Machine Learning and Data Mining

NEURAL NETWORKS – PART 2

Lecturer: Mark Cieliebak

Based on material by Mark Cieliebak and Pascal Sager

Introduction

Learning Goals

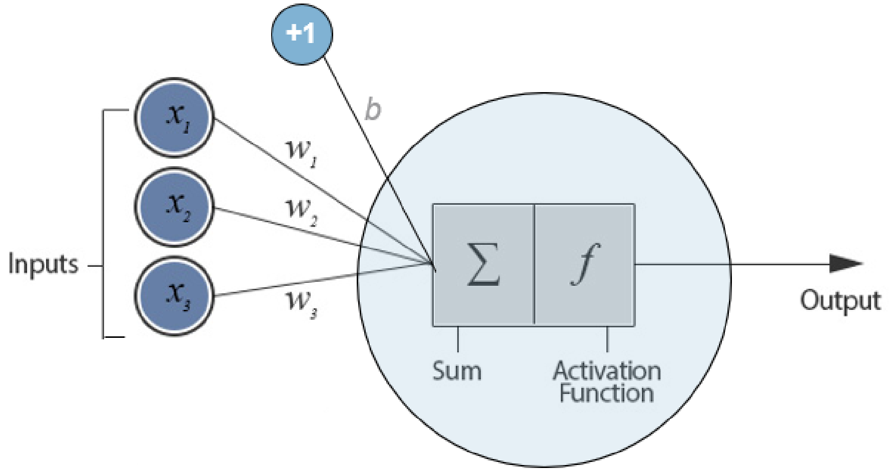
1. **Efficient Backpropagation is only possible if we re-use partial derivatives**

2. **Good initial weights help to improve training speed**

3. **Overfitting can be avoided with Dropout, Early Stopping or Data Augmentation**

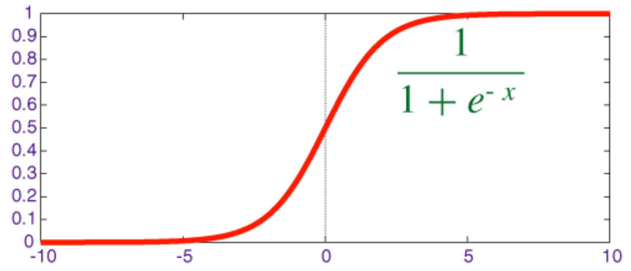
Recap: Neural Networks

Neurons

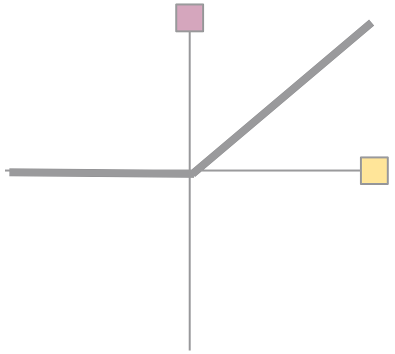


If **all components** in a neural network are **differentiable**, we can efficiently compute the gradient of the performance function.

Activation Functions

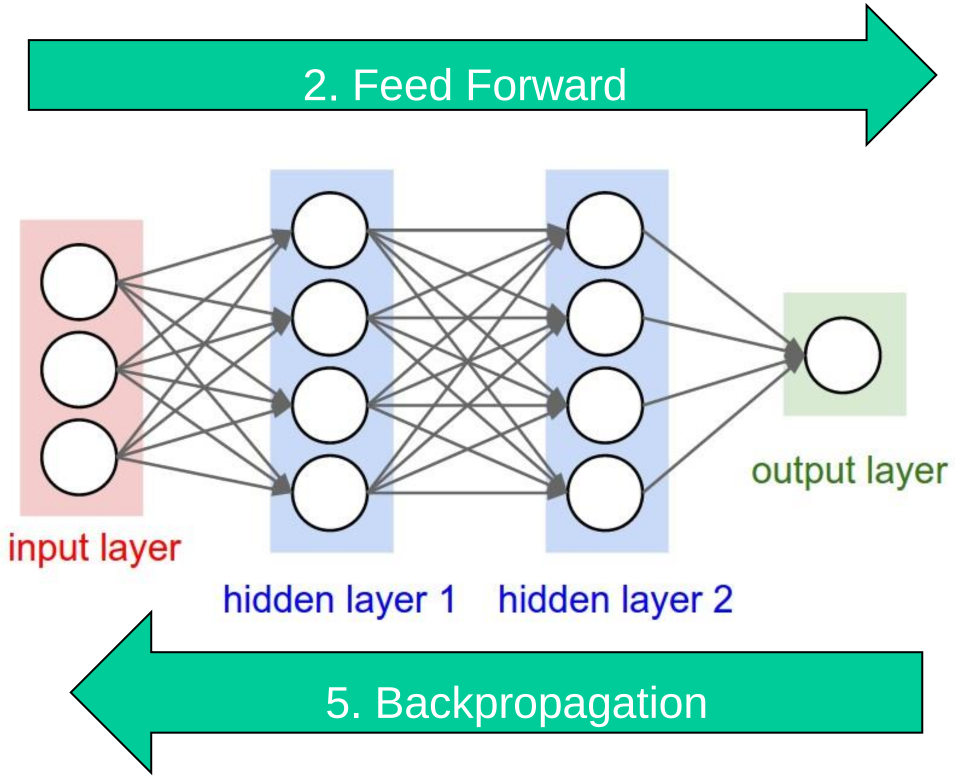


Sigmoid



RECTIFIED
LINEAR Unit
(ReLU)

Feed Forward Neural Networks



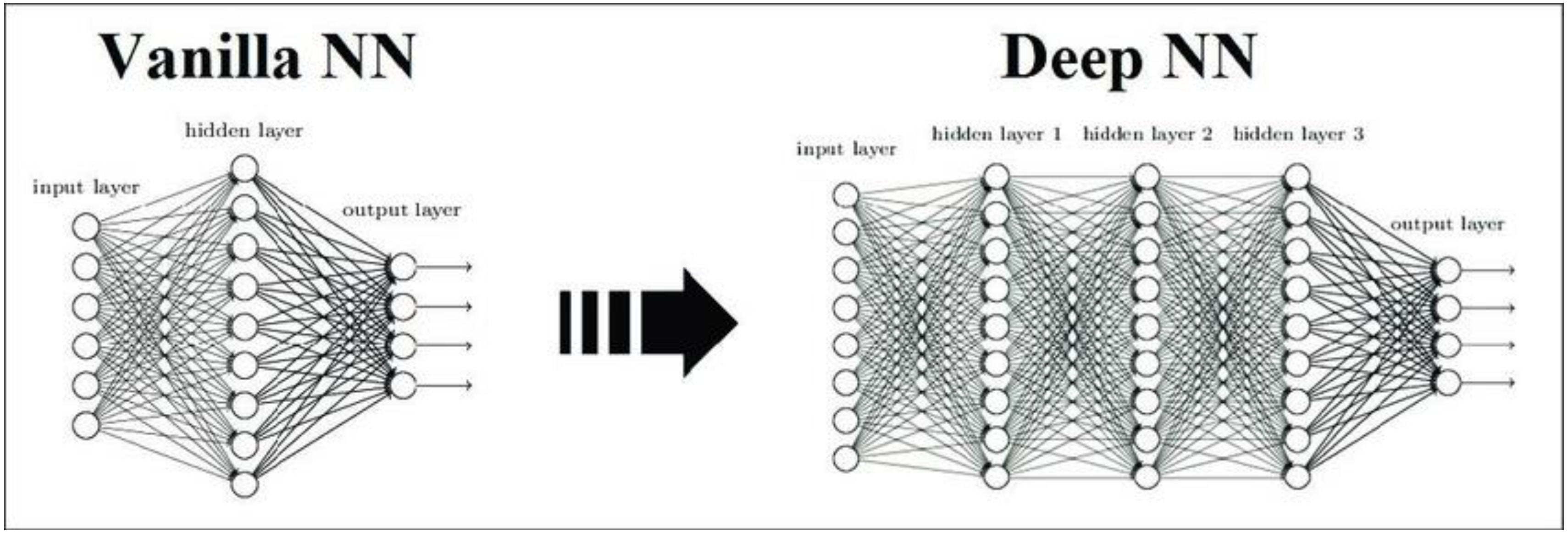
1. Training samples

3. Compare output to desired value

4. Compute gradient

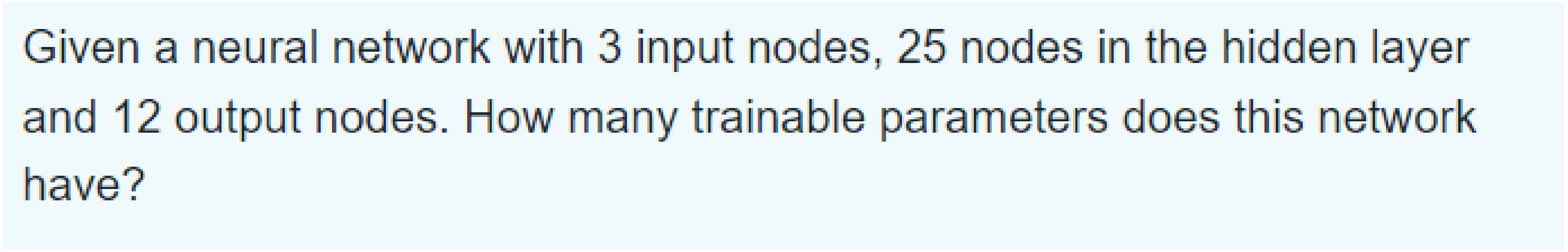
https://www.pyimagesearch.com/wp-content/uploads/2016/08/simple_neural_network_header.jpg

Deep Neural Networks



Deep Neural Networks are Neural Networks with more than one hidden layer.

