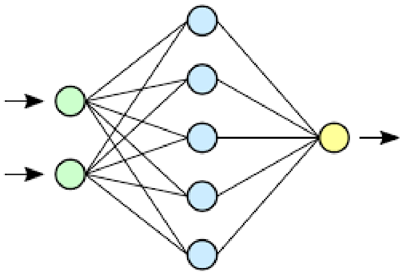
Deep Learning



Erik Suer

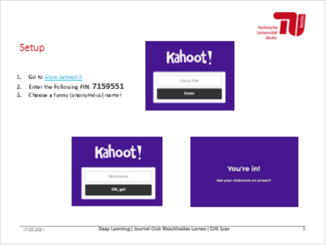
Journal Club Maschinelles Lernen

Summer Term 2021

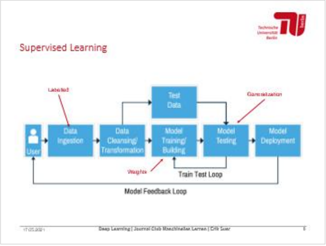
https://github.com/Erikx3/DeepLearningIntro

Roadmap

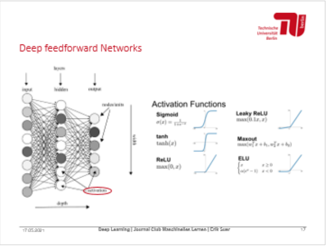
Introduction



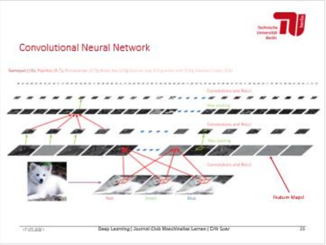
Supervised Learning, Perceptron, SGD



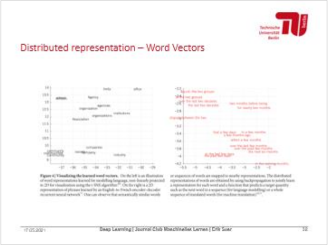
Neural networks & Backpropagation



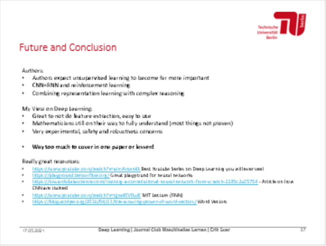
Convolutional neural networks – Image Processing



Recurrent neural networks – Language Processing

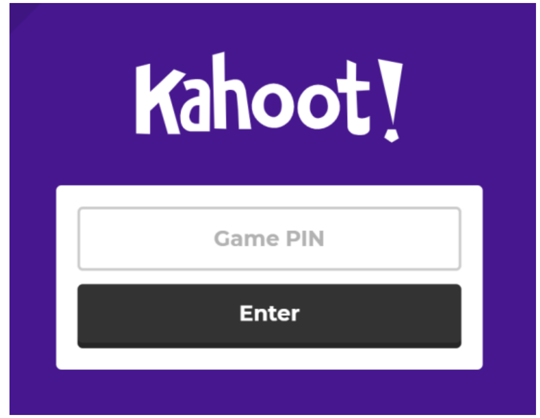


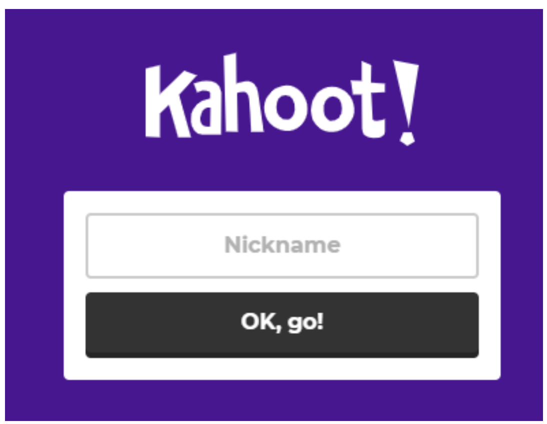
Conclusion and future of deep learning

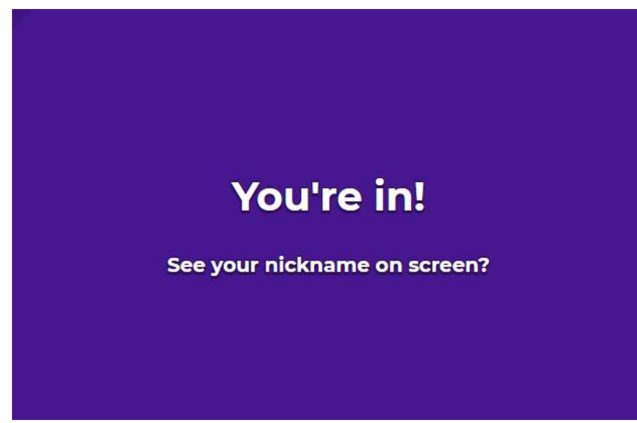


Setup

1. Go to www.kahoot.it
2. Enter the Following PIN: **7159551**
3. Choose a funny (anonymous) name!







Introduction

REVIEW

doi:10.1038/nature14539

Deep learning

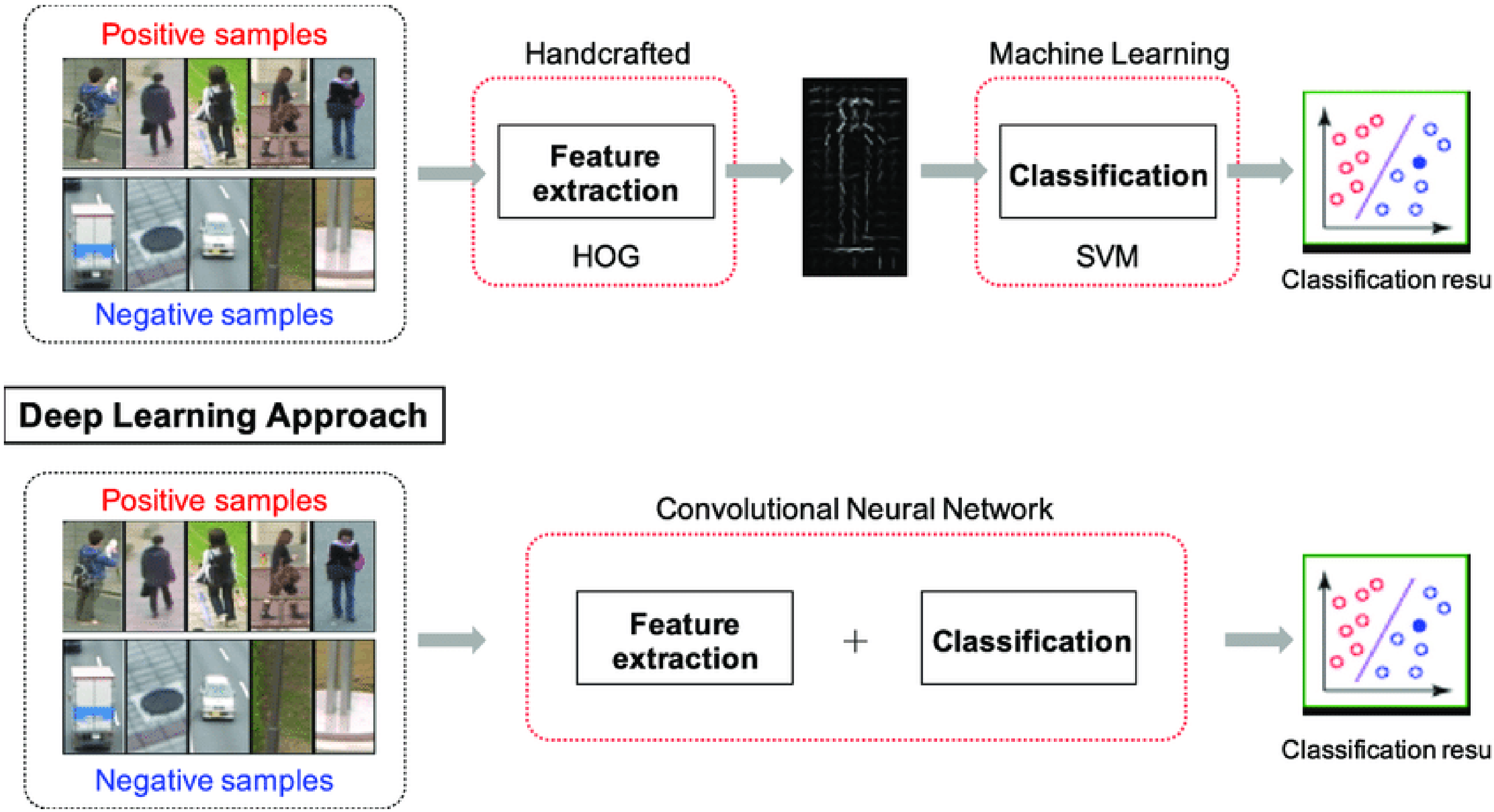
Yann LeCun[1,2], Yoshua Bengio[3] & Geoffrey Hinton[4,5]

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

→ **Introductory overview and state of the art of deep learning**

Introduction

→ **Conventional** machine learning vs. **representation** learning



Deep learning-based image recognition for autonomous driving - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Conventional-machine-learning-and-deep-learning_fig2_337804593 [accessed 10 May, 2021]

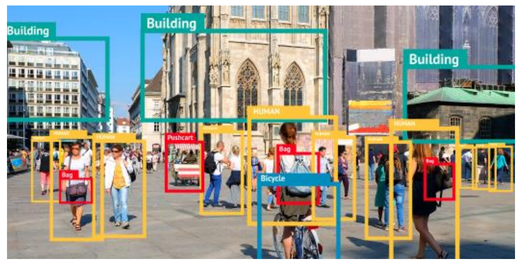
Introduction

Samoyed vs. White Wolf

Introduction - Applications

Natural Language Processing

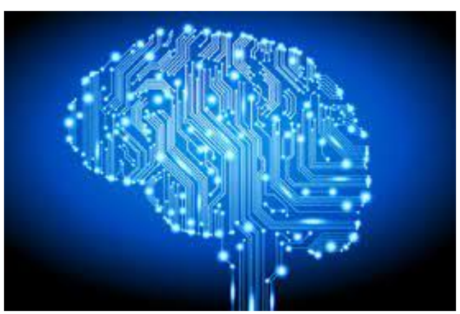
Image Processing



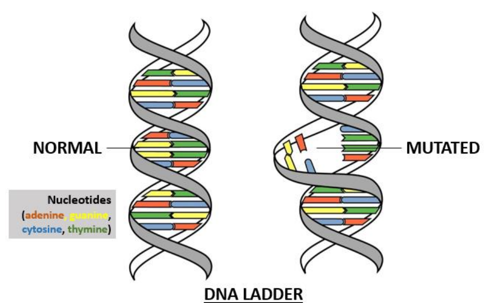
Combinations

A woman is throwing a **frisbee** in a park.

