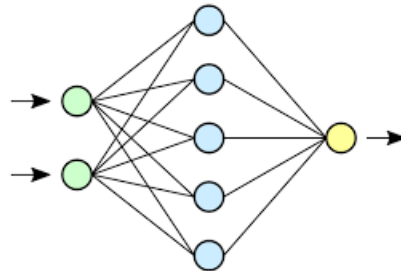


Deep Learning



Erik Suer

Journal Club Maschinelles Lernen

Summer Term 2021

<https://github.com/Erikx3/DeepLearningIntro>



Setup

1. Go to <https://kahoot.it>
2. Enter the Following ID: 7159551
3. Click on a funny (unimportant) name!

Kahoot!

Email ID

Password

Login

You're in!
We've got questions on hand!

17.02.2021

Setup Summary | Journal Club: Miscellaneous | L101: Sex

Supervised Learning

Labels

Data Ingestion

Data Cleaning/Transformation

Model Training

Model Testing

Model Deployment

Weights

Train Test Loop

Generalization

Model Feedback Loop

TUM

Technische Universität München

Deep Learning / Supervised Learning / Classification / DNN

Convolutional Neural Network

<https://www.youtube.com/watch?v=ZUu01300850>

Diagram illustrating the architecture of a Convolutional Neural Network (CNN) for image classification. The input image (a cat) is processed through three main layers:

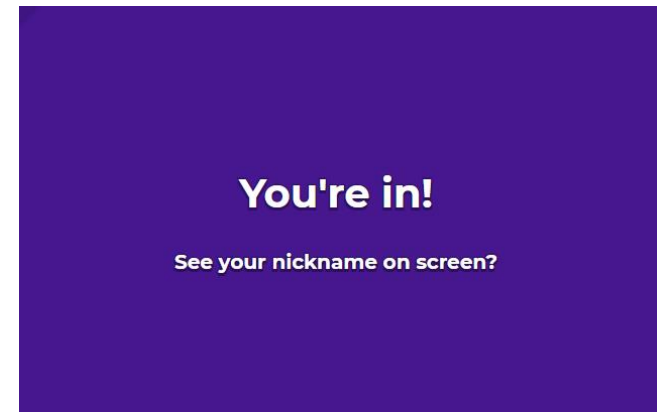
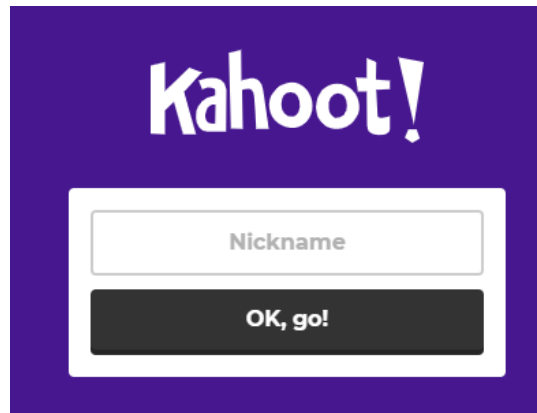
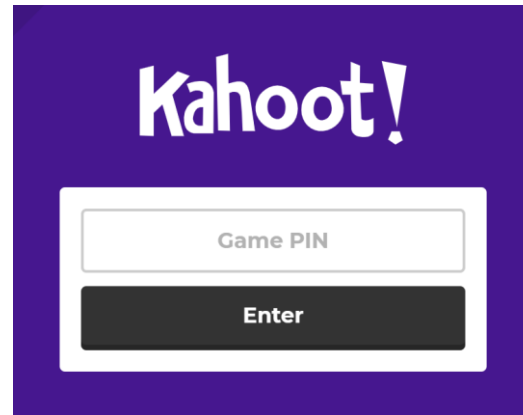
- Convolutional Layer:** The input image is convolved with a set of filters (kernels) to extract local features. This layer is labeled "Convolutional Layer".
- Max Pooling Layer:** The output of the convolutional layer is reduced to its maximum value across the spatial dimensions, resulting in a smaller, more compact representation. This layer is labeled "Max Pooling Layer".
- Fully Connected Layer:** The output of the max pooling layer is flattened and passed through a fully connected layer, which connects every node in one layer to every node in the next. This layer is labeled "Fully Connected Layer".