Quiz



Given a neural network with 3 input nodes, 25 nodes in the hidden layer and 12 output nodes. How many trainable parameters does this network have?

Good Question

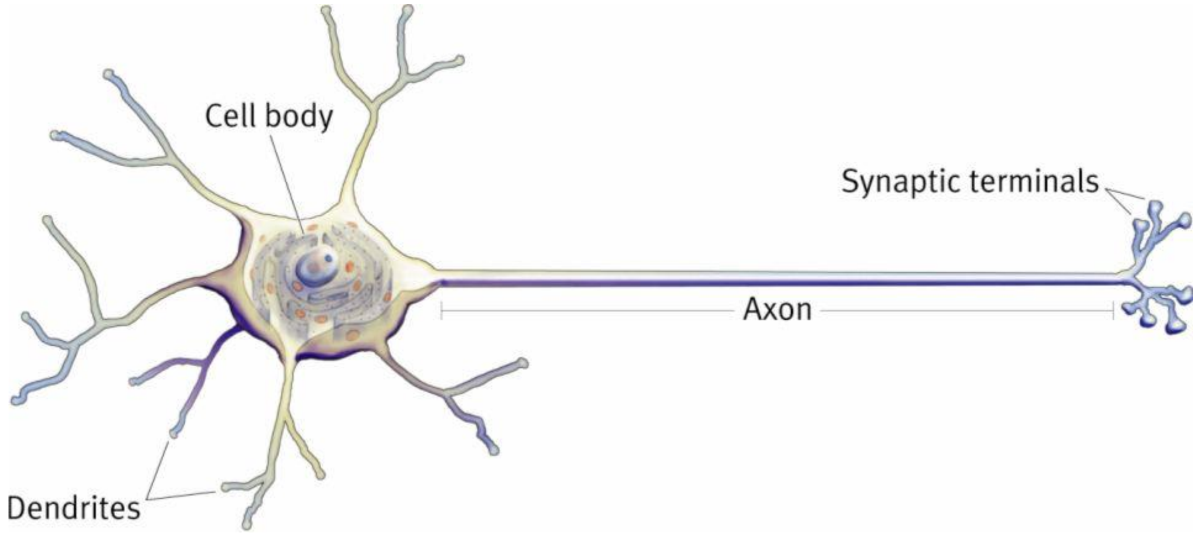
Gibt es ein Standardvorgehen bzgl. Festlegung der Anzahl Hidden-Layers und Hidden- Nodes?

Number of nodes in the hidden layers: How free are we, when setting the number of nodes in the hidden layers? How is the performance of a network with a high number of nodes, compared with a network that has a small number of nodes? Where is the tradeoff between performance and training effort.

How do we find the best activation function?

Neurons and the Brain

The Biological Neuron



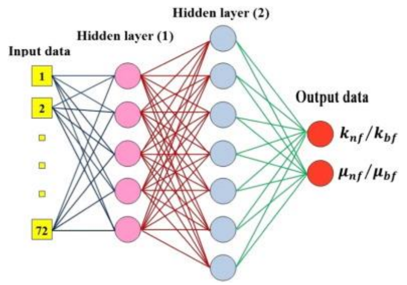
- **A neuron is connected to other neurons through up to 15'000 *synapses***
- **Once input exceeds a critical level, the neuron discharges a spike - an electrical pulse that travels from the body, down the axon, to the next neuron(s)**
- **The axon endings almost touch the dendrites or cell body of the next neuron.**

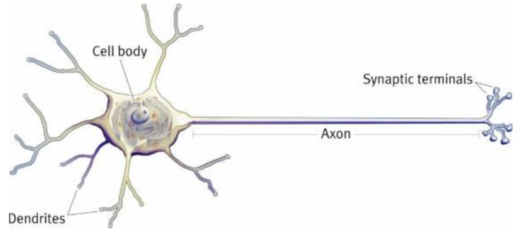
What is Artificial in "Artificial Neural Networks"?

There are two basic reasons why we are interested in building artificial neural networks (ANNs):

Technical viewpoint: Some problems such as character recognition or the prediction of future states of a system require massively parallel and adaptive processing.

Biological viewpoint: ANNs can be used to replicate and simulate components of the human (or animal) brain, thereby giving us insight into natural information processing.





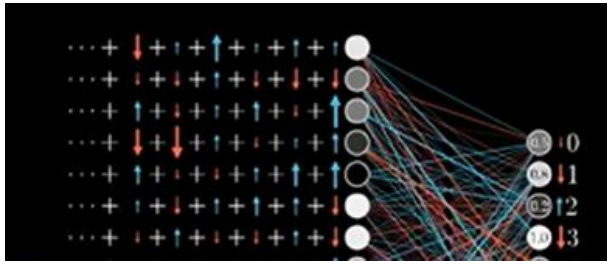
Good Question

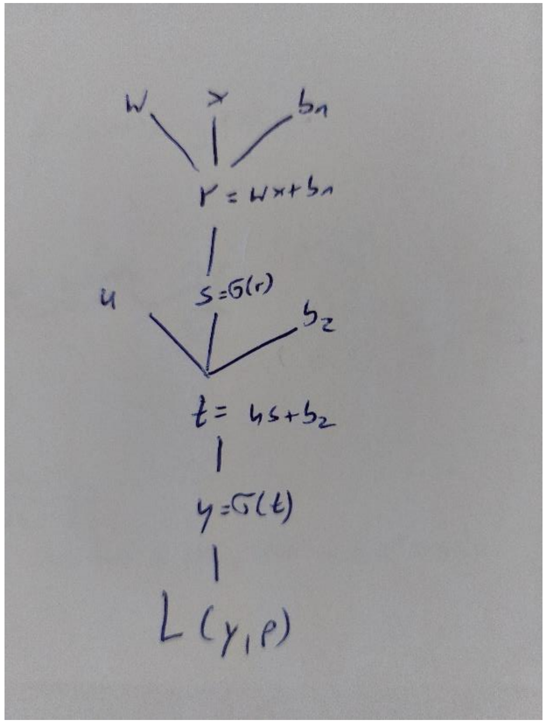
"Neurons that fire together, wire together"

Donald Hebb, Neuroscientist, 1949

Backpropagation

Backpropagation

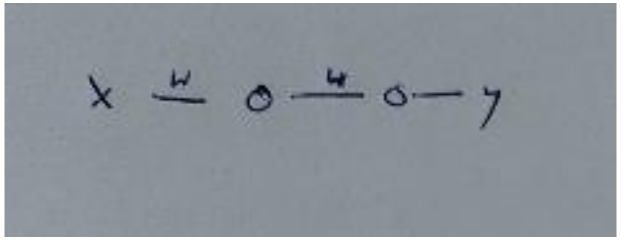




$$\frac{\partial L}{\partial u} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial t} \cdot \frac{\partial t}{\partial u} \cdot \text{[illegible]}$$