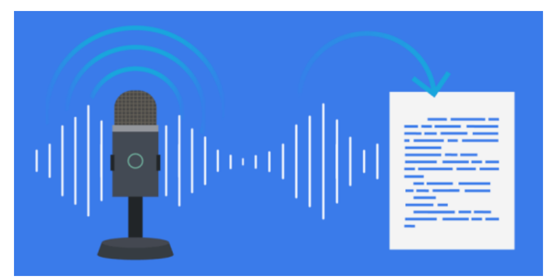
Brain Circuits



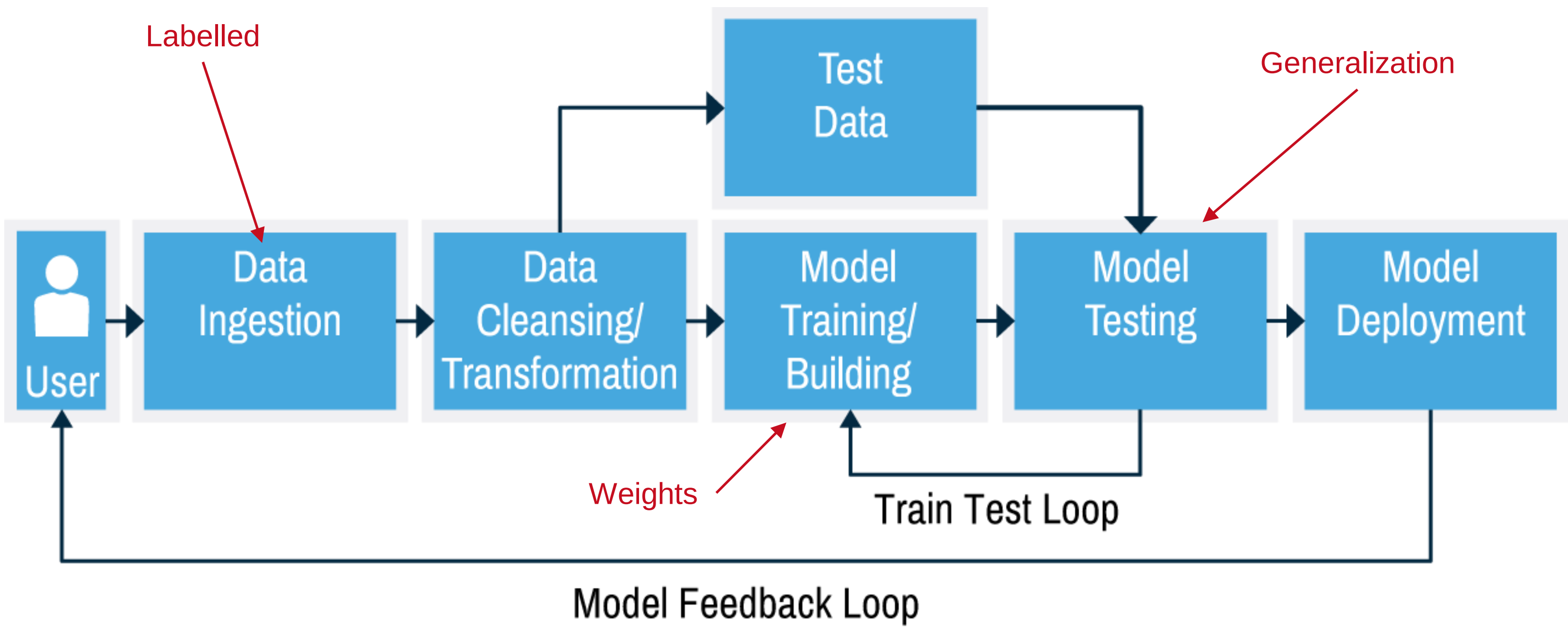
Effects of Mutation



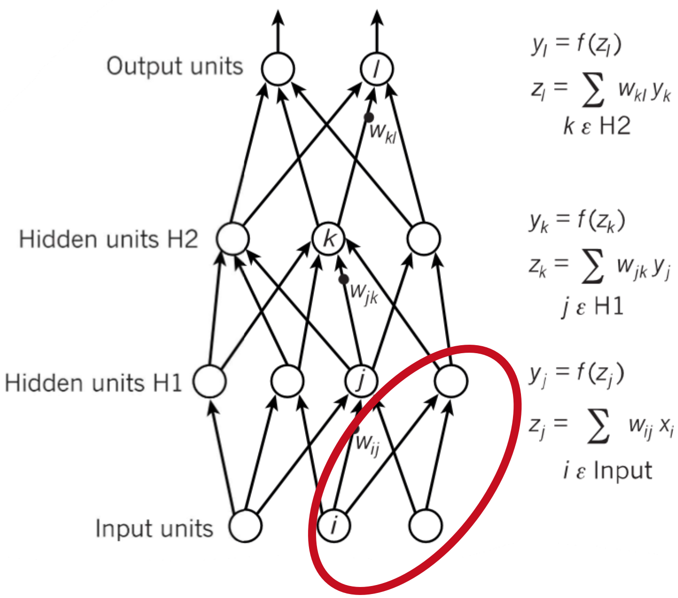
Speech Recognition

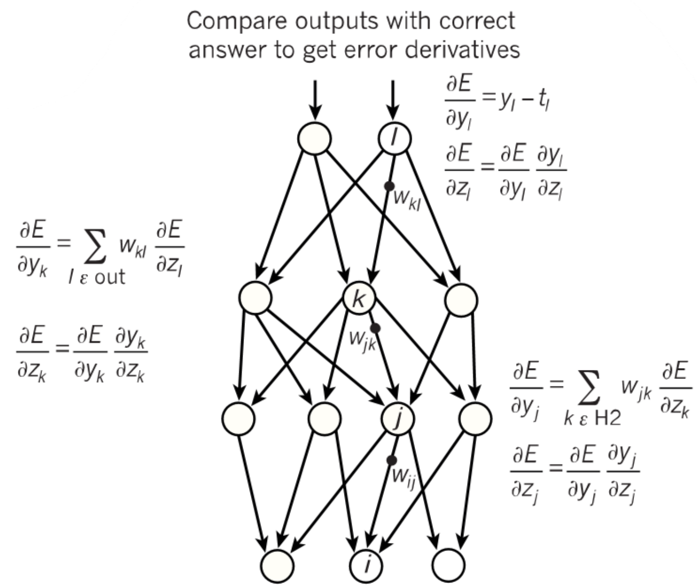


Supervised Learning



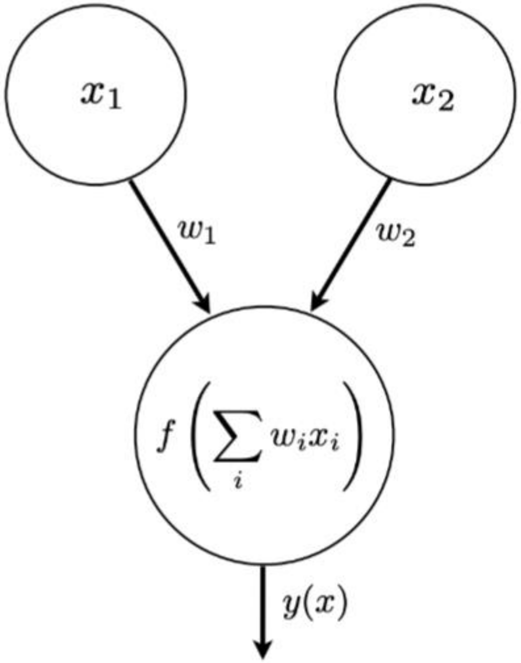
Basics





→ Let's take a step back

Perceptron



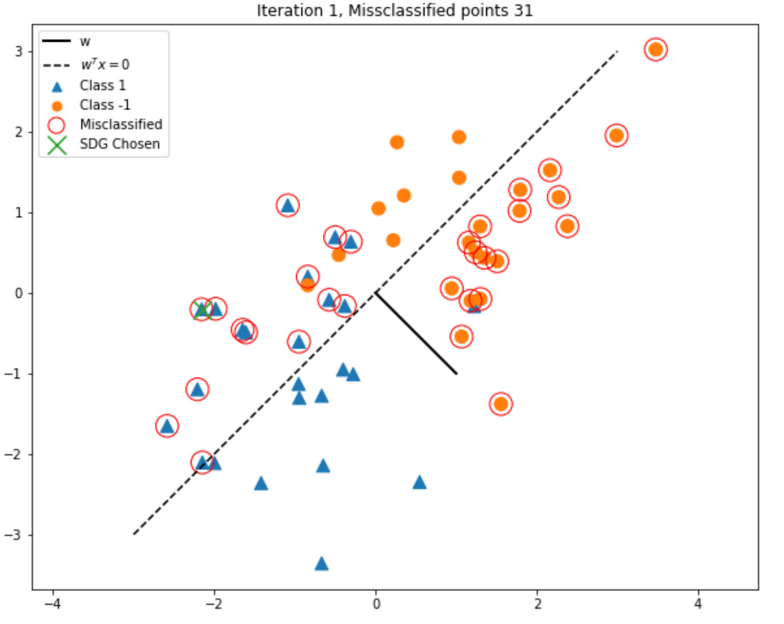
$$\hat{y}(x) = sign(\boldsymbol{w}^T \boldsymbol{x}) = \begin{cases} +1 \text{ Class } \Delta \\ -1 \text{ Class } 0 \end{cases}$$

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

For now:

$$\hat{y}(x) = \boldsymbol{w}^T \boldsymbol{x}$$

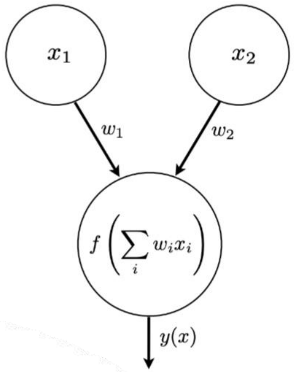


→ How do we train a perceptron?

Stochastic Gradient Descent

$$\boldsymbol{w}^{new} \leftarrow \boldsymbol{w}^{old} \overset{!}{-} \eta \frac{\partial \mathrm{E}}{\partial \boldsymbol{w}}$$

η : Learning Rate



- Need to define error term (cost function):

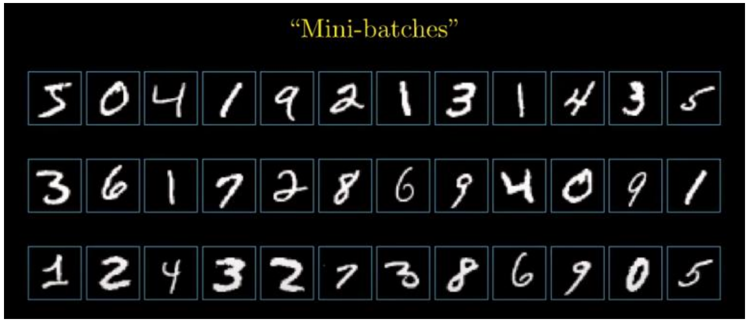
$$\mathrm{E} = \frac{1}{2} \sum_{m \in M} (\hat{y}_m - \mathrm{y_m})^2$$

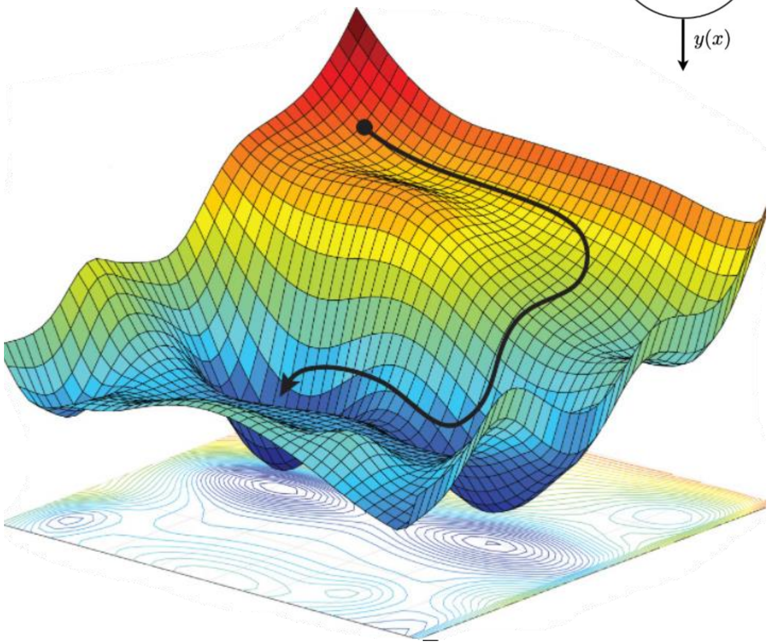
m : Misclassified

- Or used in Jupyter Notebook (for one misclassified point):