The final output of the network is a classification result, labeled "Output Layer".

[illegible][illegible]

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

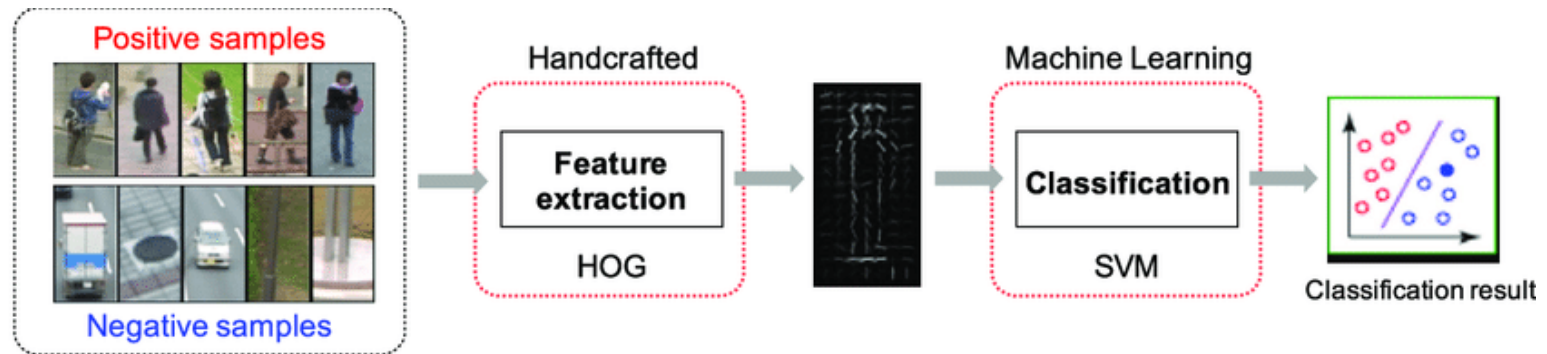
Yann LeCun^{1,2}, Yoshua Bengio³ & Geoffrey Hinton^{4,5}

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.

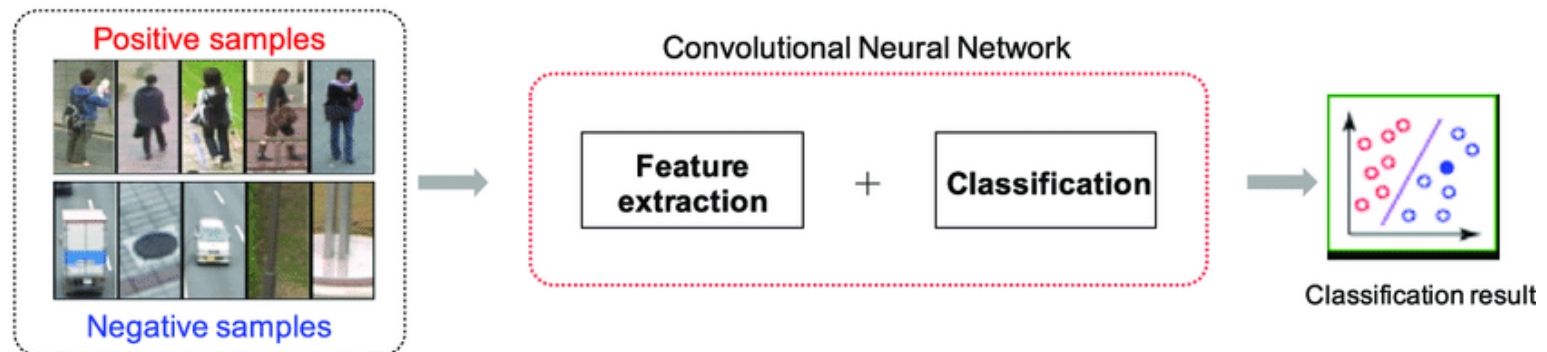
→ Introductory overview and state of the art of deep learning