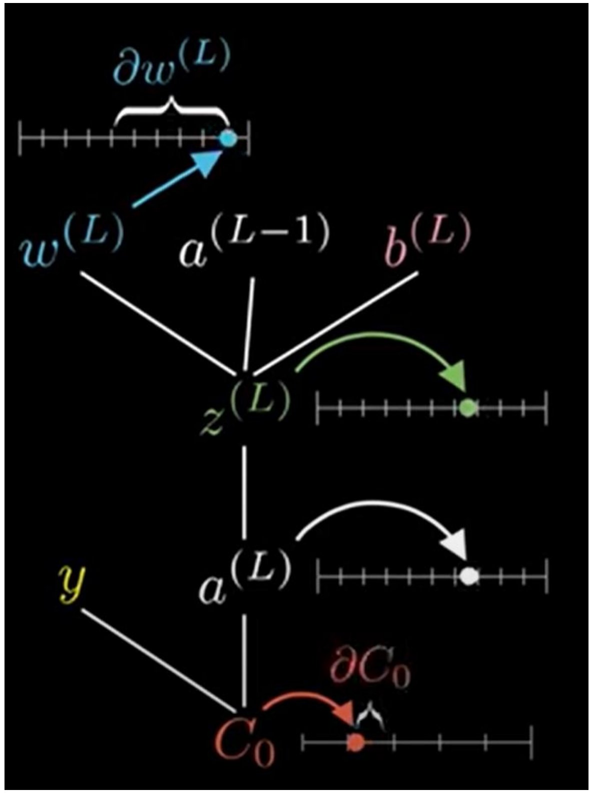
↯

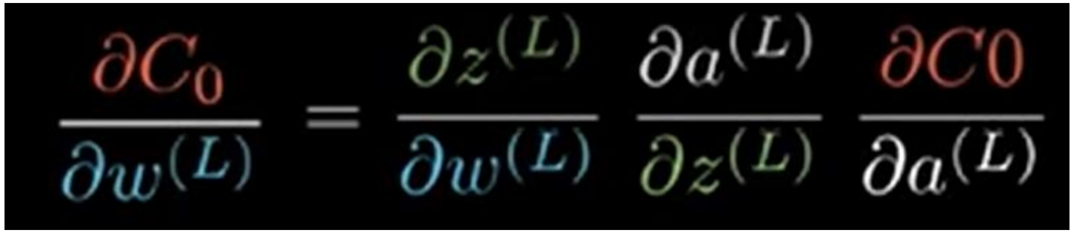
$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial t} \cdot \frac{\partial t}{\partial s} \cdot \frac{\partial s}{\partial r} \cdot \frac{\partial r}{\partial w}$$



Chain Rule for Partial Derivatives

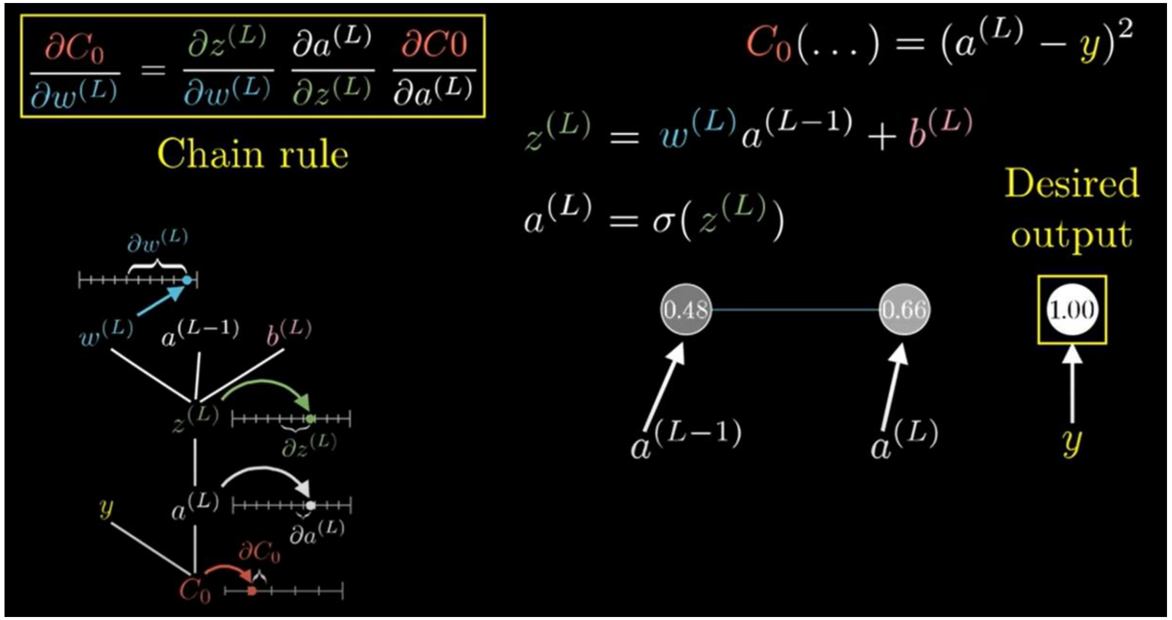
$$z = f(y) \quad y = g(x) \quad \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

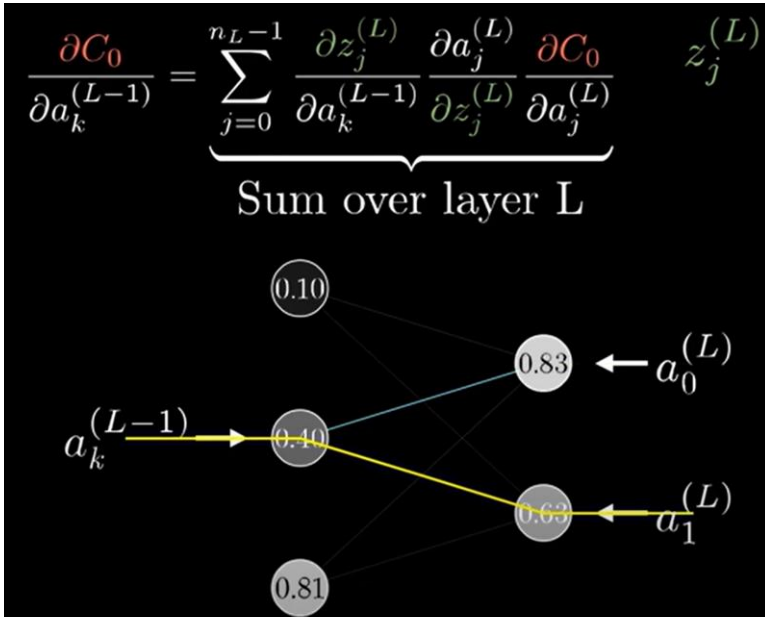




$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C0}{\partial a^{(L)}}$$

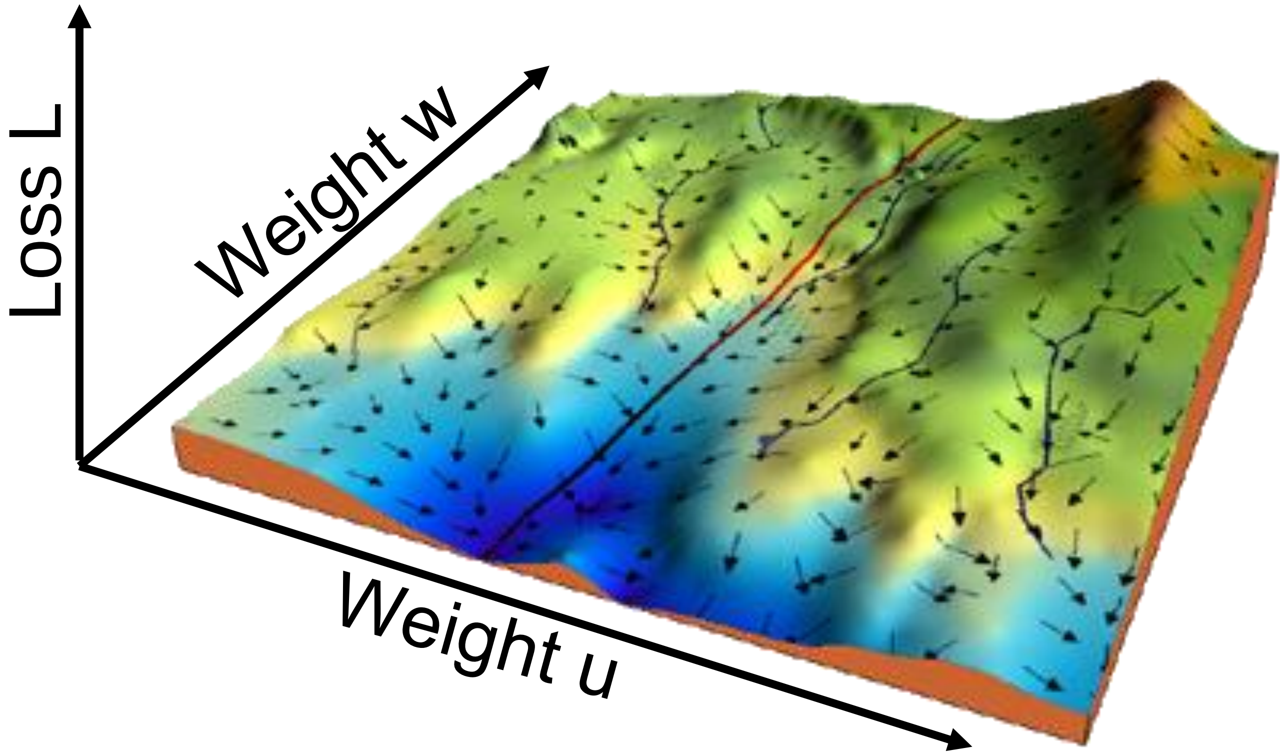
3Blue1Brown Backpropagation


$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C0}{\partial a^{(L)}}$
Chain rule
$C_0(\dots) = (a^{(L)} - y)^2$
$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$
$a^{(L)} = \sigma(z^{(L)})$
Desired output
0.48
0.66
1.00
$a^{(L-1)}$
$a^{(L)}$
y
$\partial w^{(L)}$
$w^{(L)}$
$a^{(L-1)}$
$b^{(L)}$
$z^{(L)}$
$\partial z^{(L)}$
y
$a^{(L)}$
$\partial a^{(L)}$
∂C_0
C_0


$\frac{\partial C_0}{\partial a_k^{(L-1)}} = \underbrace{\sum_{j=0}^{n_L - 1} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}}}_{\text{Sum over layer L}}$
$z_j^{(L)}$
0.10
0.40
0.81
0.83
0.63
$a_k^{(L-1)}$
$a_0^{(L)}$
$a_1^{(L)}$

Good Question

It's really hard to grasp how the derivates should update the weights and biases and somehow arrive at a good solution.