$$\mathrm{E} = -\hat{y}\, y_m = -\boldsymbol{w}^T \boldsymbol{x}_m y_m$$

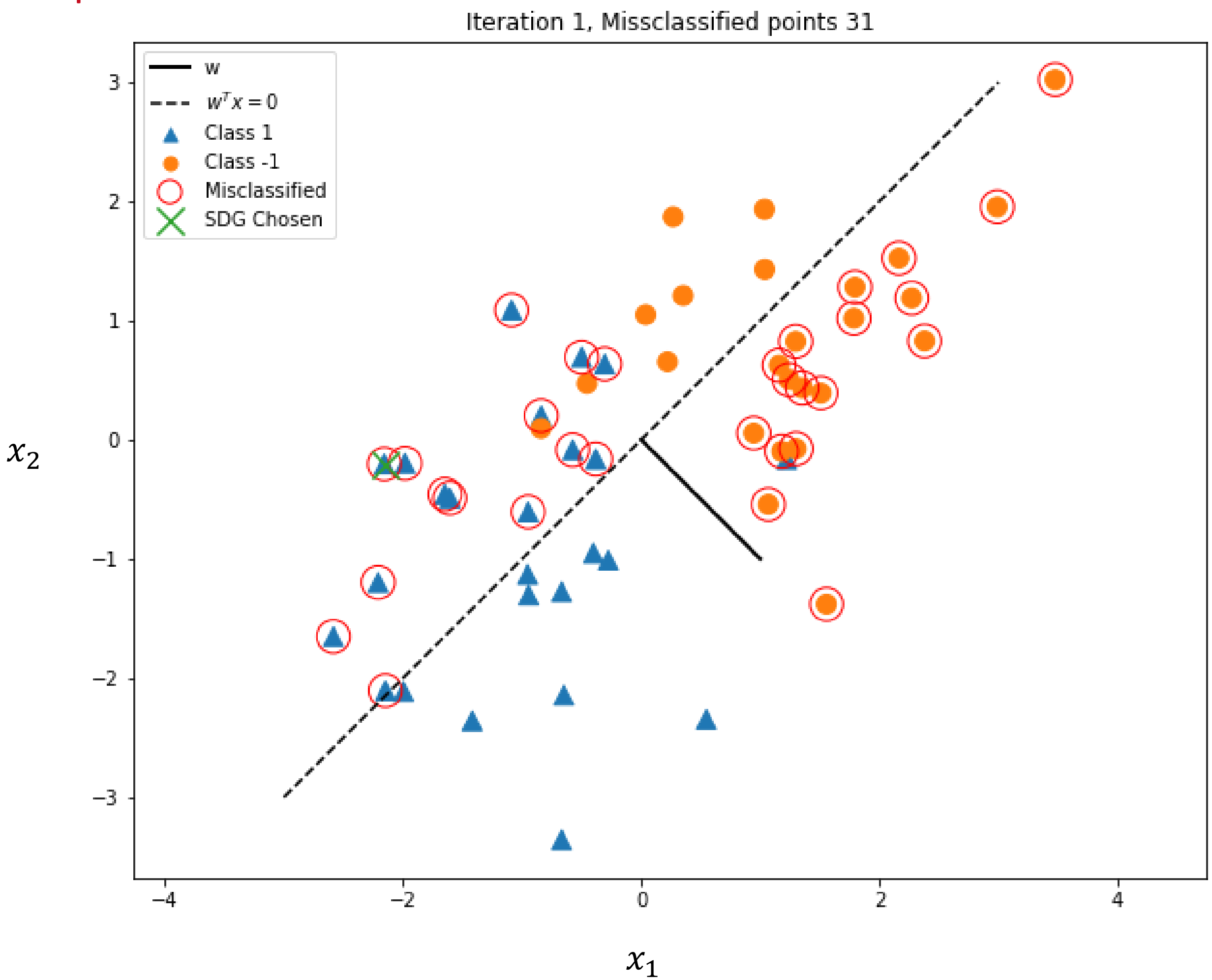
- Stochastic?