Introduction

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



Deep Learning Approach



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

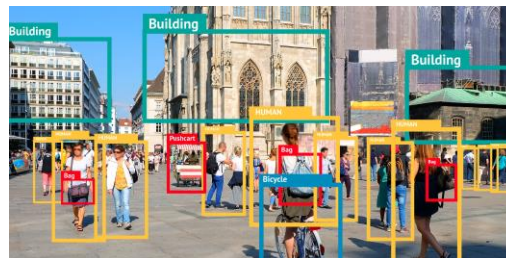


Introduction - Applications

Natural Language Processing



Image Processing

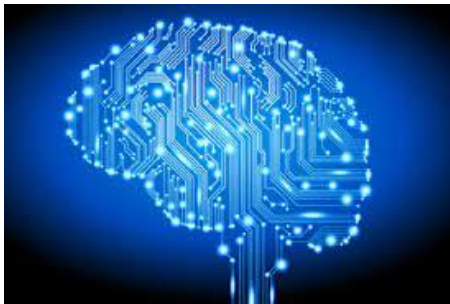


Combinations

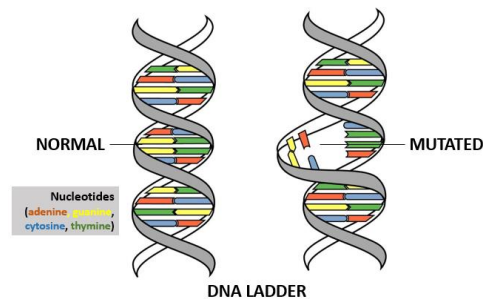


A woman is throwing a **frisbee** in a park.