Small black arrows: direction of gradient

Good Question

Hat die Chain-Rule nur einen Einfluss auf das Ändern der Weights oder auch von den Biases?

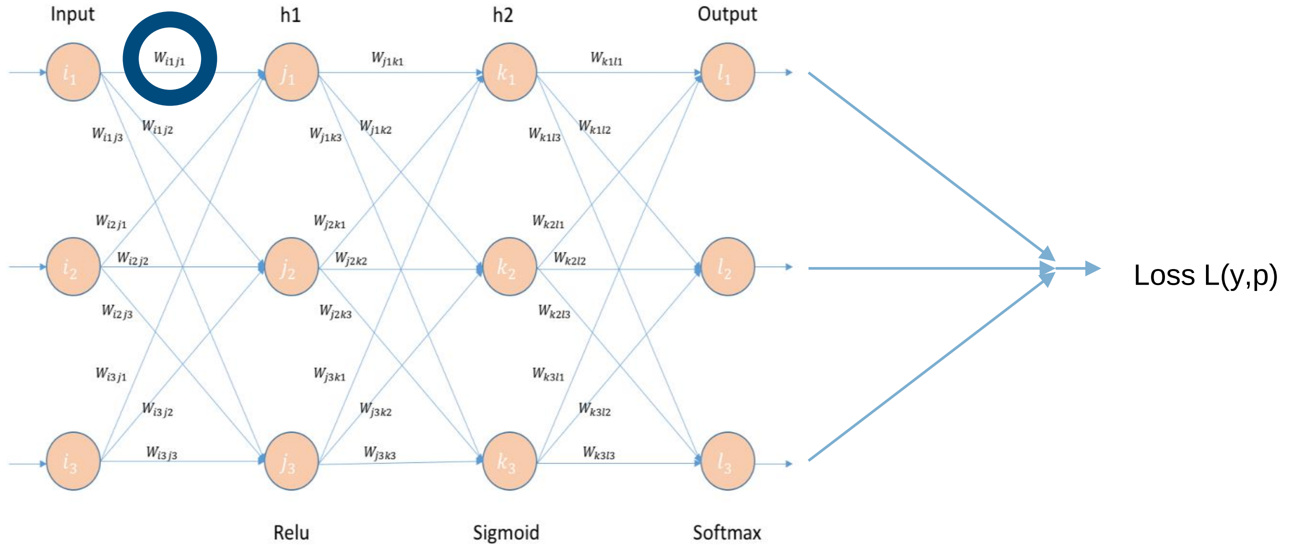
Good Question

Do we really need to calculate the derivatives with these long ass chain rules?

Or if we understand the math behind it, that's enough for the SEP

QUESTION: Number of Paths

How many paths are there to consider when computing $\frac{\partial L}{\partial w_{i1j1}}$



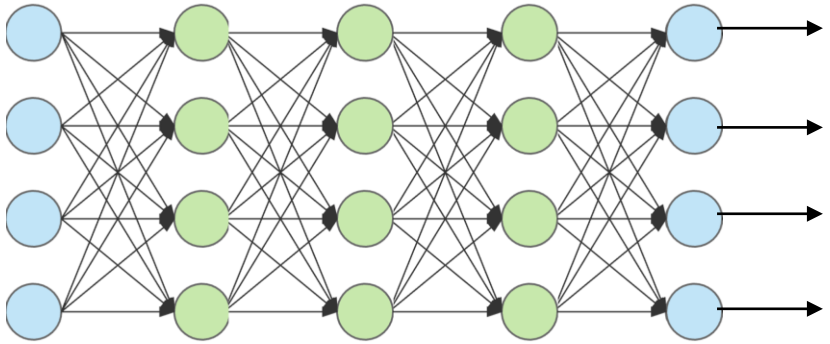
TASK: Size of Neural Network

Assume a neural network with *k* hidden layers, where all layers (incl. input and output layer) each have *w* neurons.

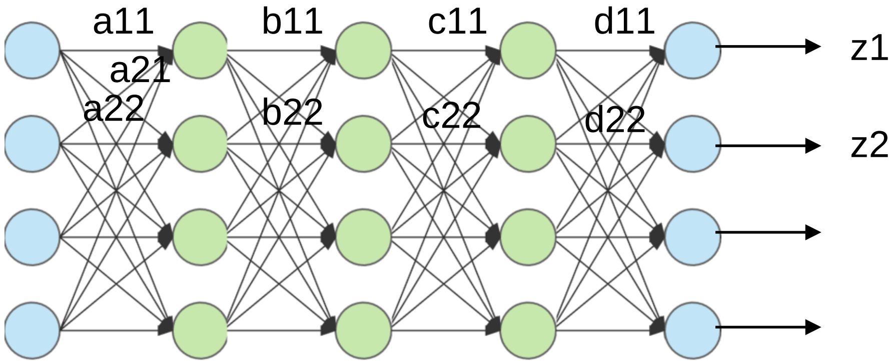
In the example: k = 3 hidden layers, width = 4.

1. How many weights does such a network have?

2. How many paths do exist that start in any input neuron and end in any output neuron?



Re-use of Partial Derivative Chains



$$\frac{\partial\, z1}{\partial a11} = \frac{\partial\, z1}{\partial d11}\,\frac{\partial\, d11}{\partial c11}\,\frac{\partial\, c11}{\partial b11}\,\frac{\partial\, b11}{\partial a11} + \ldots$$

$$\frac{\partial\, z1}{\partial a21} = \cdots + \boxed{\frac{\partial\, z1}{\partial d11}\,\frac{\partial\, d11}{\partial c11}\,\frac{\partial\, c11}{\partial b11}}\,\frac{\partial\, b11}{\partial a21}\ \ldots$$

$$\frac{\partial\, z2}{\partial a11} = \ldots + \frac{\partial\, z22}{\partial d12}\,\frac{\partial\, d12}{\partial c11}\,\boxed{\frac{\partial\, c11}{\partial b11}\,\frac{\partial\, b11}{\partial a11}}\ \ldots$$

Backpropagation

- First described by Werbos 1974
- Re-discovered by Parker 1982 and **Rumelhart, Hinton, Williams in 1986**

Core Idea: Re-use partial derivatives to compute the gradients efficiently

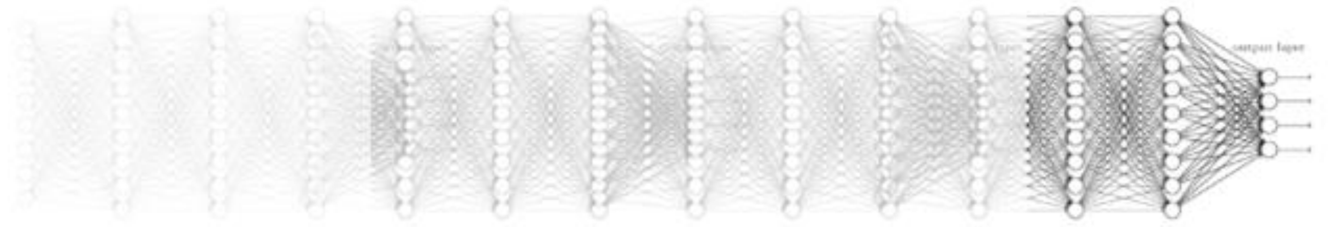
David Rumelhart

Geoffrey Hinton

Ronald J. Williams

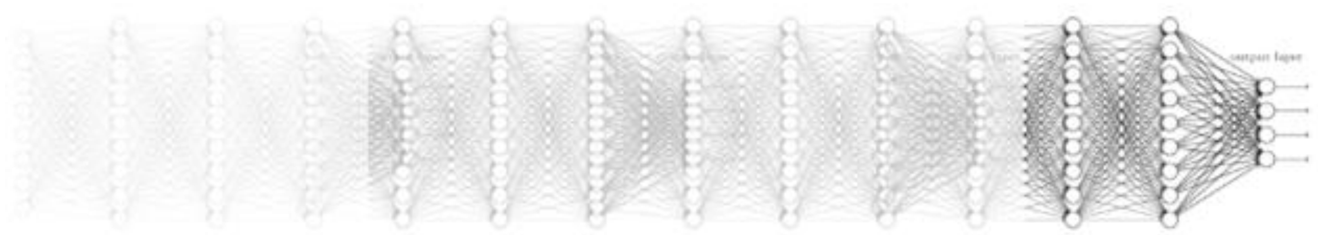
Disadvantages of Backpropagation

- Requires large amount of labeled training data
- Learning time is slow for multiple hidden layers
- Can get stuck in local optima
- Vanishing Gradient Problem