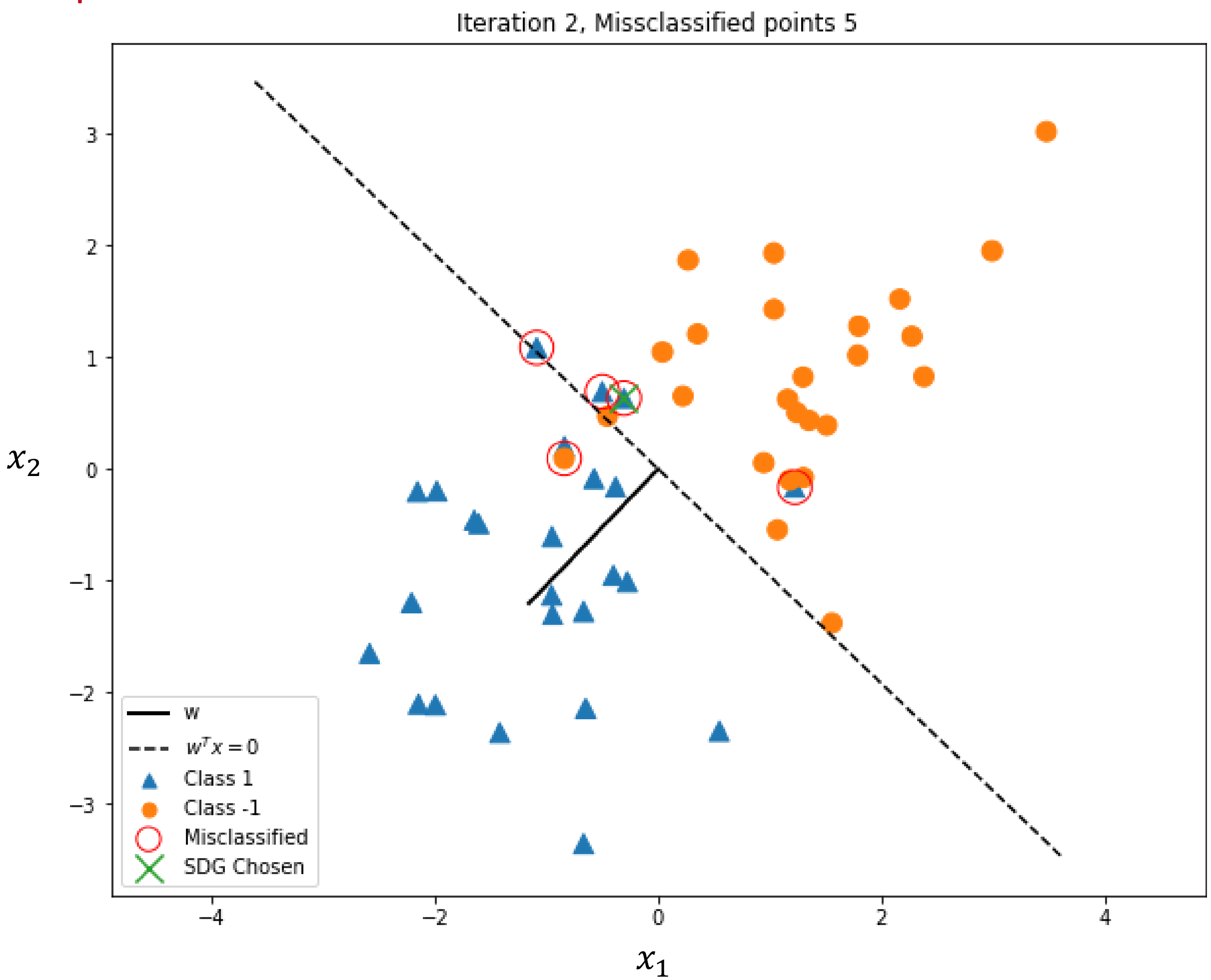


http://www.its.caltech.edu/~nazizanr/papers/SMD.html

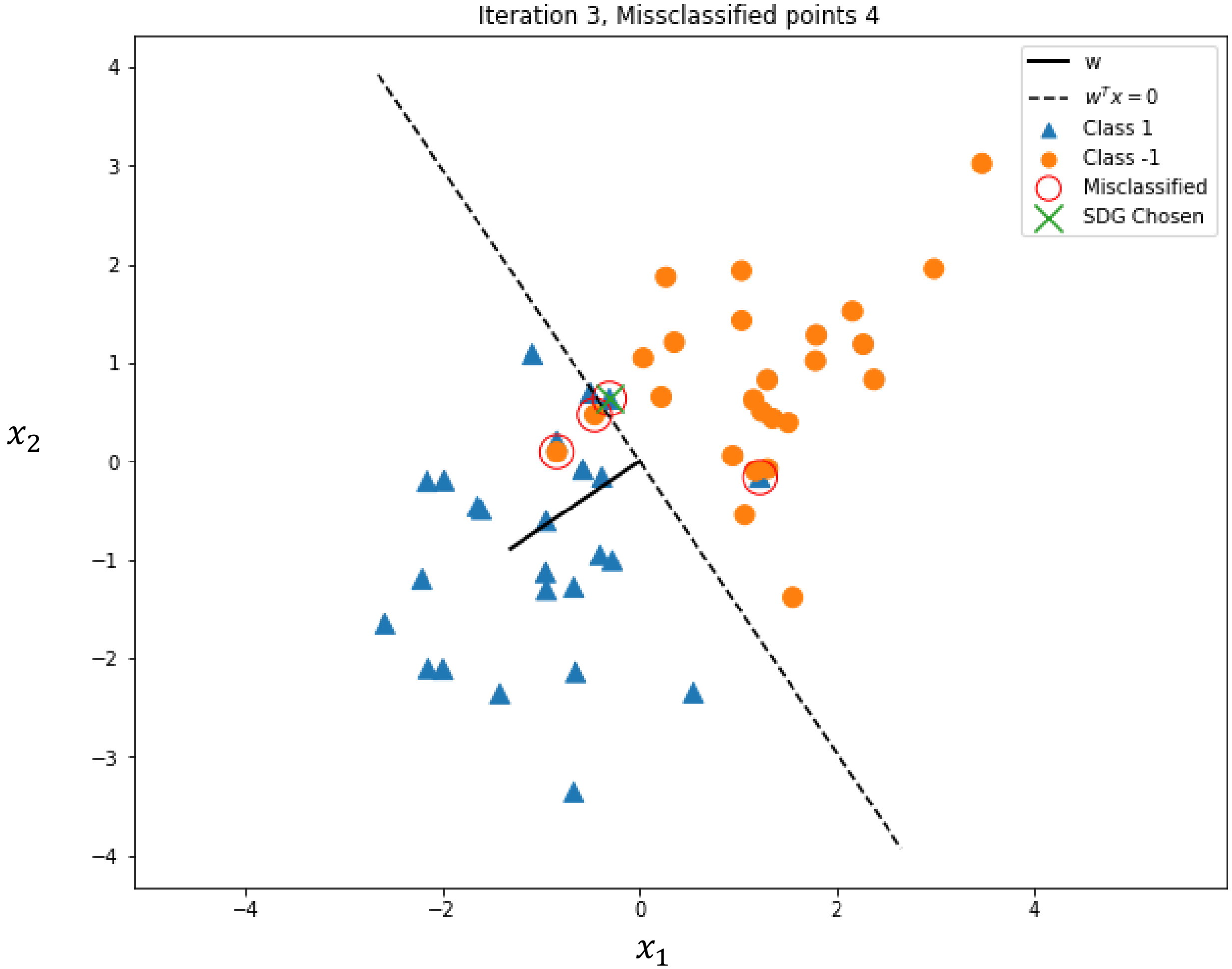
SGD Example Iteration 1



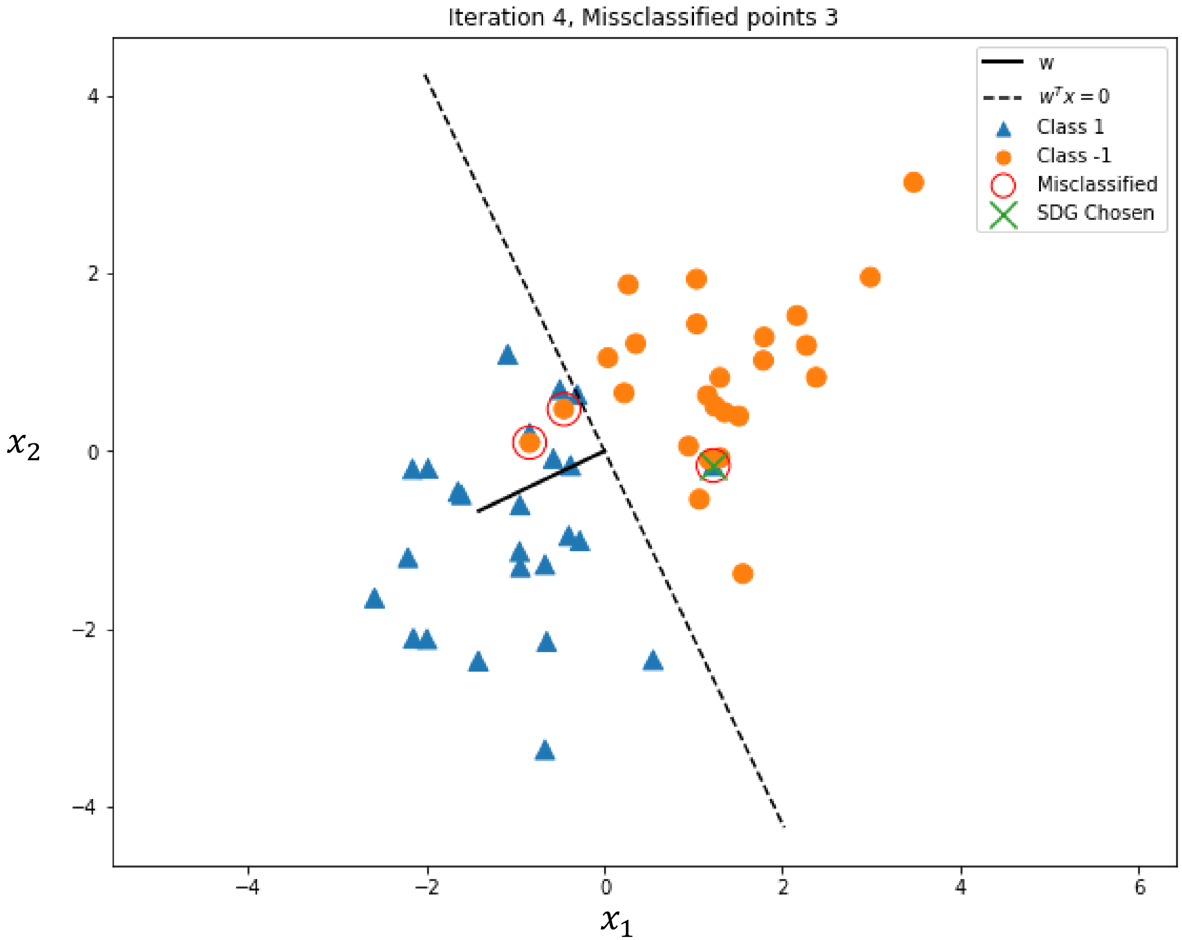
SGD Example Iteration 2



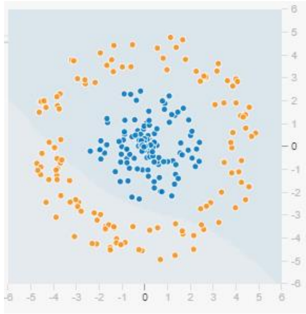
SGD Example Iteration 3



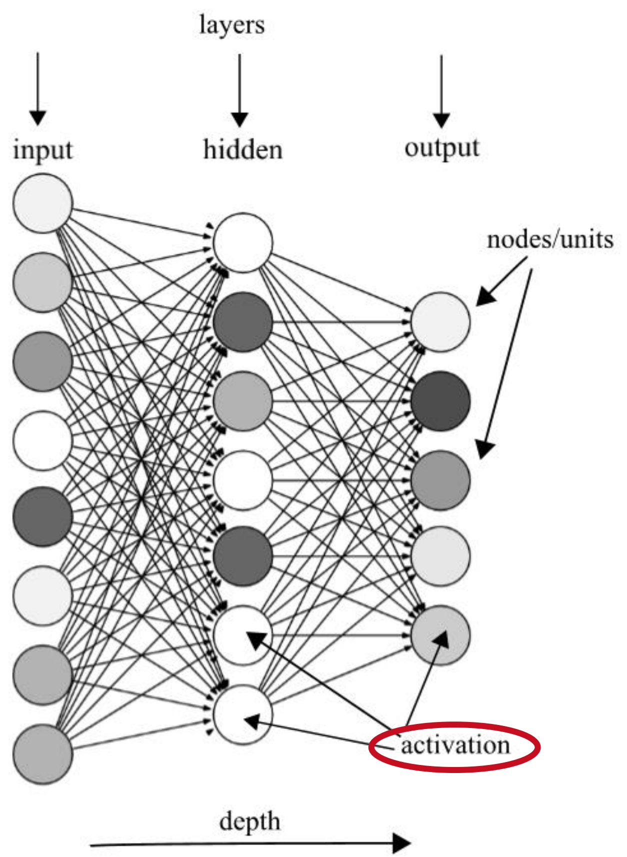
SGD Example Iteration 4

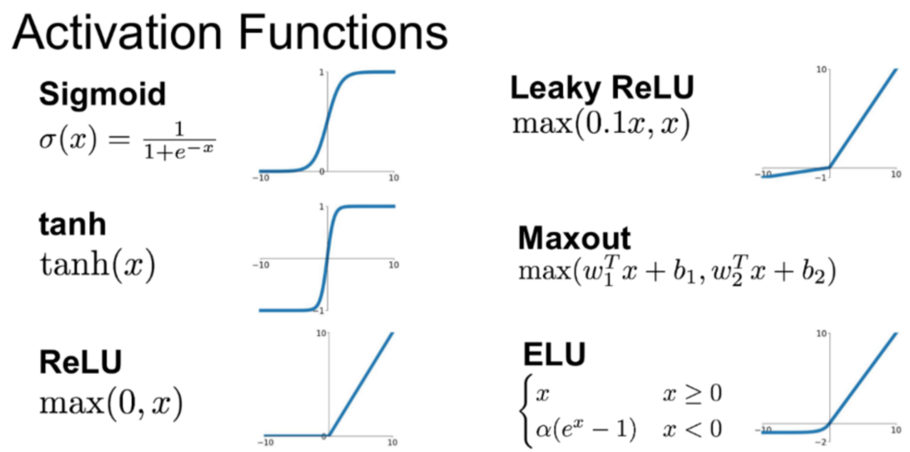


→ Limitations?

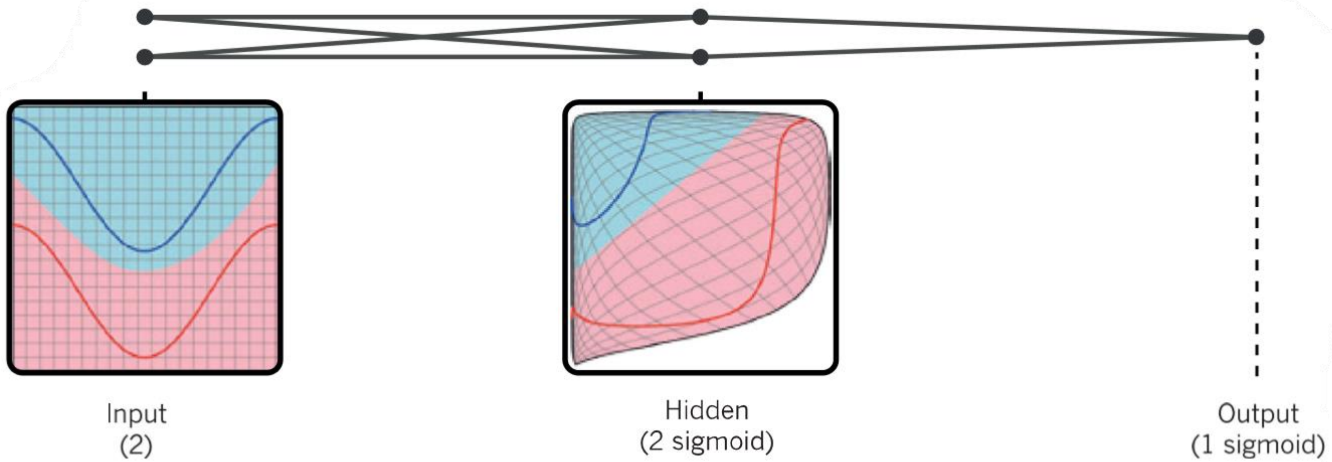


Deep feedforward Networks





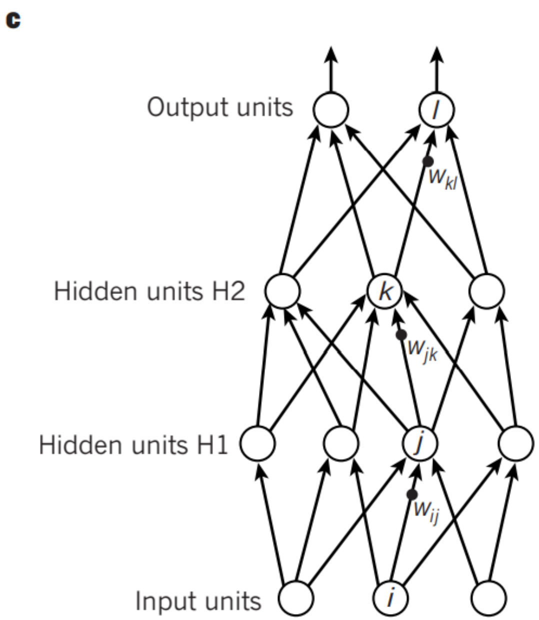
Deep feedforward Networks



Neural Network Traininig

1 Forward pass: Apply data with current weights

2 Backward pass: Error **backpropagation** for every node



$$y_l = f(z_l)$$
$$z_l = \sum_{k \,\varepsilon\, H2} w_{kl} y_k$$

$$y_k = f(z_k)$$
$$z_k = \sum_{j \,\varepsilon\, H1} w_{jk} y_j$$

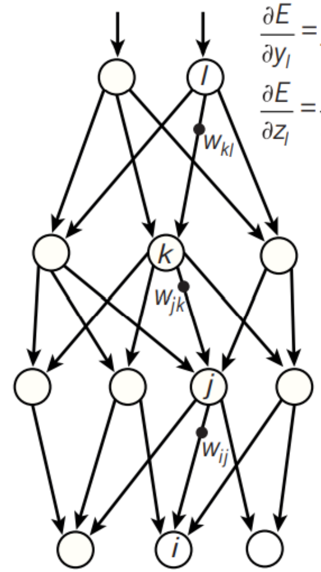
$$y_j = f(z_j)$$
$$z_j = \sum_{i \,\varepsilon\, Input} w_{ij} x_i$$

d

Compare outputs with correct answer to get error derivatives

$$E = \frac{1}{2}(y_l - t_l)^2$$

$$\frac{\partial E}{\partial y_l} = y_l - t_l$$
$$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial z_l}$$



$$\frac{\partial E}{\partial y_k} = \sum_{l \,\varepsilon\, out} w_{kl}\frac{\partial E}{\partial z_l}$$
$$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k}$$

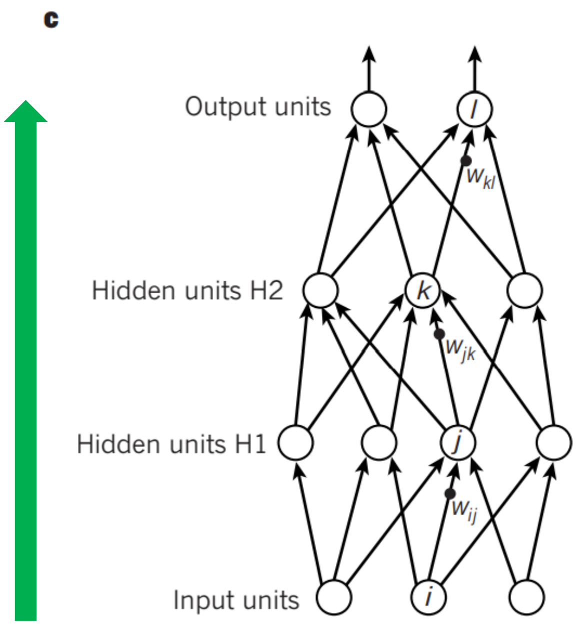
$$\frac{\partial E}{\partial y_j} = \sum_{k \,\varepsilon\, H2} w_{jk}\frac{\partial E}{\partial z_k}$$
$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j}$$

3 Update weights:

$$\boldsymbol{w}^{new} \leftarrow \boldsymbol{w}^{old} - \eta\frac{\partial \mathrm{E}}{\partial \boldsymbol{w}} \rightarrow \frac{\partial \mathrm{E}}{\partial w_{jk}} = \frac{\partial \mathrm{E}}{\partial z_k}\frac{\partial z_k}{\partial w_{jk}} = \frac{\partial \mathrm{E}}{\partial z_k} y_j$$

Neural Network Traininig

1 Forward pass: Apply data with current weights



$y_l = f(z_l)$

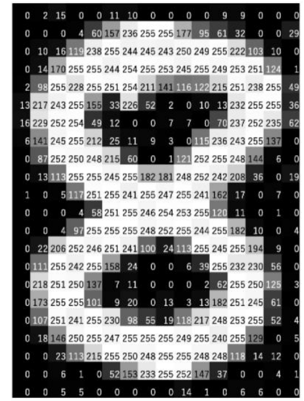
$z_l = \sum_{k \,\varepsilon\, H2} w_{kl} \, y_k$

