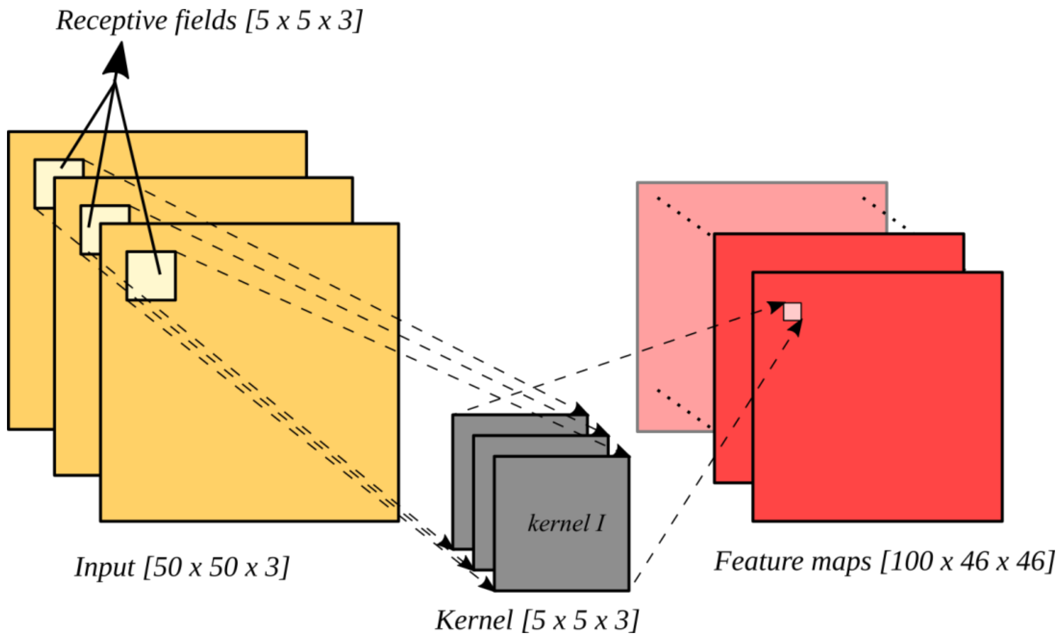
Output Feature map – aka activation map is the output of a filter

https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

https://stats.stackexchange.com/questions/234692/how-to-work-multiple-filter-region-sizes-2-3-and-4-in-cnn/234770

http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

CNN – Receptive Field



Receptive field(s): is a small portion of the input to produce only one node in a feature map.

Feature map(s): is a convolutional process output, a feature map can be said as a feature representation of filter's input. One feature map consists of many filter's outputs (from different receptive fields) from one kernel. The number of feature maps depends on the number of the kernel.

https://ai.stackexchange.com/questions/8701/what-is-the-difference-between-a-receptive-field-and-a-feature-map