Brain Circuits



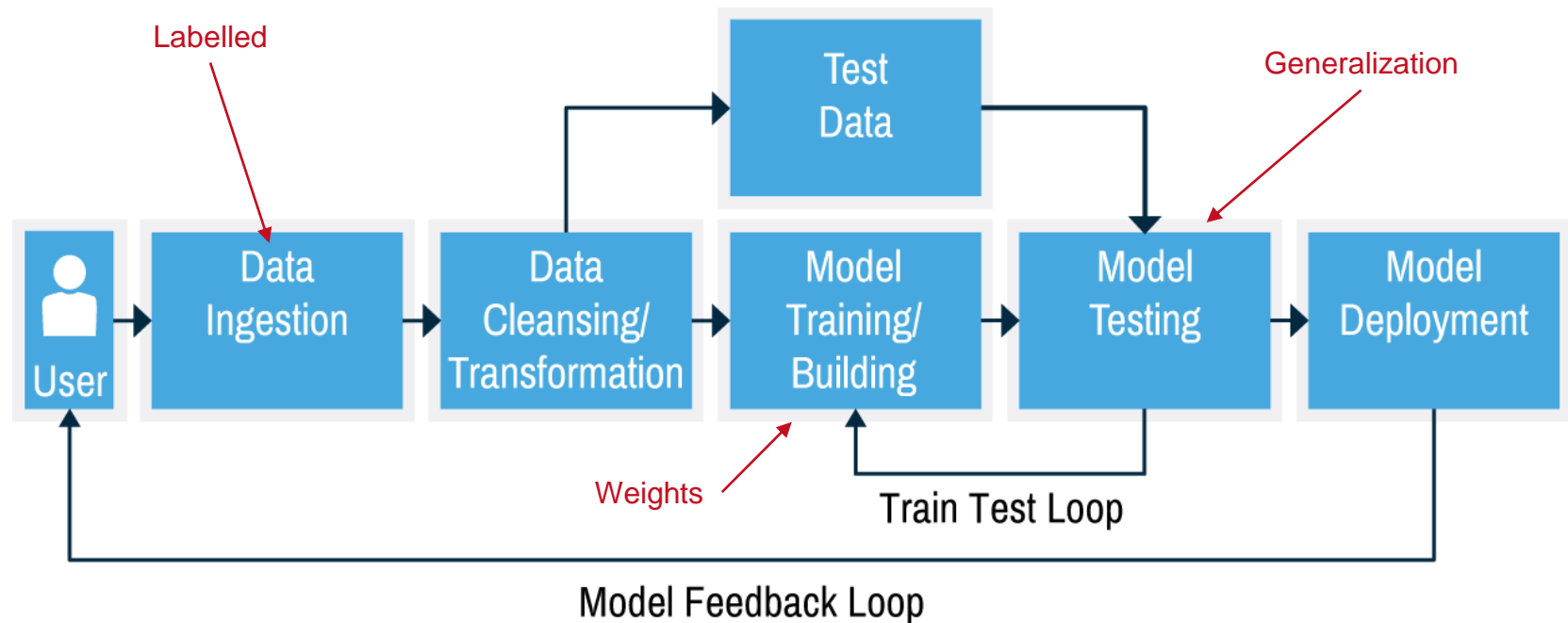
Effects of Mutation



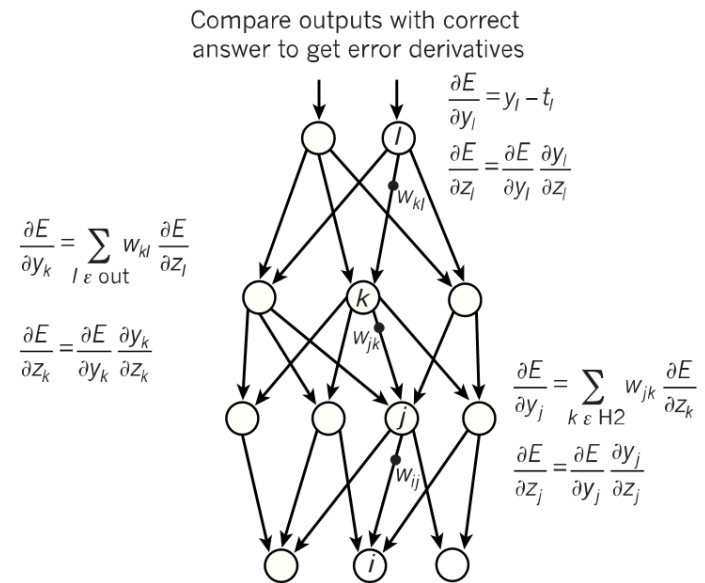
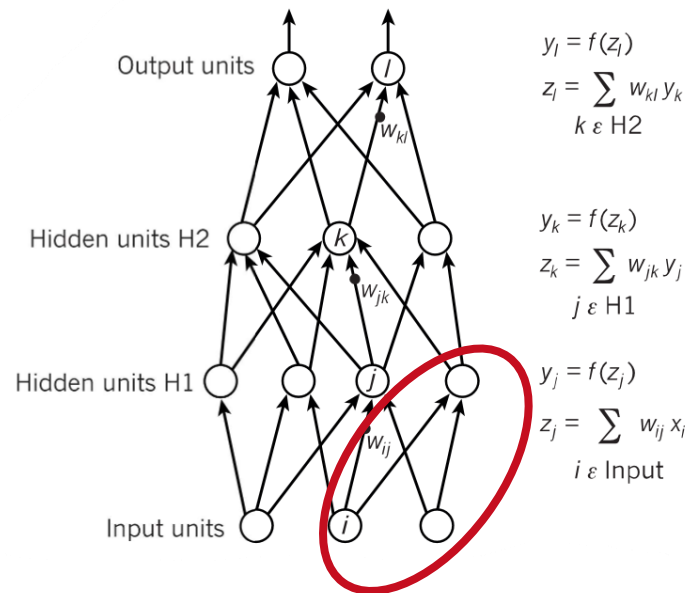
Speech Recognition