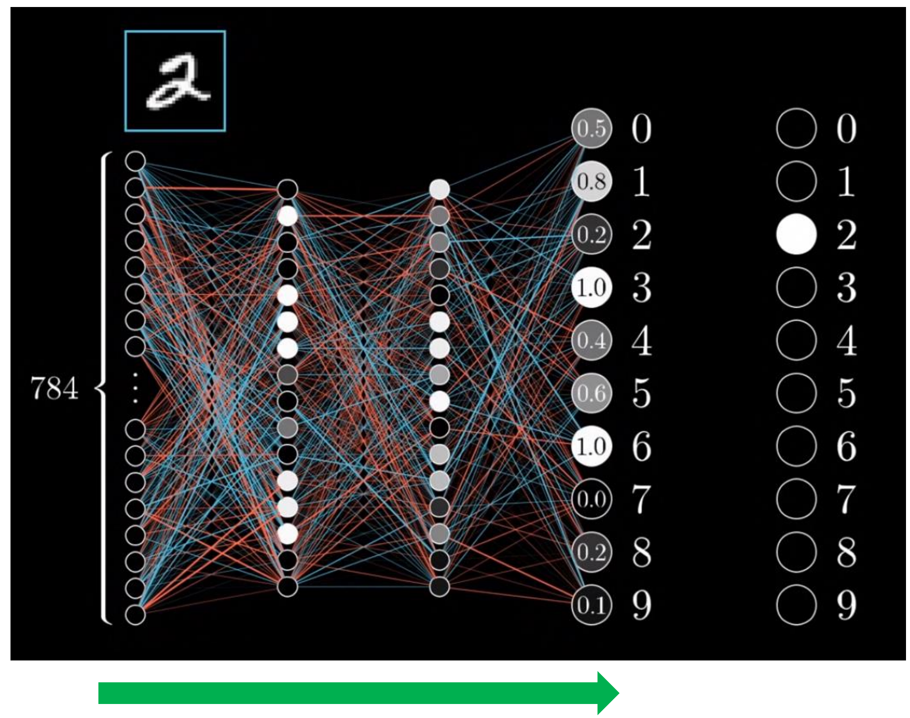
$y_k = f(z_k)$

$z_k = \sum_{j \,\varepsilon\, H1} w_{jk} \, y_j$

$y_j = f(z_j)$

$z_j = \sum_{i \,\varepsilon\, \text{Input}} w_{ij} \, x_i$





https://www.youtube.com/watch?v=Ilg3gGewQ5U&t=740s

Neural Network Traininig

2 Backward pass: Error **backpropagation** for every node

d Compare outputs with correct answer to get error derivatives

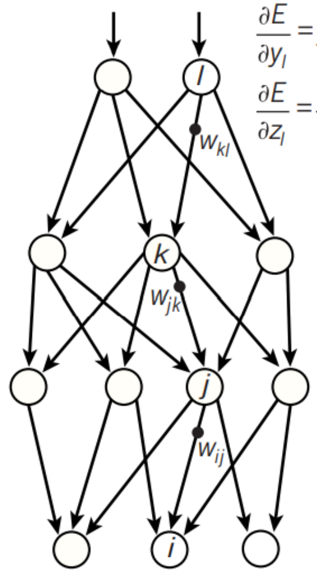
$$E = \frac{1}{2}(y_l - t_l)^2$$

$$\frac{\partial E}{\partial y_l} = y_l - t_l$$

$$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial z_l}$$

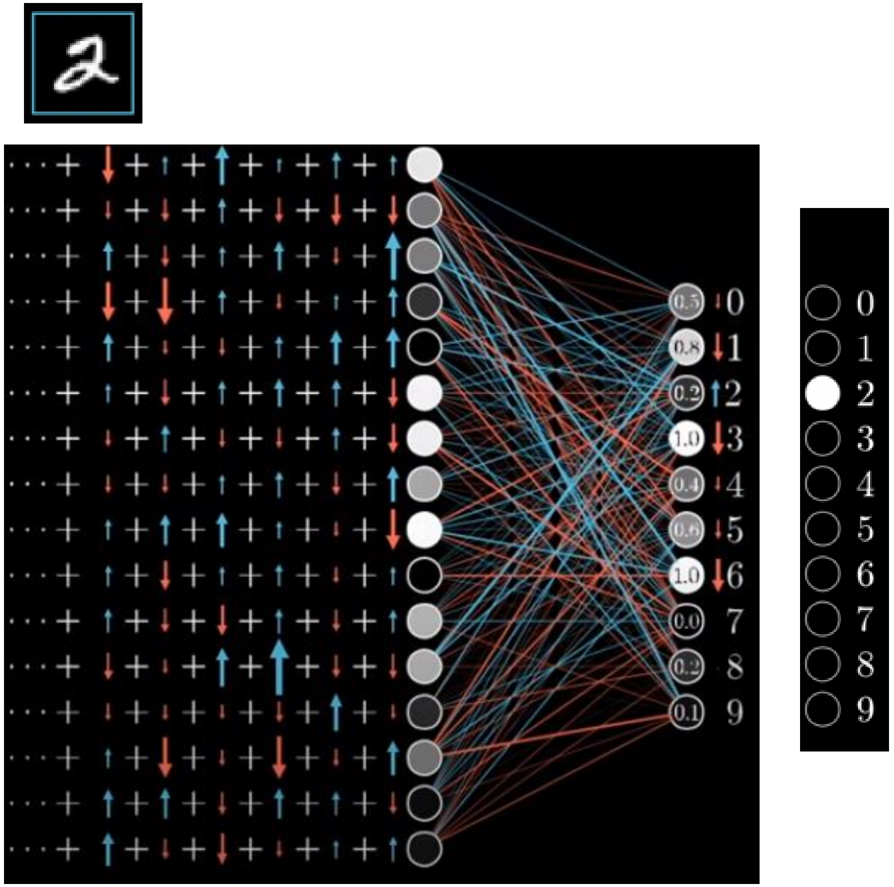
$$\frac{\partial E}{\partial y_k} = \sum_{l\,\varepsilon\,\text{out}} w_{kl}\frac{\partial E}{\partial z_l}$$

$$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k}$$



$$\frac{\partial E}{\partial y_j} = \sum_{k\,\varepsilon\,\text{H2}} w_{jk}\frac{\partial E}{\partial z_k}$$

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j}$$

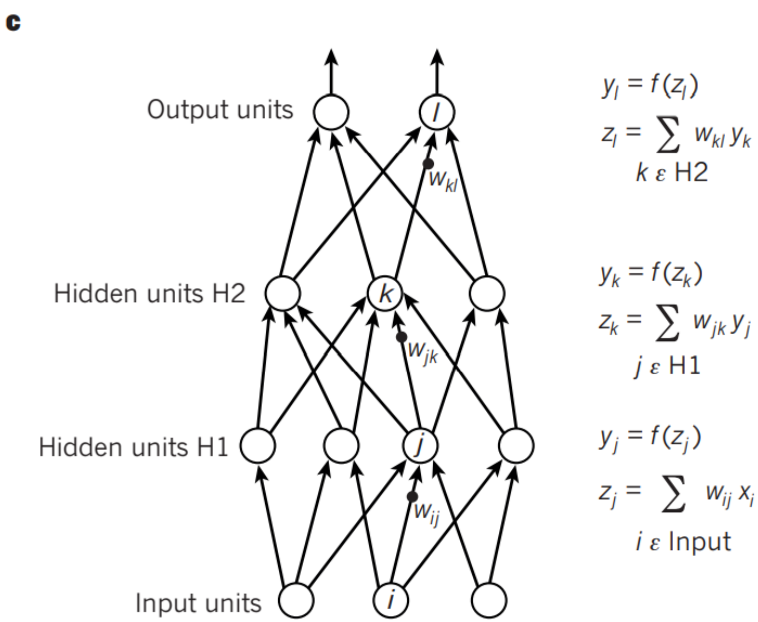


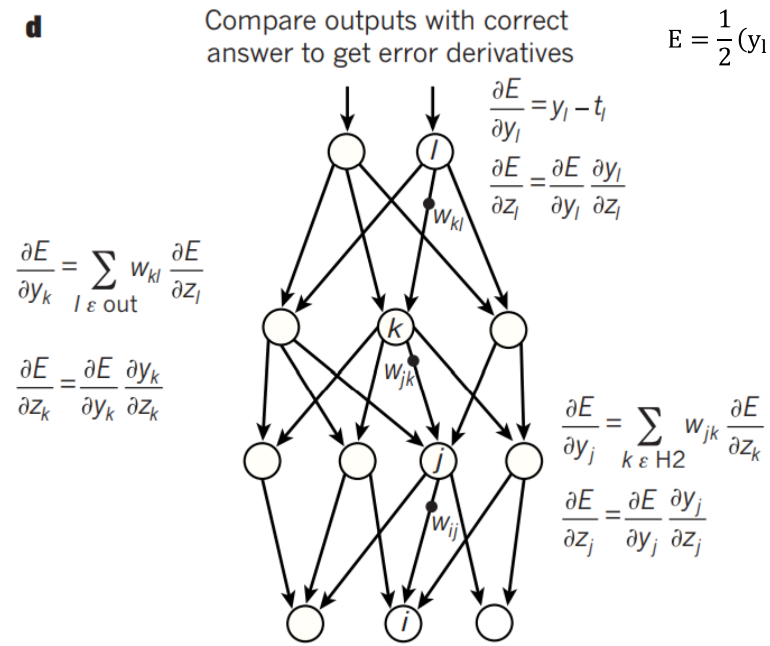
https://www.youtube.com/watch?v=Ilg3gGewQ5U&t=740s

Neural Network Traininig

1 Forward pass: Apply data with current weights

2 Backward pass: Error **backpropagation** for every node





$$E = \frac{1}{2}(y_l - t_l)^2$$

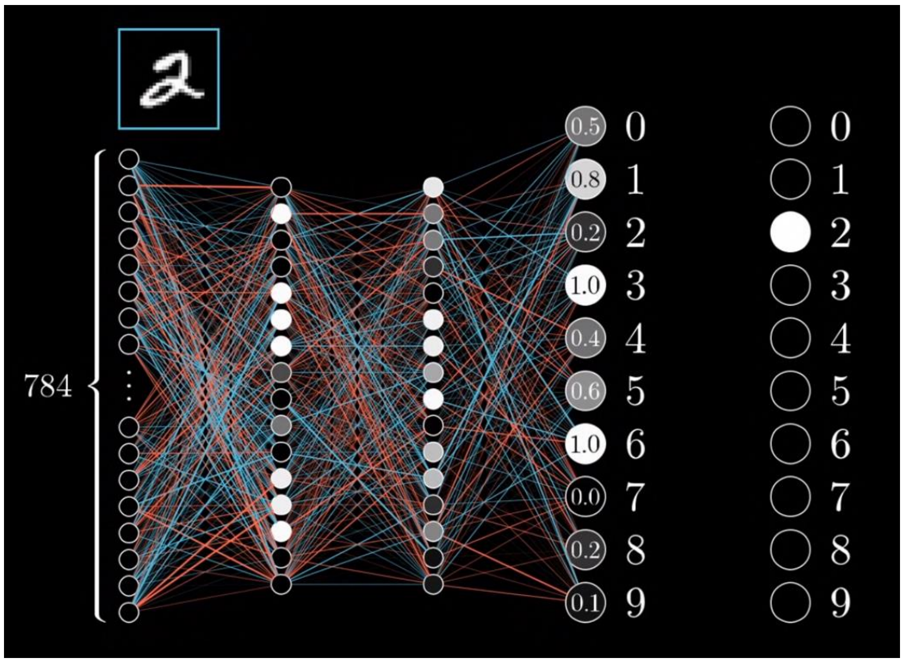
3 Update weights:

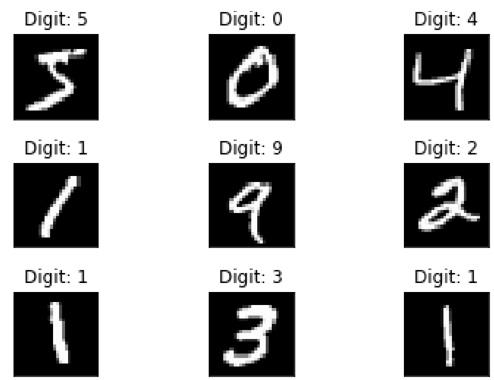
$$\boldsymbol{w}^{new} \leftarrow \boldsymbol{w}^{old} - \eta \frac{\partial E}{\partial \boldsymbol{w}} \rightarrow \frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial w_{jk}} = \frac{\partial E}{\partial z_k} y_j$$