CNN – Sparse Connectivity

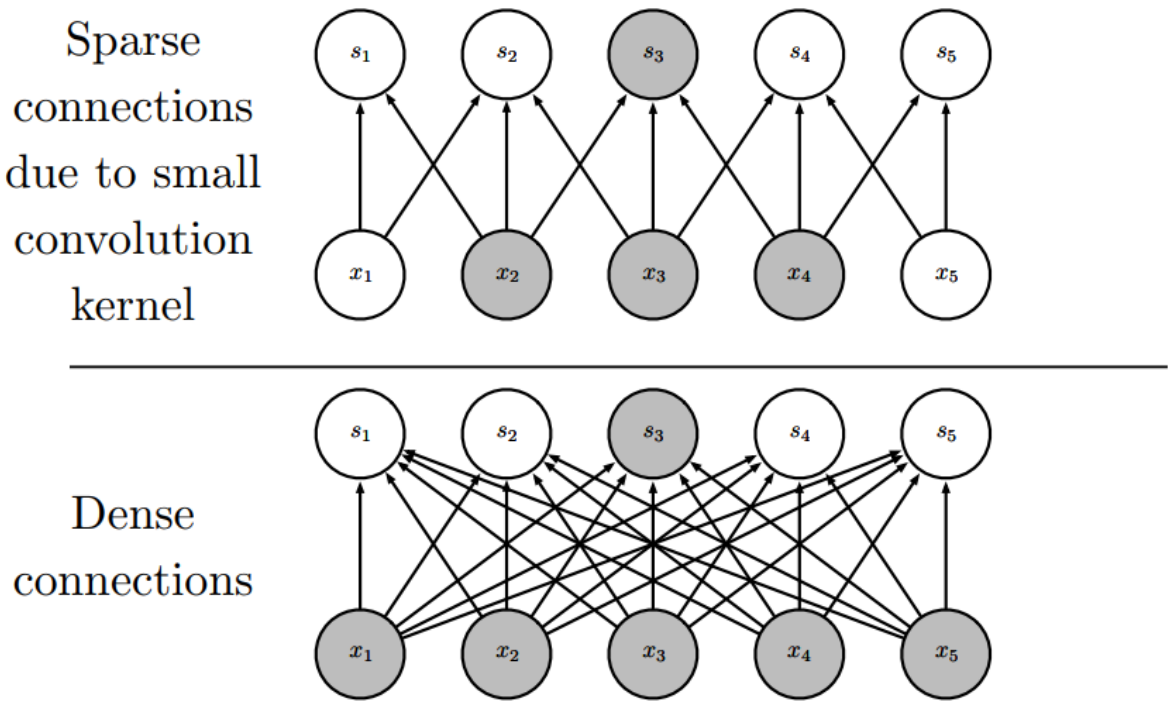


Figure 9.3

(Goodfellow 2016)

CNN – Parameter Sharing

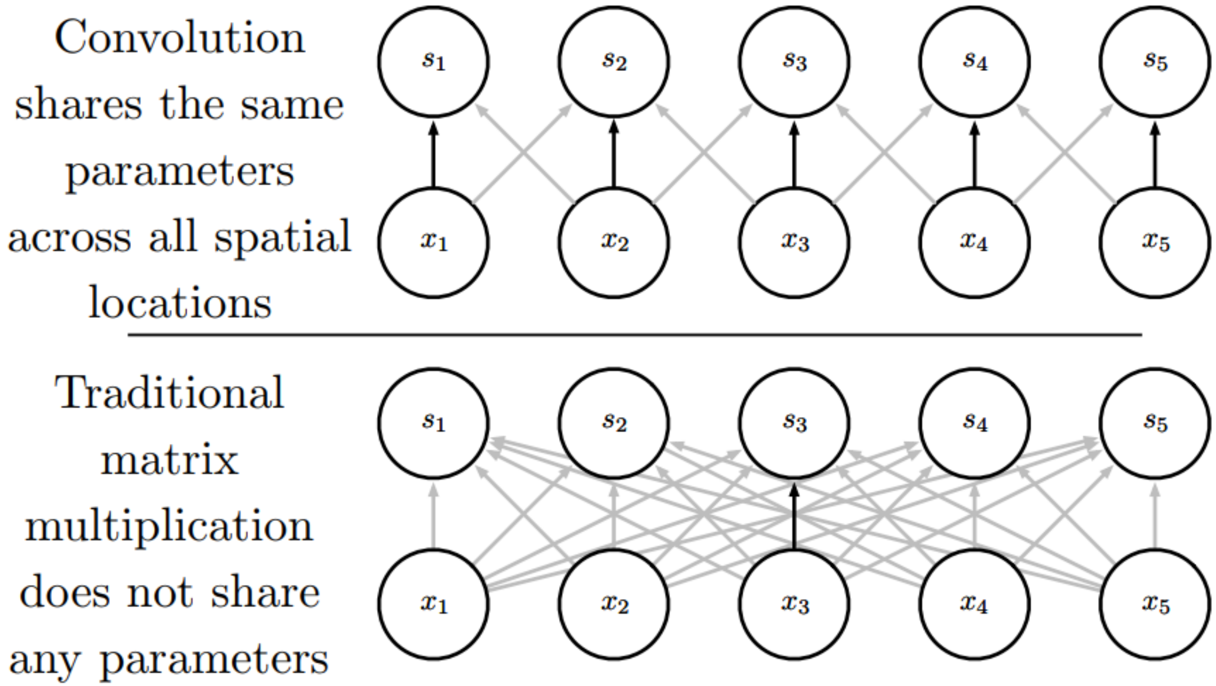


Figure 9.5

(Goodfellow 2016)

CNN – Pooling Invariance

1. Pooling replaces the output with a summary statistic such as max or average.
2. It helps to make the representation approximately **invariant** to small translations of the input.
3. Example below demonstrates max pooling.