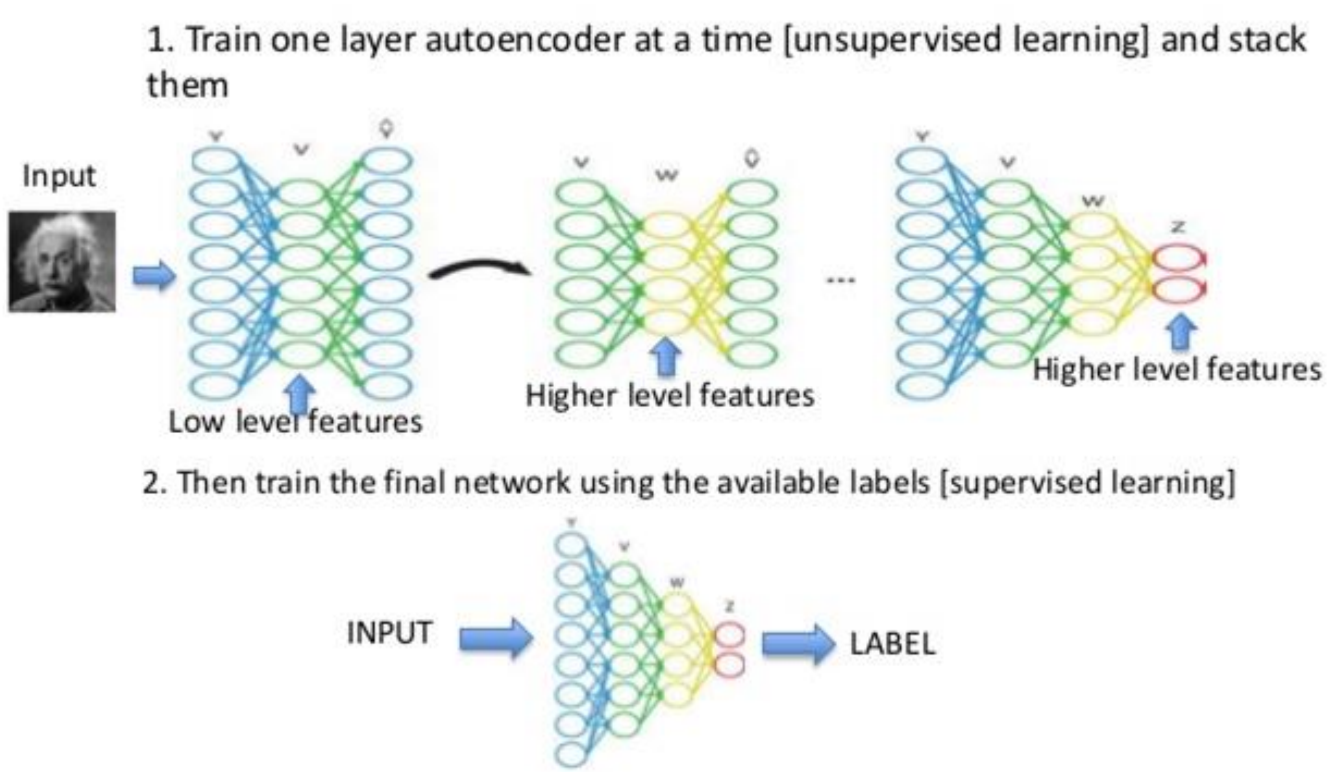


Vanishing Gradient Problem
Sepp Hochreiter 1991

- **Observation:** if we have many hidden layers, and if all partial derivatives are between -1 and 1, then multiplying them makes them exponentially small
- Thus, changes in the first layers will diminish since the gradient becomes extremely small
- Depths of a network can be easily up to 150 layers
- In practice, this results in very slowly changing weights, thus, very long training times
- **Solution:** do not use randomly initialized weights, but rather "carefully pre-trainied weights" (see next slides)



Stacked Auto-Encoders


1. Train one layer autoencoder at a time [unsupervised learning] and stack them
Input
Low level features
Higher level features
Higher level features
2. Then train the final network using the available labels [supervised learning]
INPUT
LABEL

Geoffrey Hinton

Yoshua Bengio

https://image.slidesharecdn.com/introductiontodeeplearning-160507133124/95/deep-learning-towards-general-artificial-intelligence-22-638.jpg?cb=1462627908

Resources

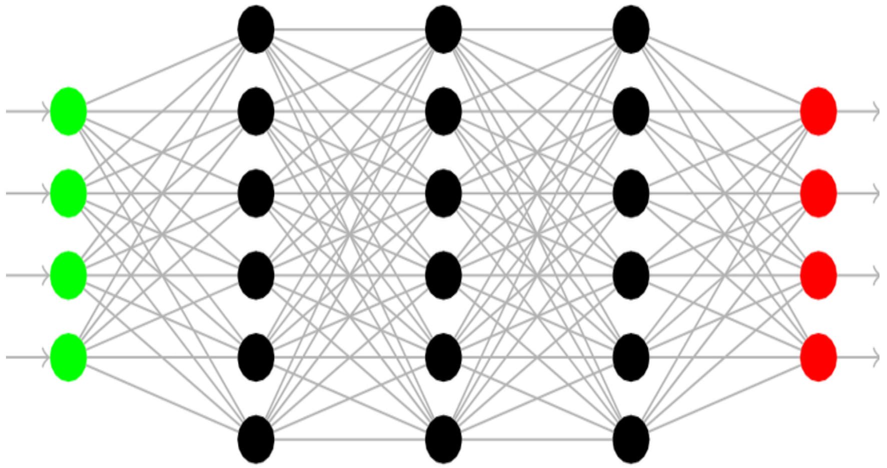
Slides are based on:
- Jun Wang, "Deep Learning" 2016. https://www.slideshare.net/JunWang5/deep-learning-61493694/10
- Patrick Winston, "Neural Nets". Video: https://www.youtube.com/watch?v=uXt8qF2Zzfo&t=2095s
- Oliver Dürr, "Machine Intelligence: Deep Learning", 2017. MAS Data Science, ZHAW
- Ahmad Aljebaly: "Artificial Neural Networks"

Scientific Papers:
- W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4):115–133, 1943.
- F. Rosenblatt. The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory, 1957.
- D. E. Rumelhart, G. E. Hinton, R. J. Williams: Learning Representation by Back-Propagating Errors. Nature, 1986. http://www.cs.toronto.edu/~hinton/absps/naturebp.pdf
- S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut f. Informatik, Technische Univ. Munich, 1991.
- Kurt Hornik: Approximation Capabilities of MultiLayer FeedForward Networks, 1991
- G. E. Hinton, S. Osindero, Y.W. Teh: A fast learning algorithm for deep belief networks, 2006.
- Y. Bengio, P. Lamblin, P. Popovici, H. Larochelle: Greedy layer-wise training of deep networks, 2007.

Blog Posts:
- http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html
- https://machinelearningmastery.com/inspirational-applications-deep-learning/

Softmax

Softmax



1.3
-1
3.2
0.5

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

0.121
0.012
0.812
0.055

Goal: For classification tasks with mutually exclusive classes, in the output layer we want to have 1.0 for the correct class, and 0 for all other classes

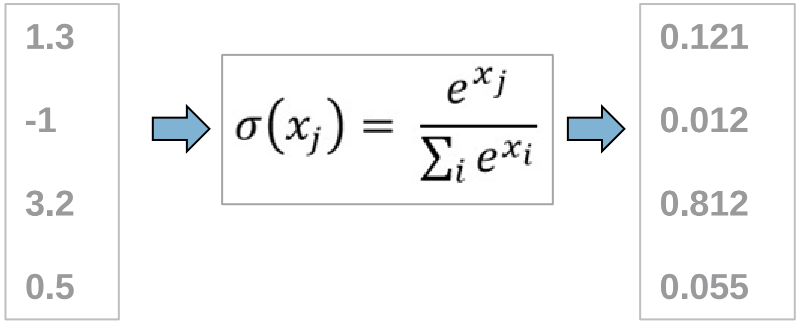
Softmax

Goal: For classification tasks with mutually exclusive classes, in the output layer we want to have 1.0 for the correct class, and 0 for all other classes

Idea: A "Max-Layer" could set 1 for maximum value of previous layer, and 0 to all other outputs.
Problem: this is not differentiable