Technische Universität Berlin

Neural Network – MNIST

→ MNIST: The hello world of Neural Networks



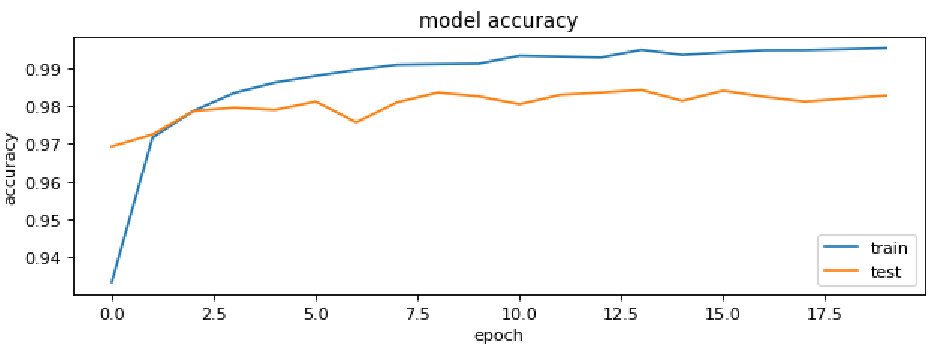


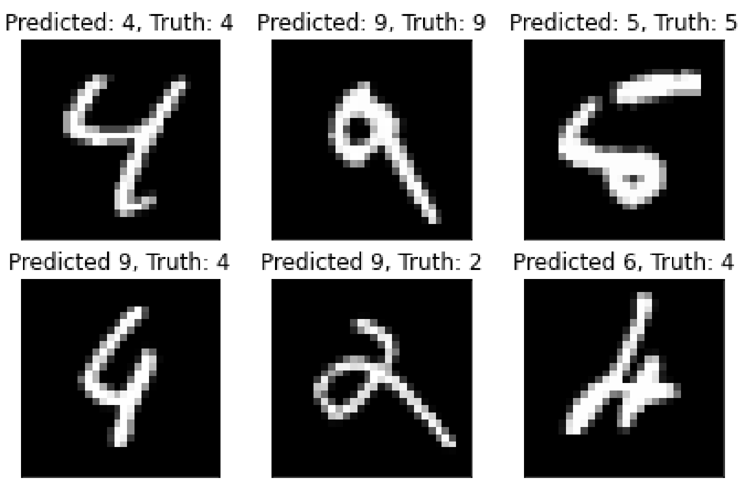
28x28 Pixel Images with B/W values between 0..255

1st time:	784	784	10
2nd time:		10	10

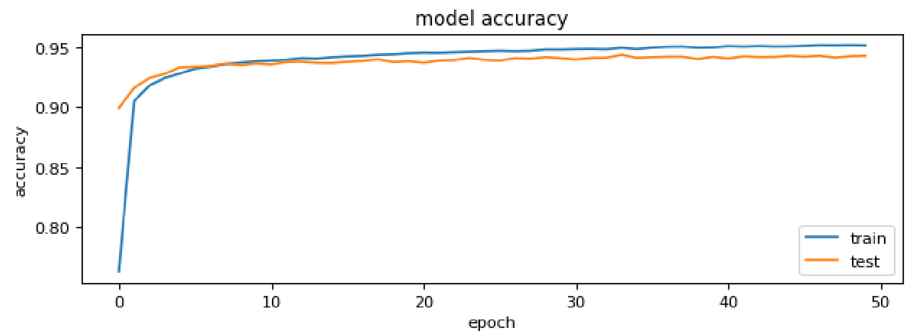
Neural Network – MNIST

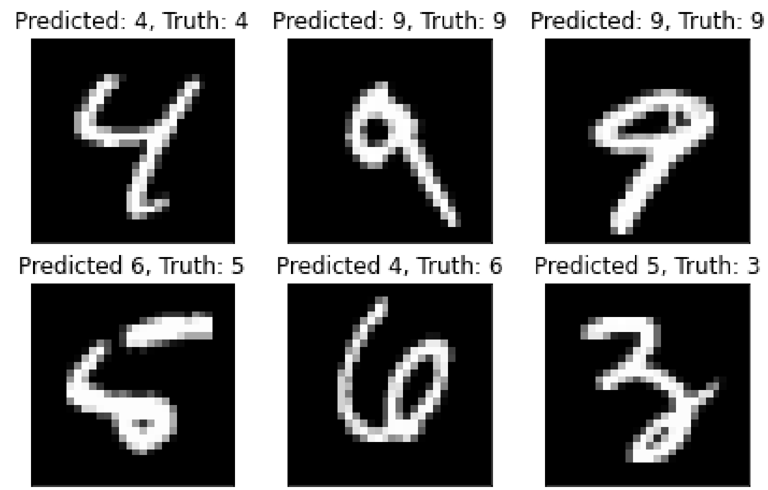
1st Model – 1568 hidden layer nodes





2nd Model – 10 hidden layer nodes

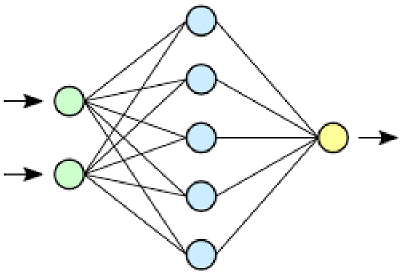




Deep neural networks – Pre Training

- 1970s and 1980s solution widely understood for backpropagation
- *“Forsaken by the machine-learning community and ignored by the computer-vision and speech-recognition communities”* until..
- 2006 **pre training** for handwritten digits at CIFAR, 2009 for record breaking speech recognition
  - Solves the vanishing gradient problem
  - Simplified training process
  - Facilitates the development of deeper networks
  - Useful as a weight initialization scheme
  - Lower generalization error

https://machinelearningmastery.com/greedy-layer-wise-pretraining-tutorial/

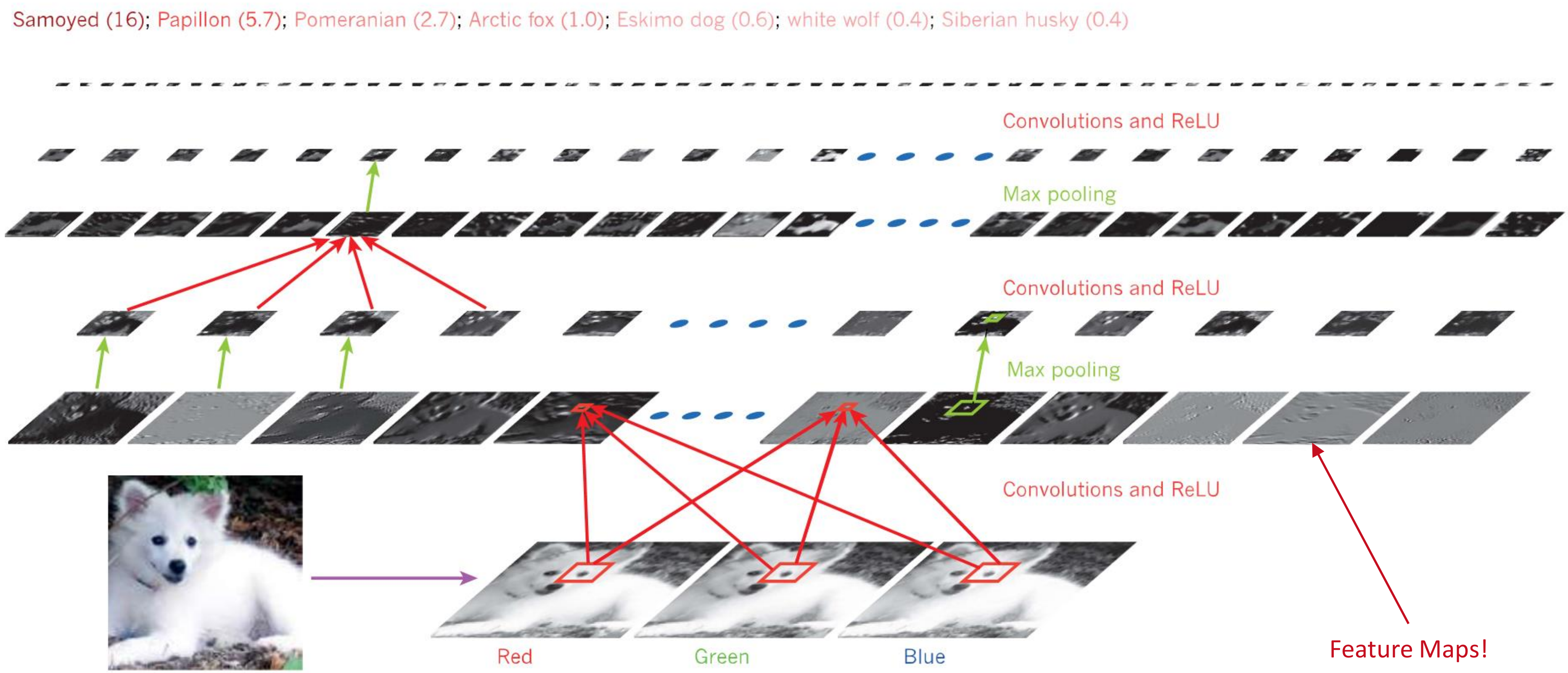


“(…) no one knows why exactly this works, but the idea is that by pre-training you start from more favorable regions of feature space.”

https://www.quora.com/What-is-unsupervised-pre-training

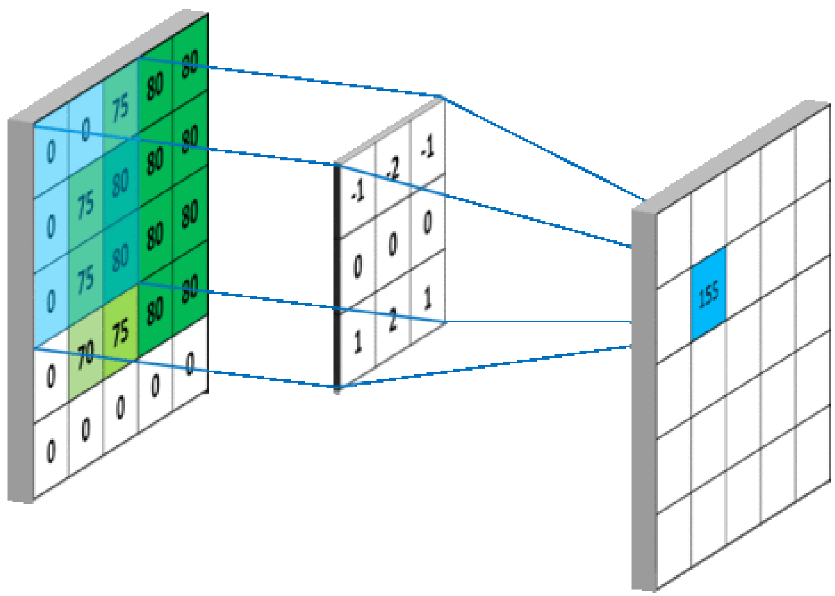
What about the computer vision community? → Convolutional neural networks