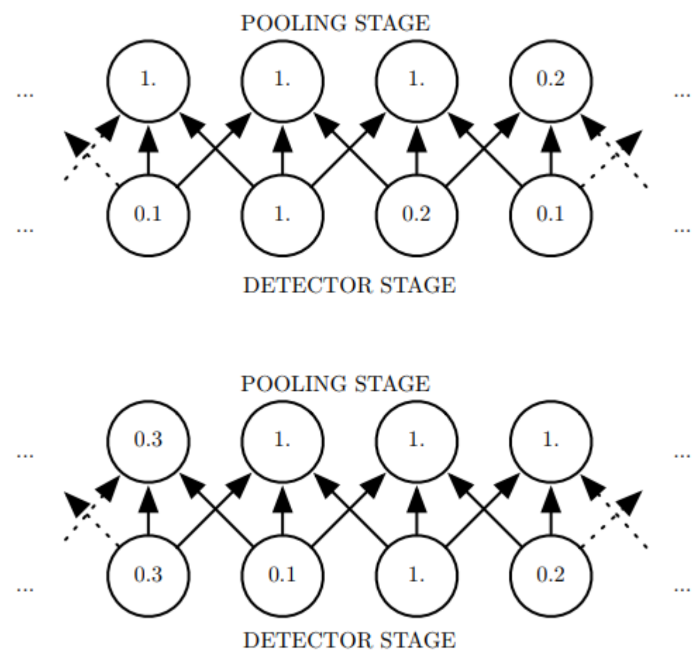


Figure 9.8

(Goodfellow 2016)

CNN – Stride

1. Stride is the distance between spatial locations where the convolution kernel is applied.
2. Default of stride of 1.
3. Below, top example demonstrates stride 2 while bottom demonstrates (default) stride 1.

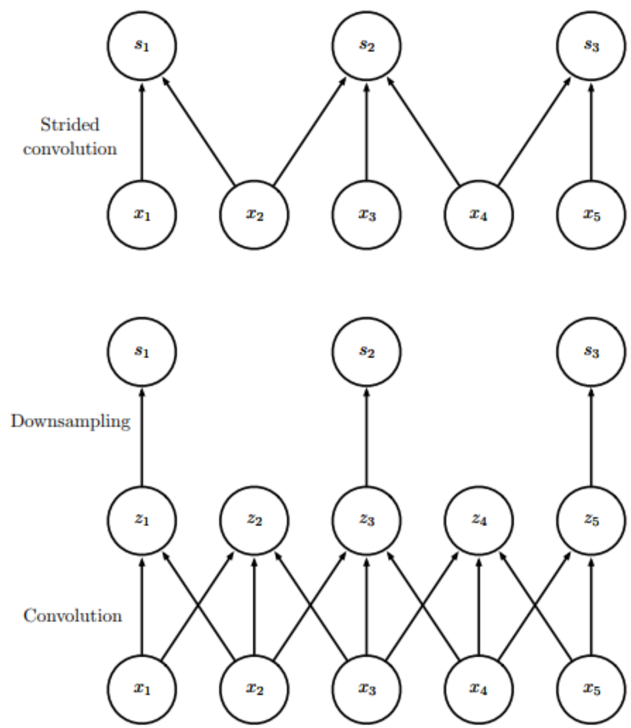


Figure 9.12

(Goodfellow 2016)

CNN – Pooling with Stride 2 Downsampling

Pooling width 3 and stride 2 reduces representation size by factor of 2.