Solution: Softmax function

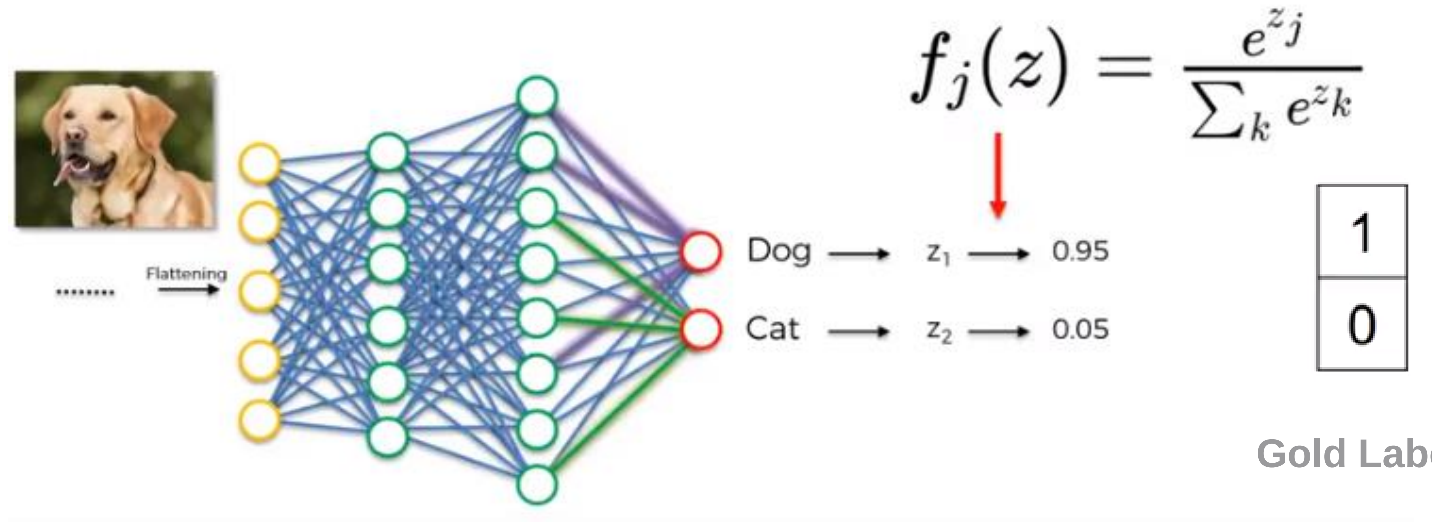
- Scales all values between 0..1
- Sum of all values is 1, like a probability distribution
- Maximum value of previous layer will be "large" in comparison to other values



QUESTION

Where was a formula similar to Softmax used?

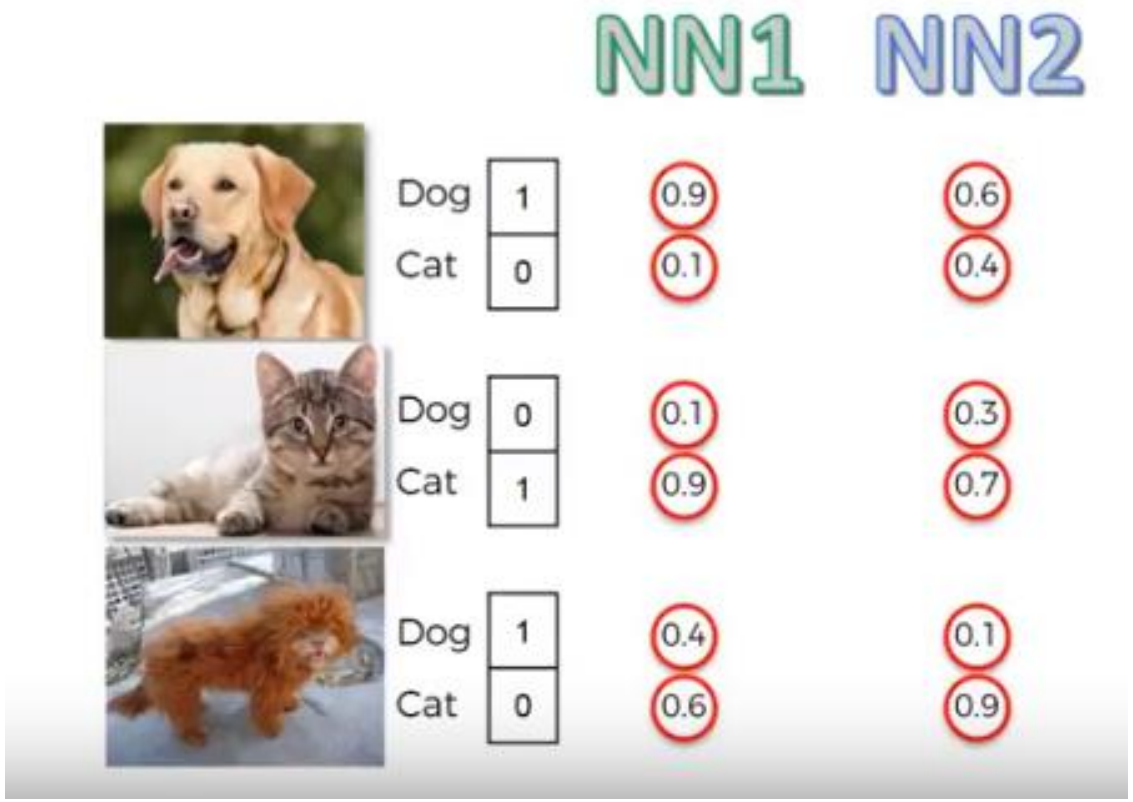
Softmax



Gold Label

Measuring Classification Error: Cross-Entropy Loss

How to Measure the Error



Goal: For a multi-class classification problem, measure how good the actual output (a probability distribution over the classes) resembles the desired result (a one-hot bit vector).

	NN1	NN2
Classification Error	0.33	0.33
Mean Squared Error	0.25	0.71

Cross Entropy Loss

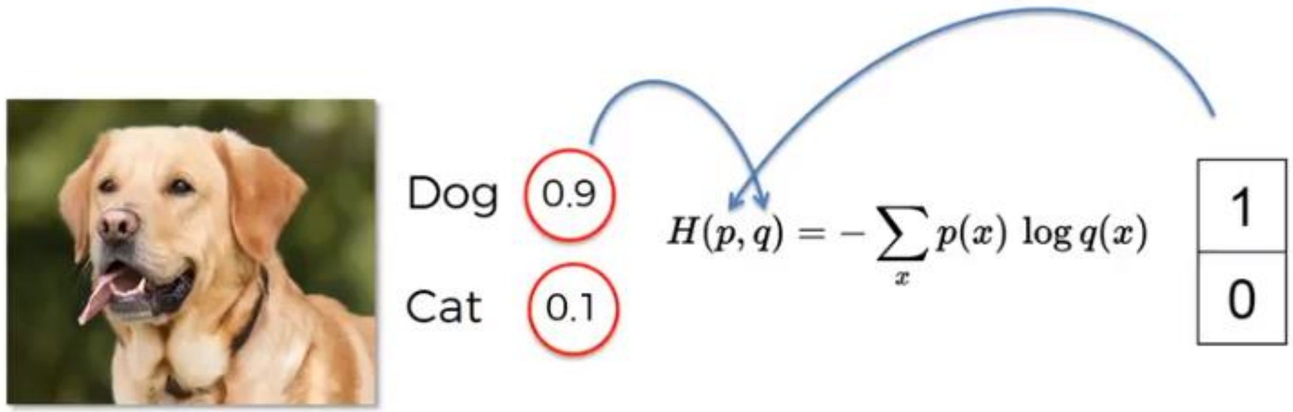
Given a sample {x, y}, where x is input and $y = (y_1, \dots y_r)$ is a distribution of expected output values for r classes, i.e. each value $y_i \in [0,1]$ and $\sum_{i=1}^{r} y_i = 1$.

For a Classifier that outputs a distribution $\hat{y} = (\hat{y}_1, \dots \hat{y}_r)$, with $\hat{y}_i \in [0,1]$ and $\sum_{i=1}^{r} \hat{y}_i = 1$, the **Cross Entropy Loss** is defined as follows:

$$\mathrm{H}(y, \hat{y}) = -\sum_{i=1}^{r} y_i \log \frac{1}{\hat{y}_i}$$

Further reading: https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/

Cross Entropy Loss



Remarks:

- Cross Entropy is assymetric
- In machine learning, base *e* instead of base 2 is often used

Rule of Thumb: use MSE for regression, and Cross-Entropy for multiclass-classification problems

Further reading: https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/

Optimizing a Neural Network

Good Question

How do we find the proper architecture?