Convolutional Neural Network



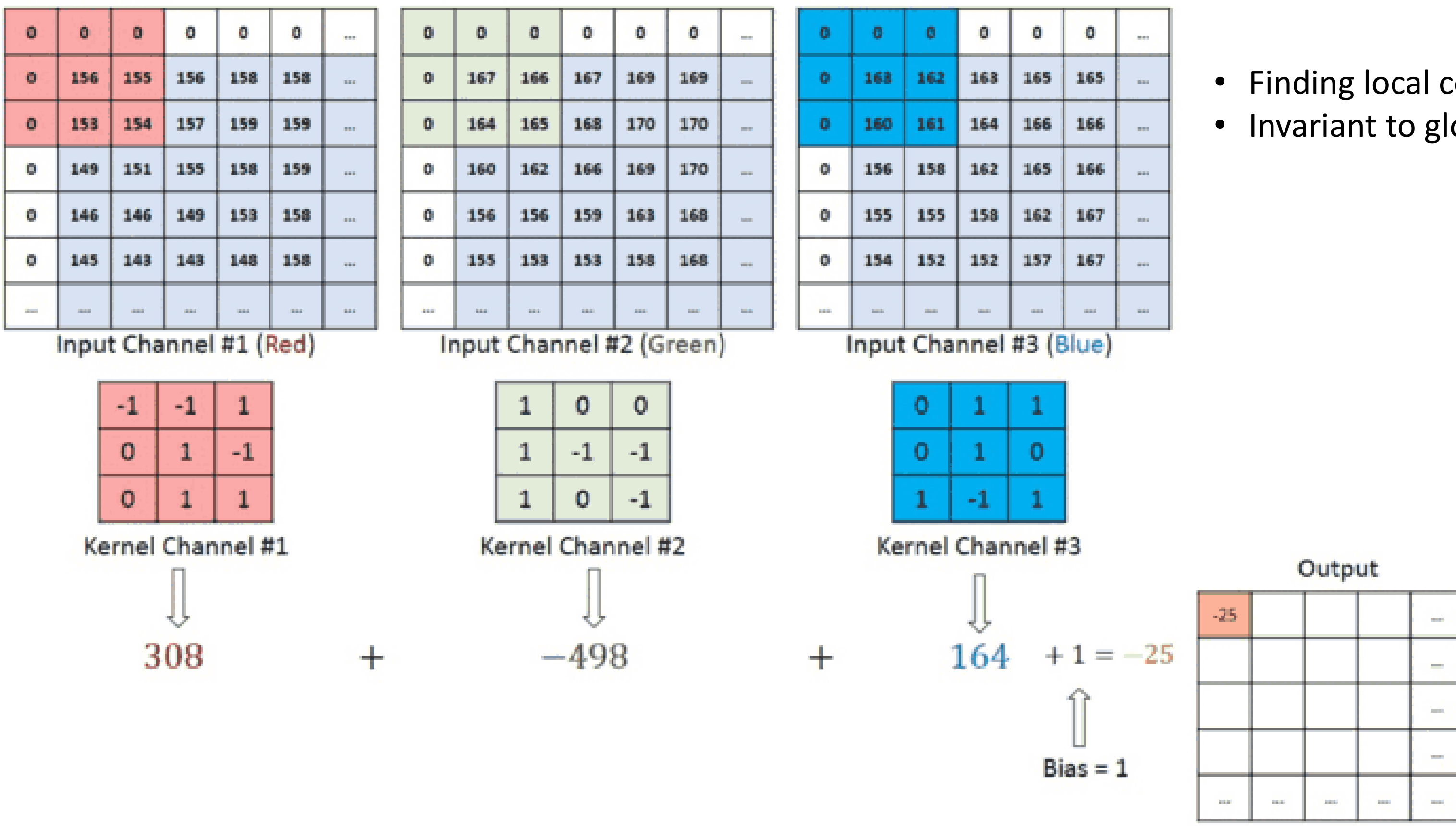
Convolutional Neural Network

Convolution with a 3x3 filter:



Convolutional Neural Network

Convolution with a 3x3x3 Filter:

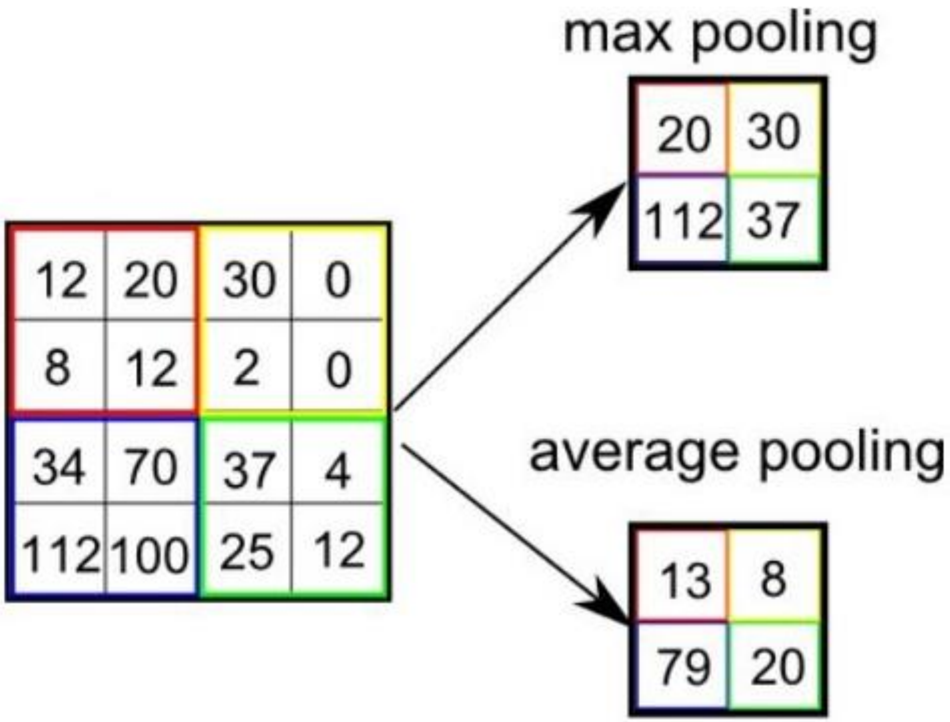


- Finding local connections
- Invariant to global location

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolutional Neural Network

Max and Average Pooling:

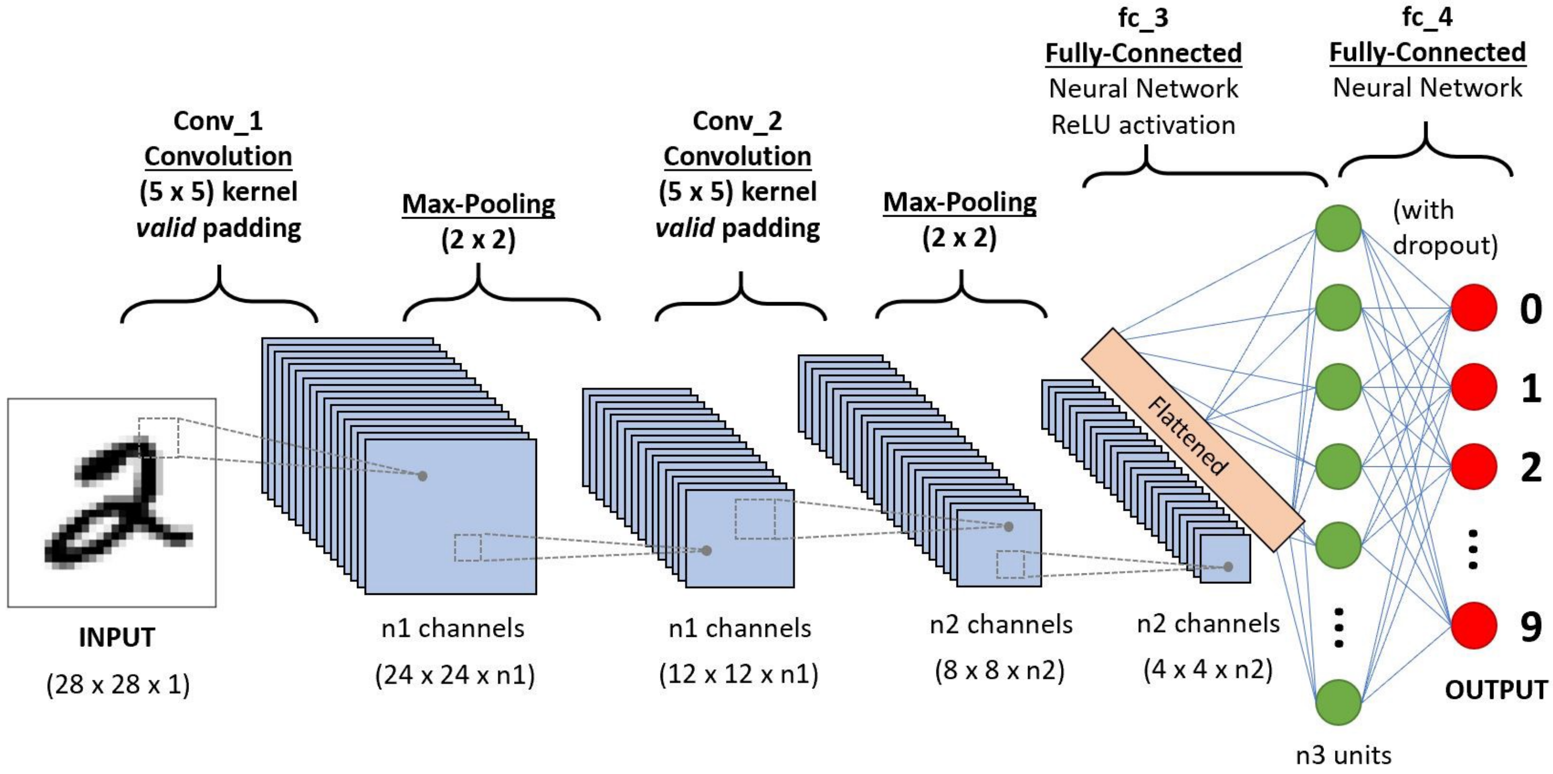


- Merge semantically similar features into one
- Reduces dimension
  - Just imagine an 8192 x 4608 = 37748736 Pixel Image

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

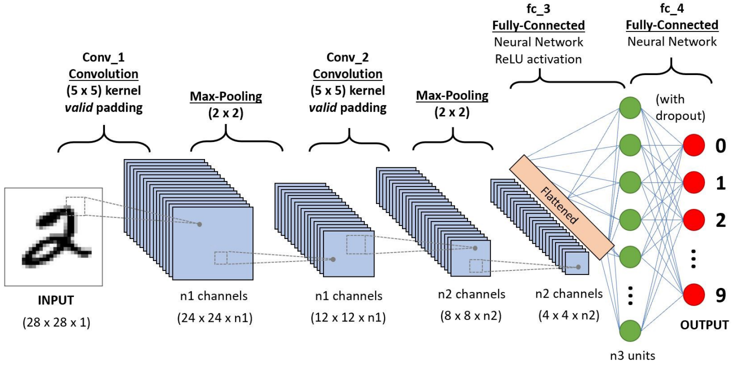
Convolutional Neural Network

Fully connected Layer:



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

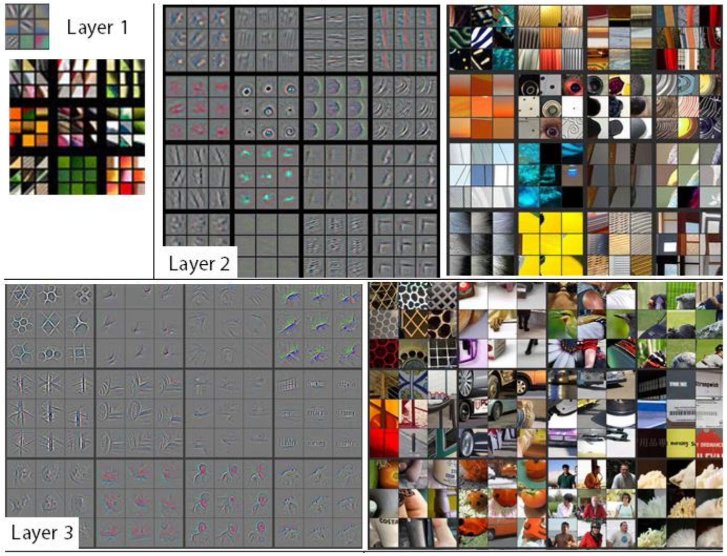
Convolutional Neural Network



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Why does it work so well?
- Higher level features are obtained by composing lower level one
- Inspired by visual neuroscience

Breakthrough?
- 2012 ImageNet competition
- Efficient use of GPUs, ReLUs and dropout regularization technique



→ What about natural language processing?