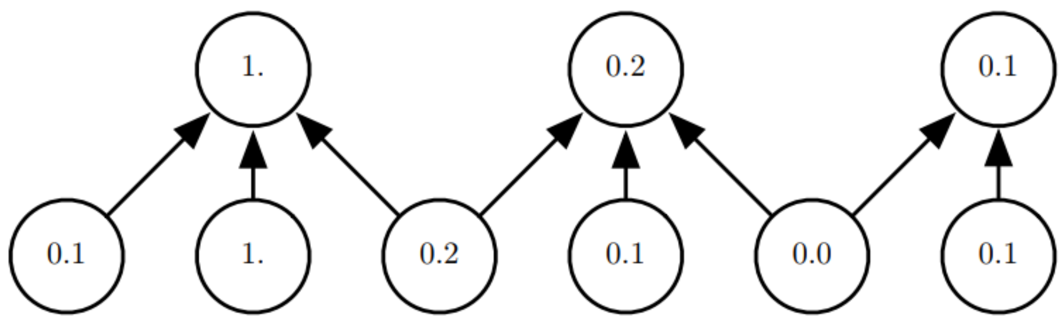


Figure 9.10

(Goodfellow 2016)

CNN – Connectivity Comparisons

1. Unshared convolution is a local connection with a small kernel and no parameter sharing.
2. Example below top is locally connected layer with width of 2 that demonstrates unshared convolution, middle is convolution with width of 2 (same connectivity as top but with parameter sharing) and bottom is fully connected.

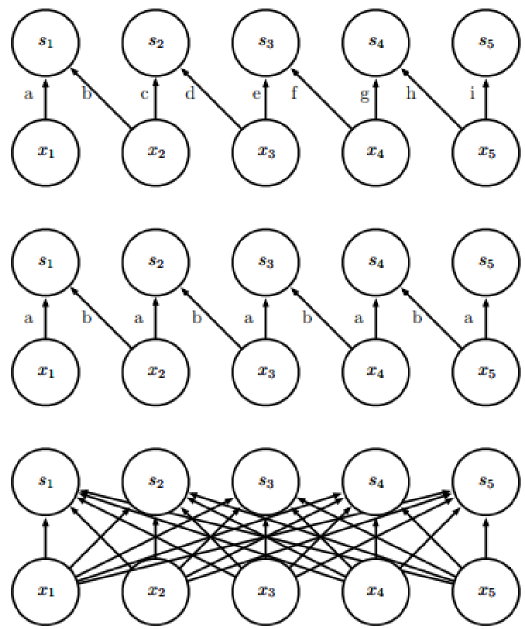


Figure 9.14

CNN – Tiled Convolution

1. Tiled convolution is a compromise between a local connection and a convolution layer.
2. Tiled convolution uses multiple kernels in rotation – in example below, 2 kernels (a, b) & (c, d).

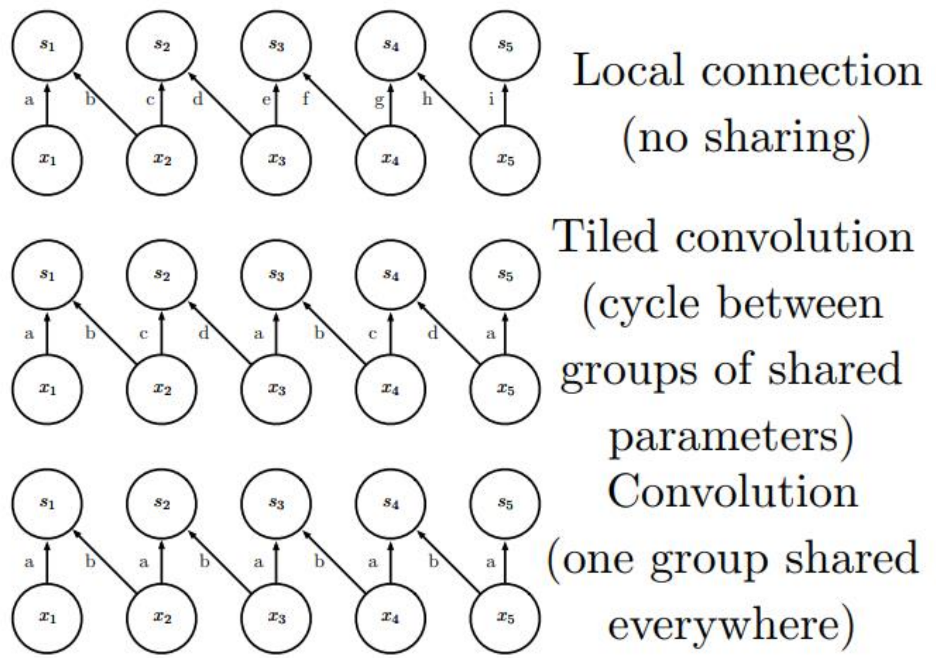
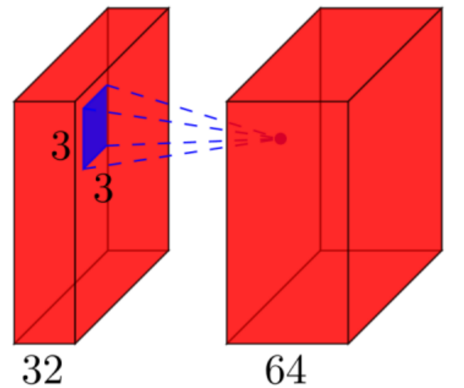