In Case of Bad Performance:

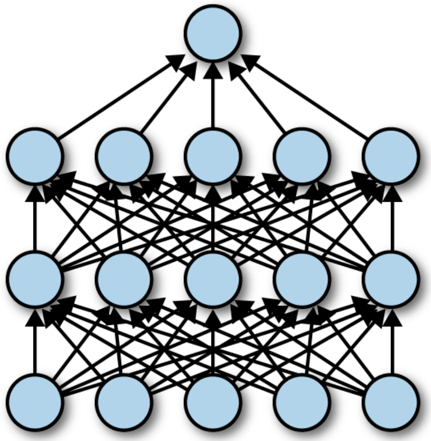
Analyze Learning Curves and try to change...

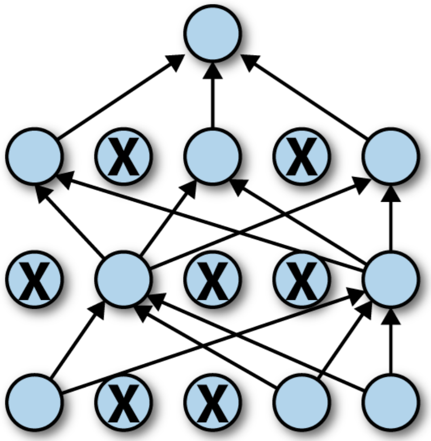
- The number of training samples
- The number of hidden neurons
- The activation functions
- Regularization
- Learning rate / Learning rate decay
- The batch size
- The optimization algorithm (Adadelta, Adam etc.)
- Number of epochs

- Early Stopping
- Dropout
- Data Augmentation

Dropout Regularization

Dropout:
Skip some nodes randomly during Training


(a) Standard Neural Net



(b) After applying dropout

Dropout Regularization

Goal: Prevent over-fitting through better generalization

Idea: ignore (zero out) a random fraction of nodes and corresponding activations during training.

Remarks:

- Introduced by Srivastava et al. 2014
- Dropout removes nodes during TRAINING, but uses ALL nodes during Testing
- Scale activation function during testing due to more input nodes
- It forces the network to learn redundant features: if a neuron is dropped out, then other neurons will have to step in and handle the representation required to make predictions for the missing neuron
- Rule of thumb: typical dropout rates are 20-50%
- Experience says: Prefer larger network with dropout than smaller networks

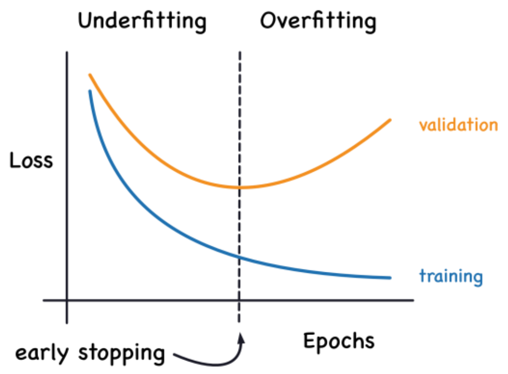
Further Reading: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

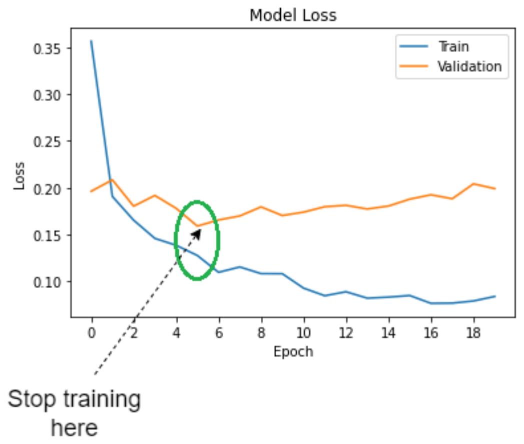
https://www.safaribooksonline.com/library/view/tensorflow-for-deep/9781491980446/assets/tfdl_0408.png

Early Stopping

Early Stopping can reduce risk of overfitting

- Train a network for large number of epochs
- Introduce checkpoints at regular intervals, e.g. after each or after 10 epochs (depends on size of training data)
- Evaluate loss or other quality measures for training data and validation data for each checkpoint
- Stop if validation scores get worse and revert to "best" checkpoint with optimum validation score

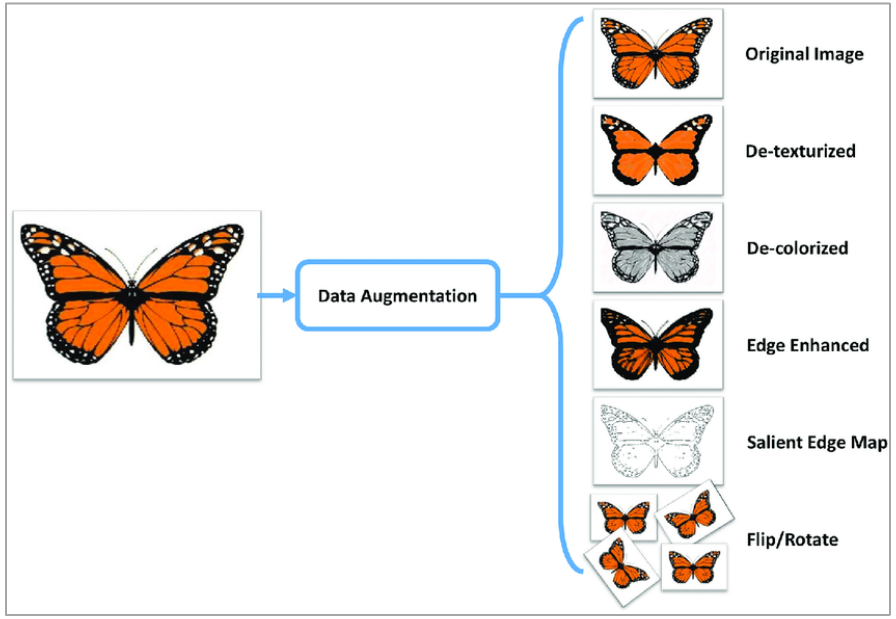




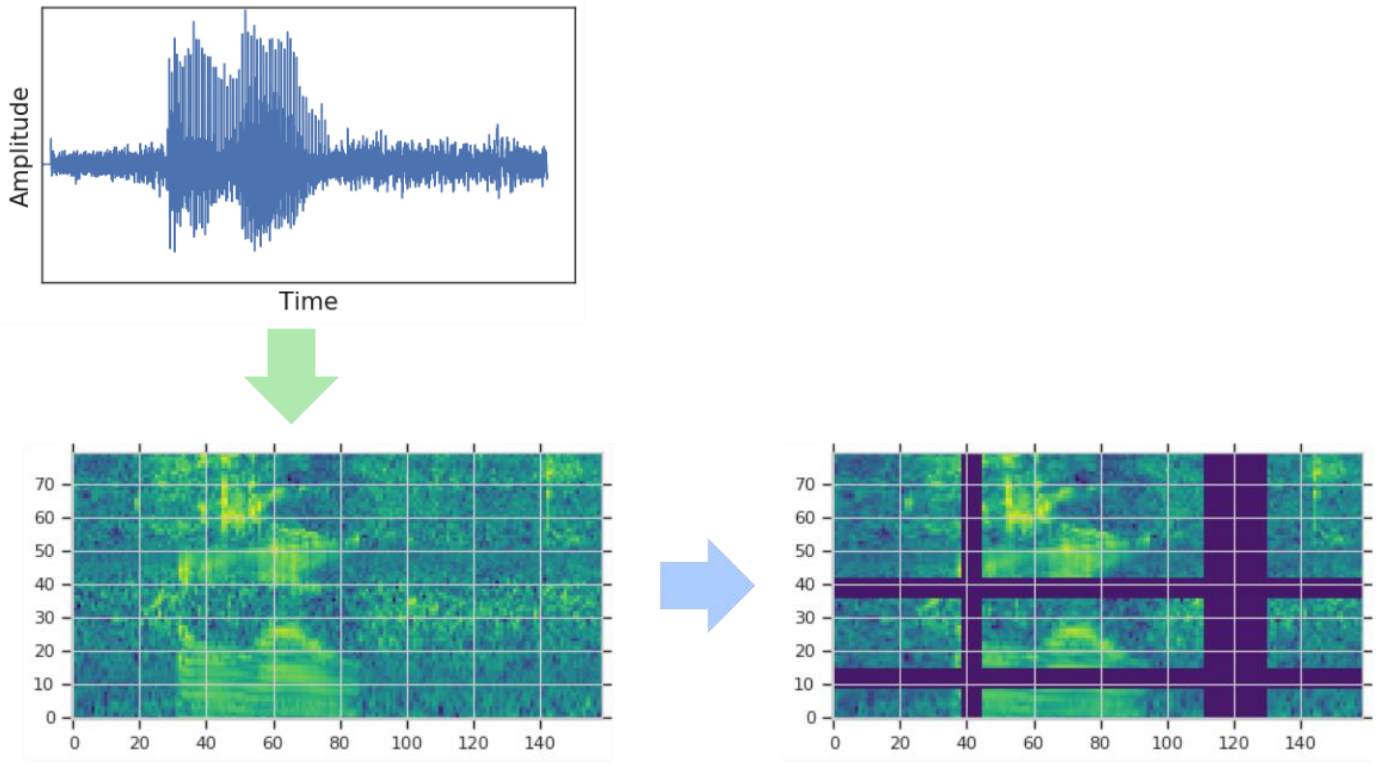
Data Augmentation

Data Augmentation increases the amount of training data

- Add random noise
- Combine or extrapolate traning samples
- Modify training samples