Distributed representation – Word Vectors

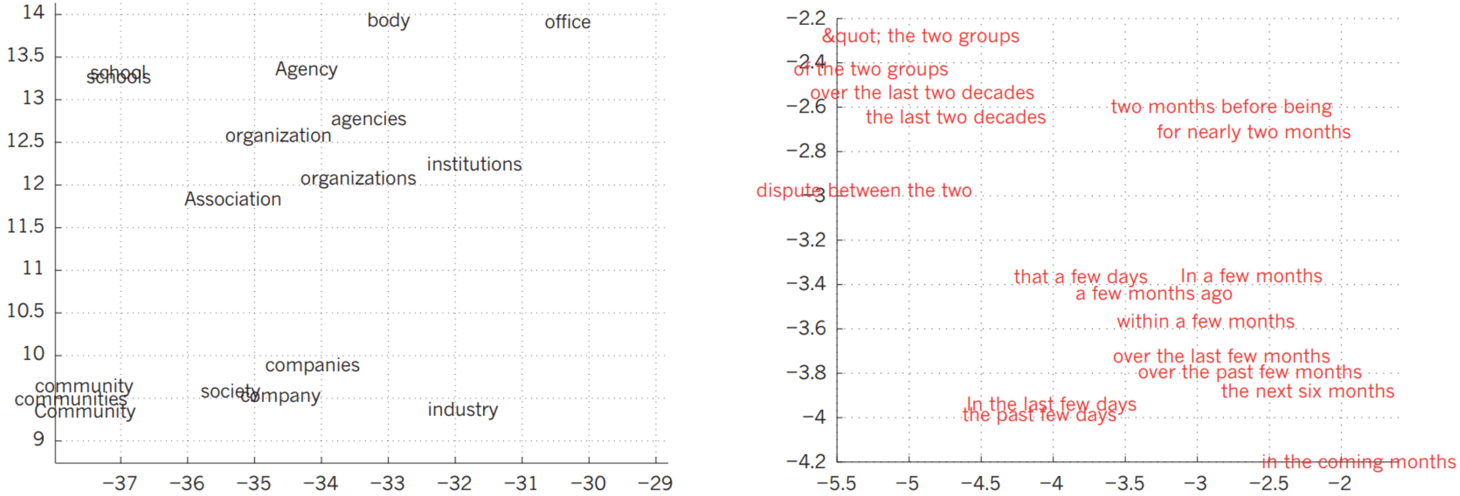
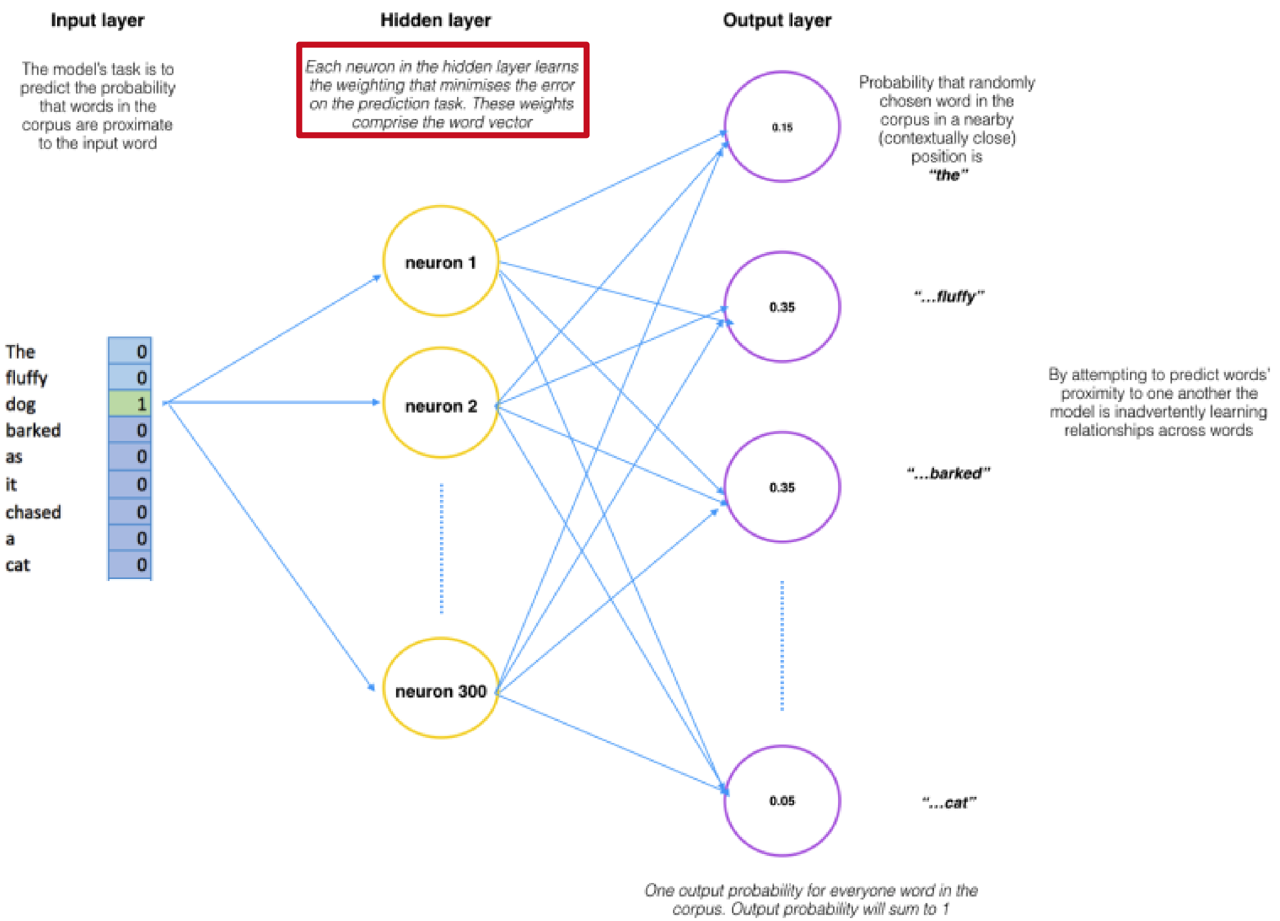


Figure 4 | Visualizing the learned word vectors. On the left is an illustration of word representations learned for modelling language, non-linearly projected to 2D for visualization using the t-SNE algorithm[103]. On the right is a 2D representation of phrases learned by an English-to-French encoder–decoder recurrent neural network[75]. One can observe that semantically similar words or sequences of words are mapped to nearby representations. The distributed representations of words are obtained by using backpropagation to jointly learn a representation for each word and a function that predicts a target quantity such as the next word in a sequence (for language modelling) or a whole sequence of translated words (for machine translation)[18,75].

Distributed representation – Word Vectors



Recurrent Neural Networks

“(..) For tasks that involve sequential inputs, such as speech and language, it is often better to use RNNs (Fig. 5). RNNs process an input sequence one element at a time, maintaining in their hidden units a ‘state vector’ that implicitly contains information about the history of all the past elements of the sequence.”

- Handle variable-length sequences
- Track long-term dependencies
  - Share parameters across the sequence
- Maintain information about order

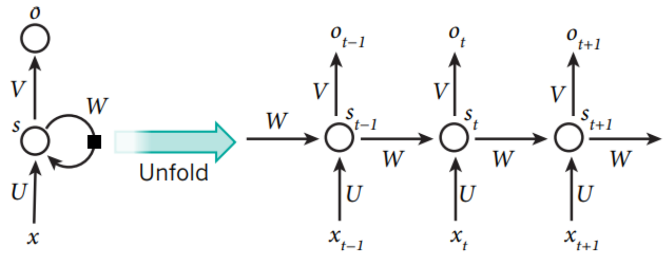


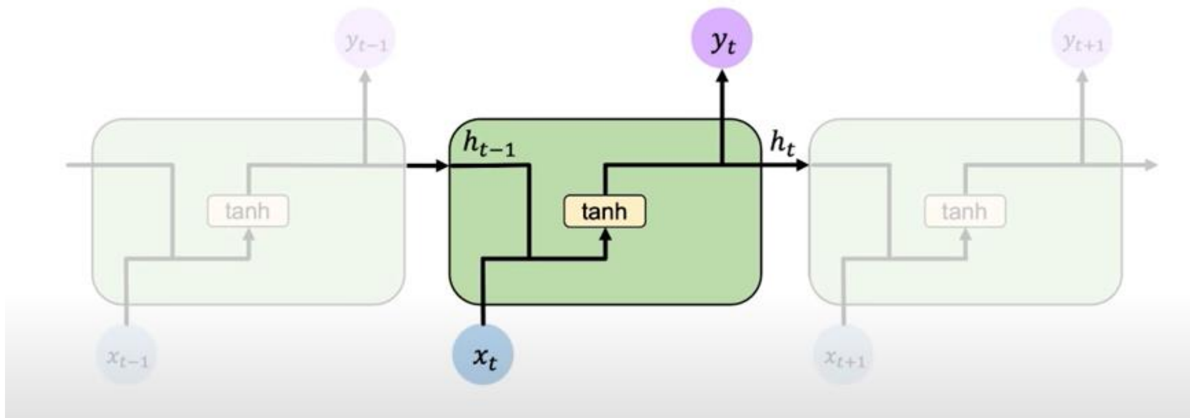
Figure 5 | A recurrent neural network and the unfolding in time of the computation involved in its forward computation. The artificial neurons

U,V,W: Weights
s: Hidden units/ state
x: Input
o: Output

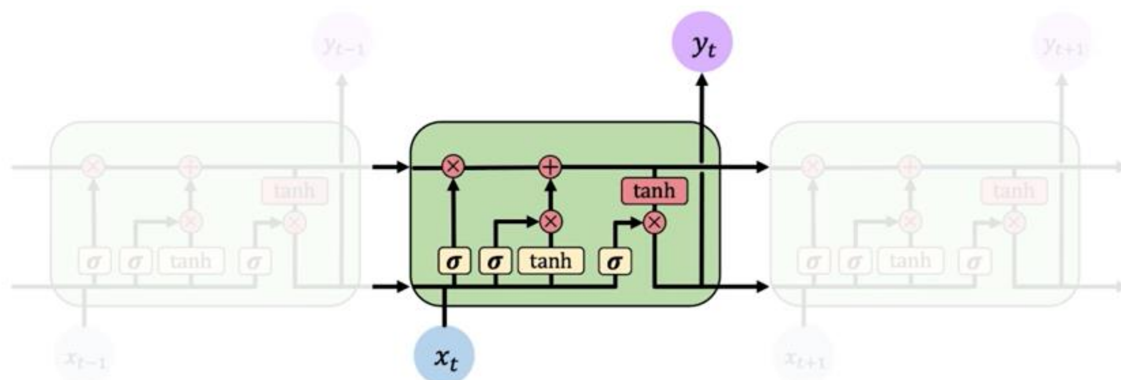
Recurrent Neural Networks - LSTM

→ LSTM: Long short-term memory

RNN:



LSTM:



Future and Conclusion

Authors:
- Authors expect unsupervised learning to become far more important
- CNN+RNN and reinforcement learning
- Combining representation learning with complex reasoning

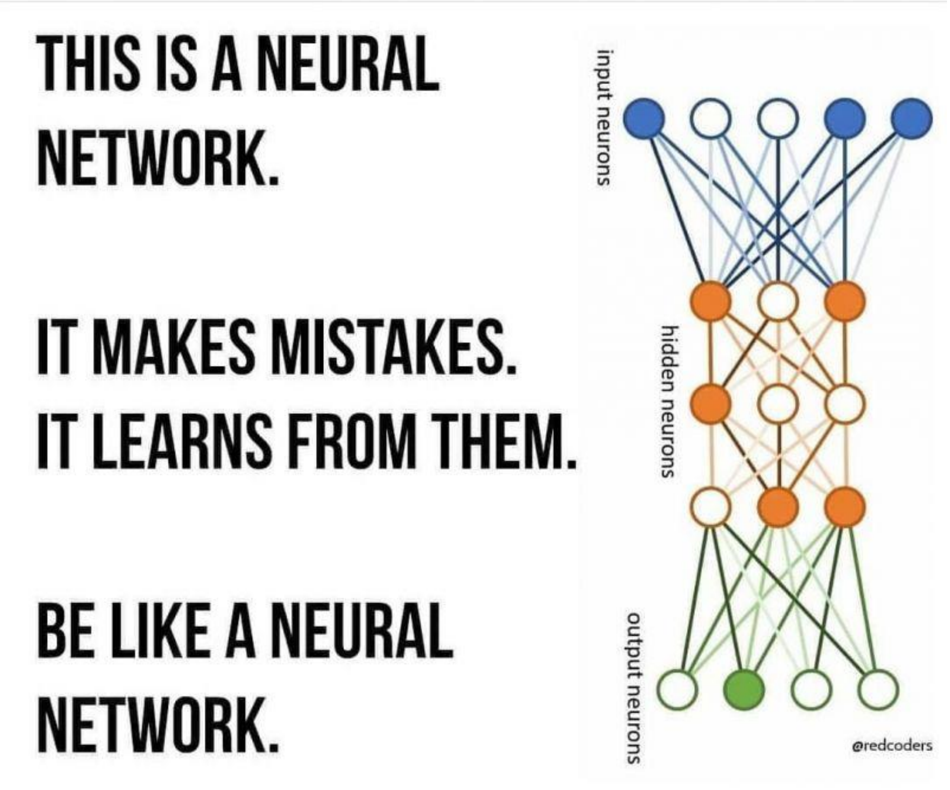
My View on Deep Learning:
- Great to not do feature extraction, easy to use
- Mathematicians still on their way to fully understand (most things not proven)
- Very experimental, safety and robustness concerns

- **Way too much to cover in one paper or lessen!**

Really great resources:
- https://www.youtube.com/watch?v=aircAruvnKk Best Youtube Series on Deep Learning you will ever see!
- https://playground.tensorflow.org/ Great playground for neural networks
- https://towardsdatascience.com/training-a-convolutional-neural-network-from-scratch-2235c2a25754 - Article on how CNN are trained
- https://www.youtube.com/watch?v=qjrad0V0uJE MIT Lecture (RNN)
- https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/ Word Vectors

Technische Universität Berlin

Questions?



https://devrant.com/rants/1922673/be-like-a-neural-network