Figure 9.16

(Goodfellow 2016)

CNN – Number of Parameters

Consider a convolutional layer which takes *l* feature maps at the input, and has *k* feature maps as output. The filter size is *n x m*.



Above,

l=32 feature maps as input

k=64 feature maps as output

filter size is *n*=3 x *m*=3.

Note – filter size is actually 3x3x32, as our input has 32 dimensions.

And we learn 64 different 3x3x32 filters.

Thus, the total number of weights is *n*m*k*l*. Then, there is also a bias term for each feature map, so we have a total number of parameters of *(n*m*l+1)**k = (3*3*32 + 1) * 64 = 18,496

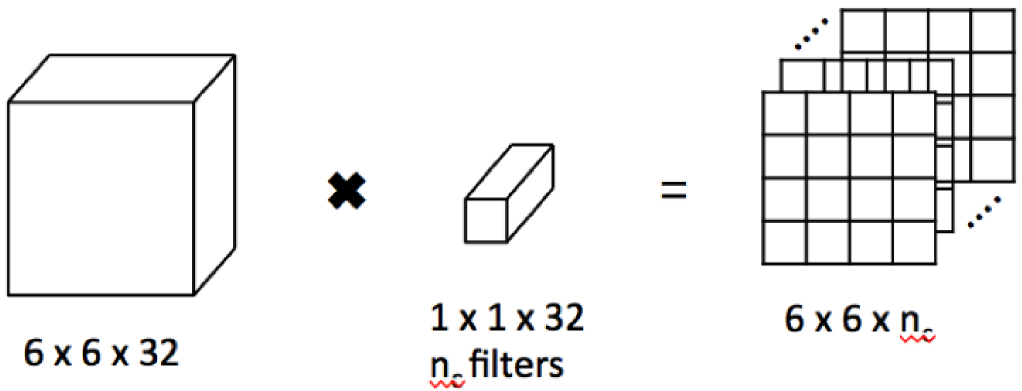
https://stackoverflow.com/questions/42786717/how-to-calculate-the-number-of-parameters-for-convolutional-neural-network

CNN 1x1 kernel size

Used for dimension reduction – for example, an image of 200 x 200 with 50 features on convolution with 20 filters of 1x1 would result in size of 200 x 200 x 20.

Think of them as coordinate dependent transformations in the filter space.

Equivalent to cross channel parametric pooling layer.