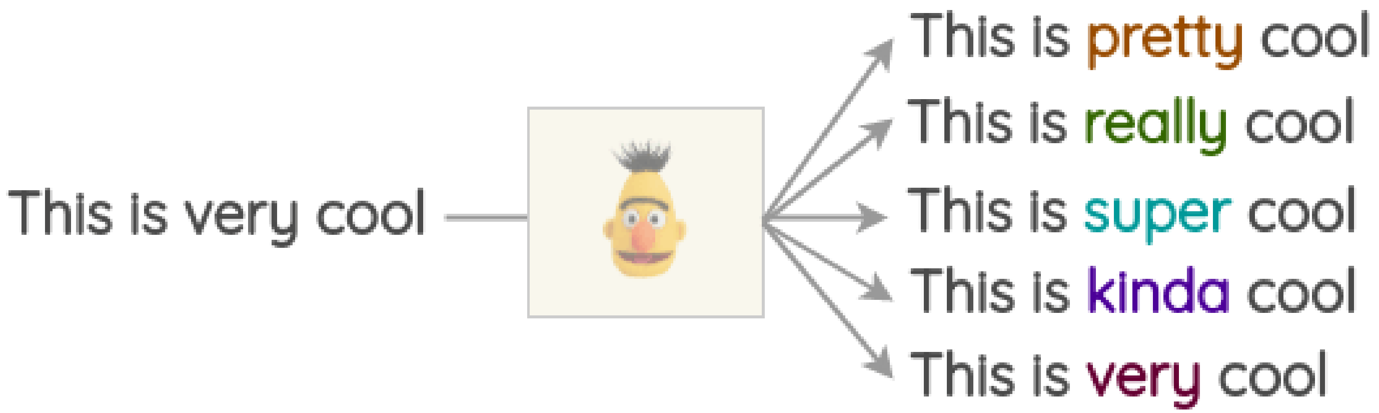


Data Augmentation for Audio



SpecAugment: https://arxiv.org/abs/1904.08779

Data Augmentation for Text



I have a new ca$\overset{r}{\cancel{t}}$

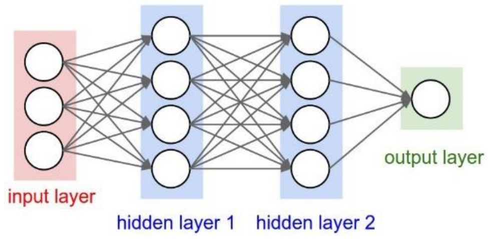
Network Topologies

Good Question

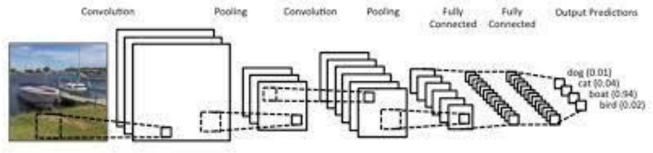
Is it possible that there are ways with more and less layers in one network?
Meaning for example after the second layer, one activation triggers the output node whilst the activation of another node triggers a third layer and there one new node."

Architectures for Neural Networks

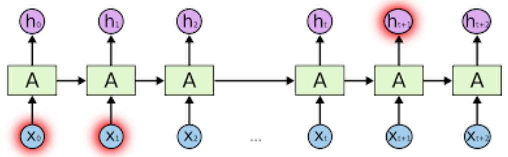
Vanilla Neural Networks



Convolutional Neural Networks



Recurrent Neural Networks



Transformers

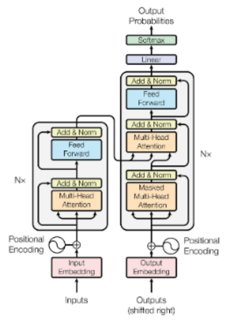
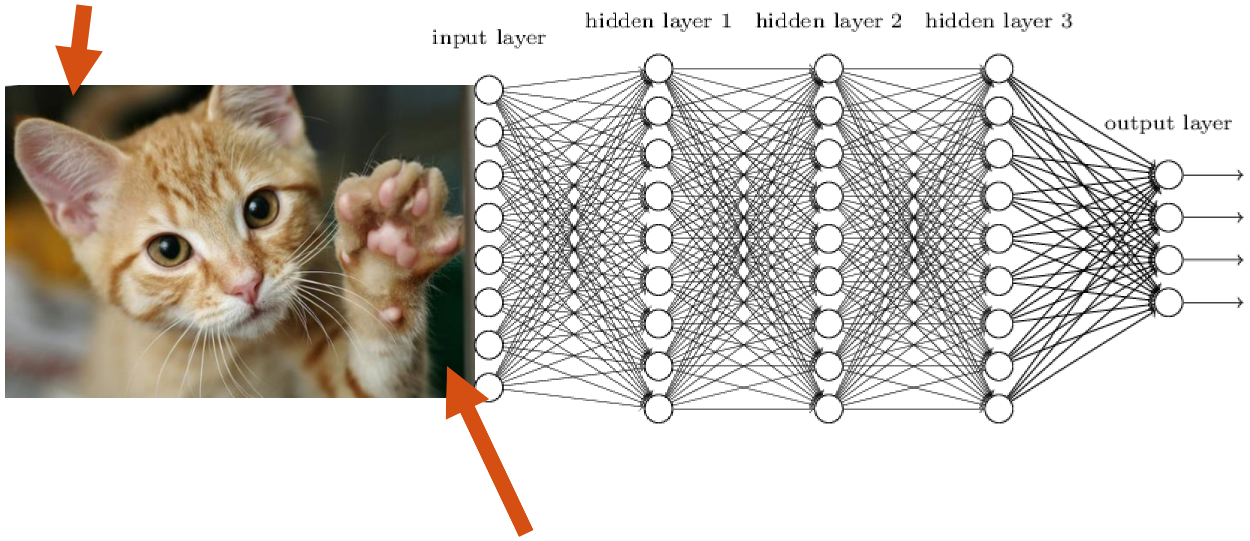
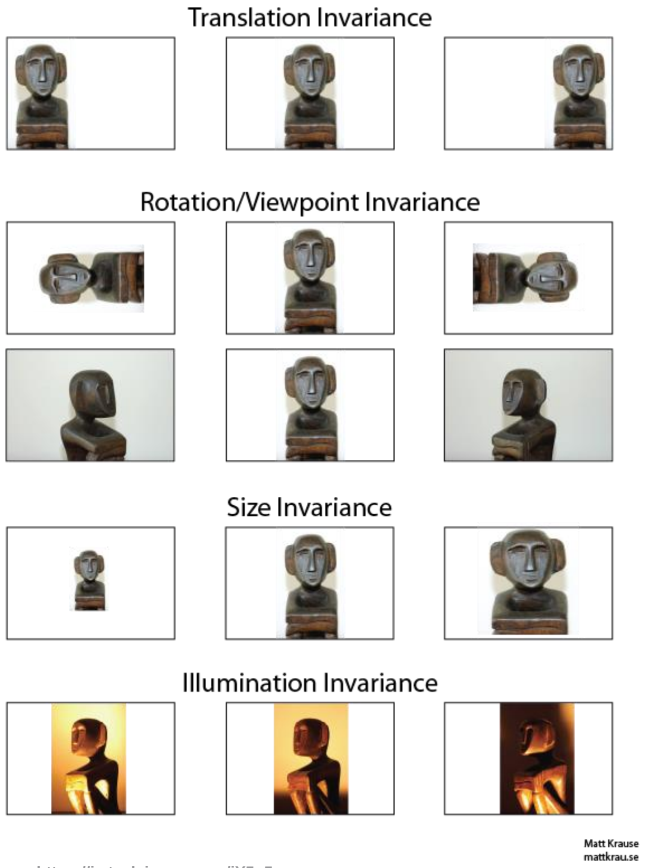


Figure 1: The Transformer - model architecture.

What's wrong with Fully Connected Networks?



What's wrong with Fully Connected Networks?



https://i.stack.imgur.com/iY5n5.png

Closing

Preparation for Upcoming Lecture

Watch Video on Convolutional Neural Networks (CNN's):

https://www.youtube.com/watch?v=FmpDIaiMIeA

Solutions

SOLUTION: Size of Neural Network

Assume a neural network with *k* hidden layers, where all layers (incl. input and output layer) each have *w* neurons.

In the example: k = 3 hidden layers, width = 4.

1. How many weights does such a network have?
Number of weights = w^2 (k+1)

2. How many paths do exist that start in any input neuron and end in any output neuron?
Number of paths = w^{k+2}