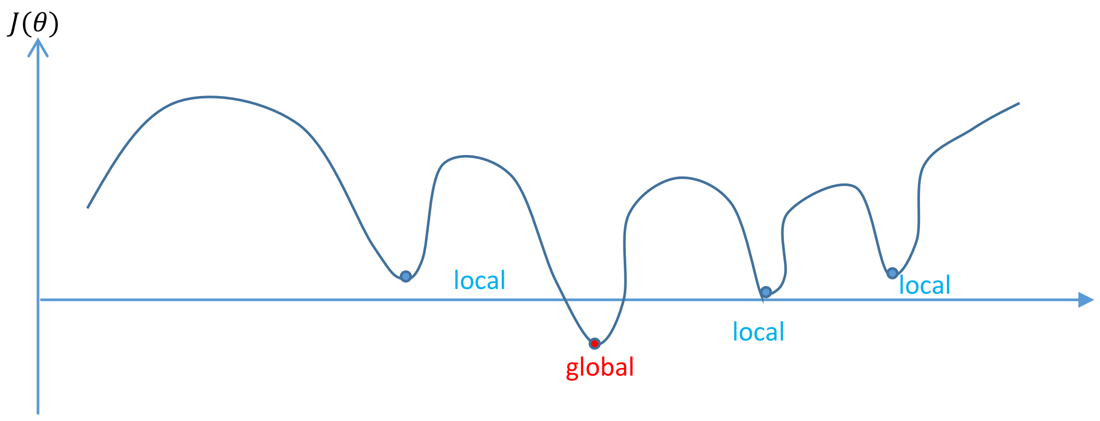


https://iamaaditya.github.io/2016/03/one-by-one-convolution/

https://stats.stackexchange.com/questions/194142/what-does-1x1-convolution-mean-in-a-neural-network

Is Local Minima a Problem for ANN?

1. The source of local minima is model non-identifiability.
2. However, for large deep learning networks the local minima are equivalent in terms of cost function.
3. For large networks, most local minima have low cost function in comparison to the global minima.
4. So it is ***not imperative*** to find the global minimum, but rather find a local minima that has low but not minimal cost.
5. A way to deal with local minima is to restart the network parameter search multiple times and allow the optimization to find a better local minima.



Regularization

"Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error."

Goodfellow, 2017

Example – model ensemble, a variance reduction technique

Early Stopping in Learning Curves

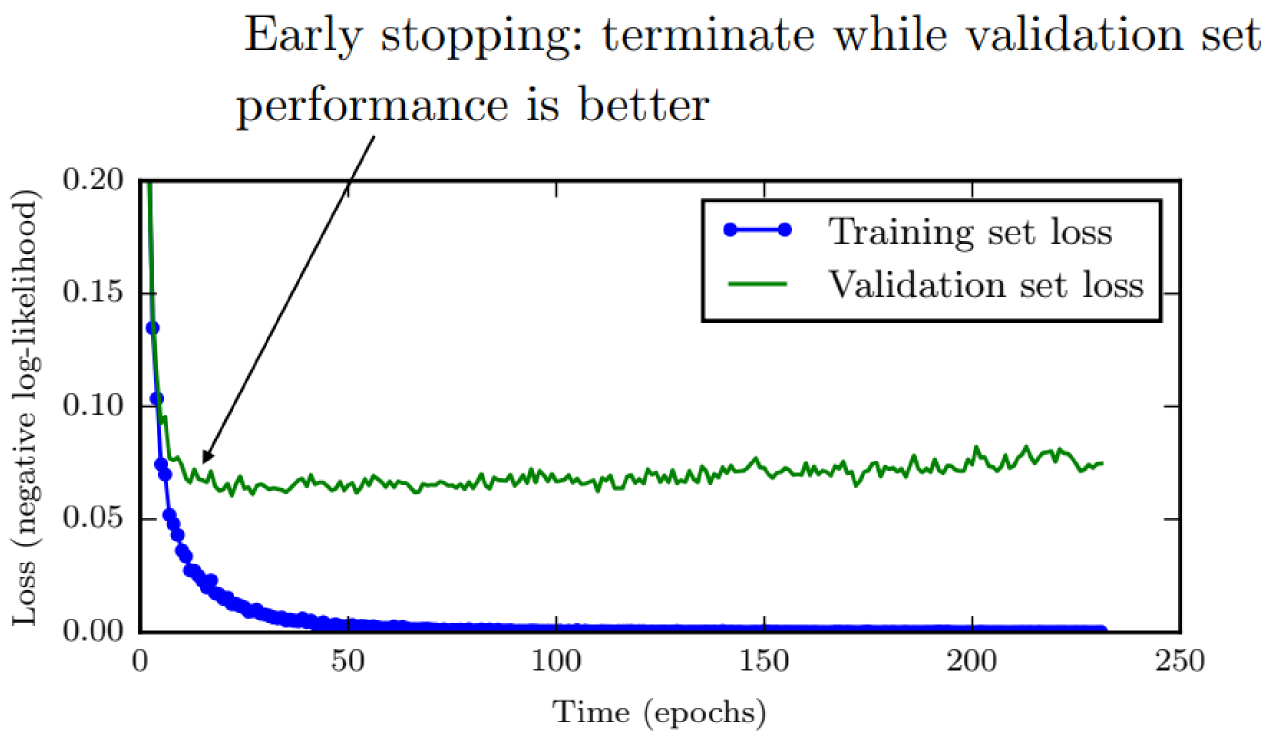
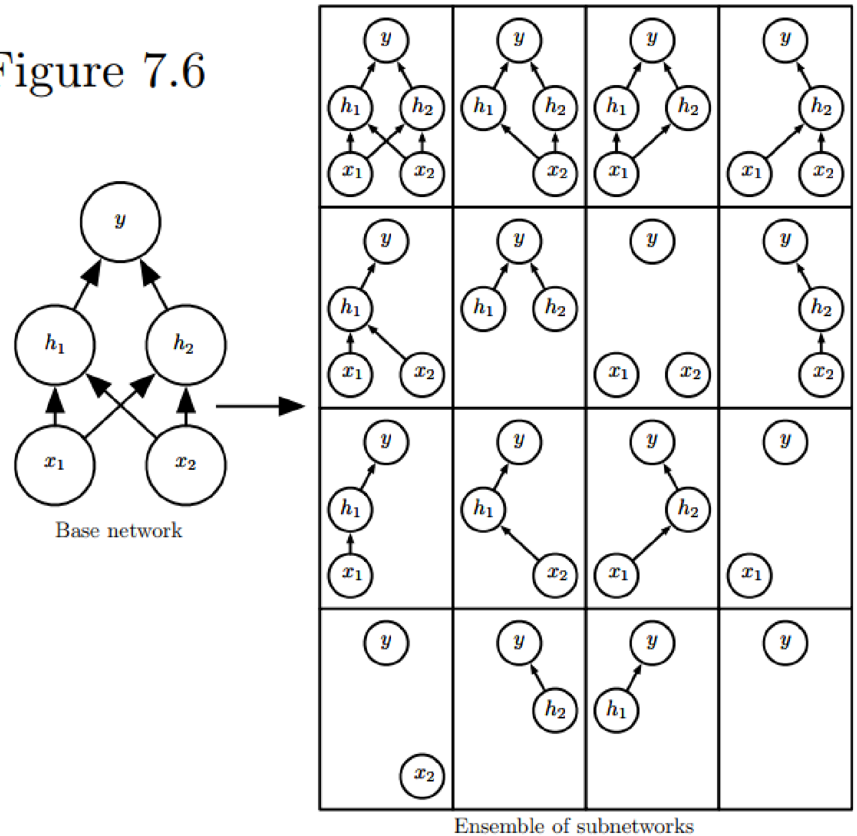


Figure 7.3

(Goodfellow 2016)

Convolution Neural Network (CNN) – Dropout

Figure 7.6



(Goodfellow 2016)

Dropout (different subsets of same parameters) not same as bagging (independent models + parameters).

Dropout

1. Done only during training where for each minibatch, randomly sample a different binary mask to apply to al input and hidden layers.

2. Effectively reduces the capacity of model, so need large network size and also a lot of labeled data.

3. Equivalent to bagging with parameter sharing.

4. Computationally cheap – not additional expense.

5. Noise is multiplicative – prevents large weights to make noise insignificant.

6. **Weight scaling inference rule** – if p is probability of a node being dropped, multiply the output by $\frac{1}{1-p}$ at training so that no change is necessary at validation and test (TensorFlow implementation*). Other alternative is to multiply each input weight by p for test.

* https://www.tensorflow.org/api_docs/python/tf/nn/dropout

Textbook Chapters and Assignment

- Materials covered available in book:
  - DL: Chapters 6, 7, 9
  - HML: Chapters 10 – 11, 14
  - DLP: Chapters 4 – 5

- http://neuralnetworksanddeeplearning.com/ - Chapters 3 and 4

- Code: https://github.com/ageron/handson-ml2