Supervised Learning

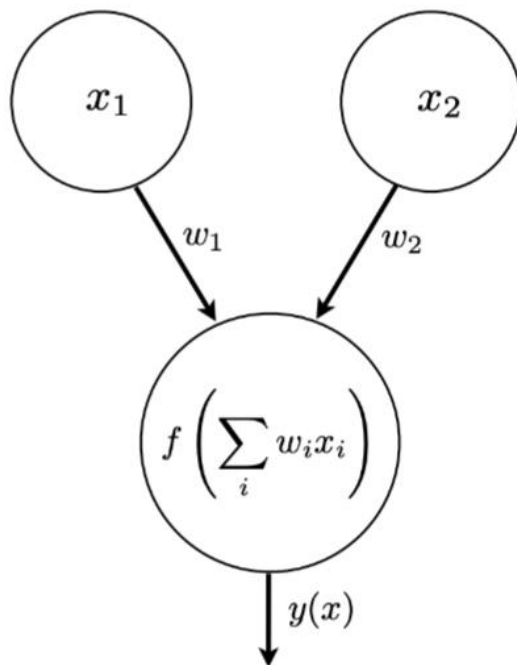


Basics



→ Let's take a step back

Perceptron



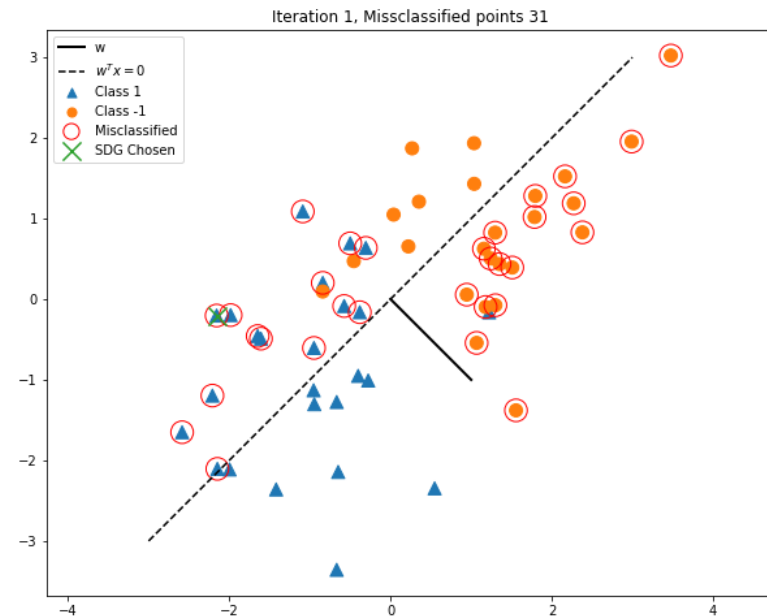
$$\hat{y}(x) = \text{sign}(\mathbf{w}^T \mathbf{x}) = \begin{cases} +1 & \text{Class } \Delta \\ -1 & \text{Class } \circ \end{cases}$$

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

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

For now:

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



→ How do we train a perceptron?

Stochastic Gradient Descent

$$\mathbf{w}^{new} \leftarrow \mathbf{w}^{old} - \eta \frac{\partial E}{\partial \mathbf{w}} \quad \eta : \text{Learning Rate}$$

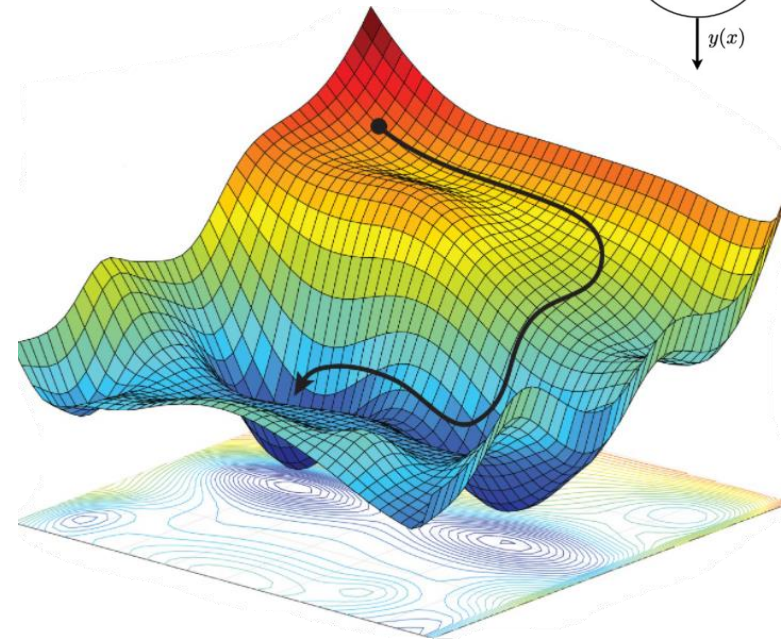
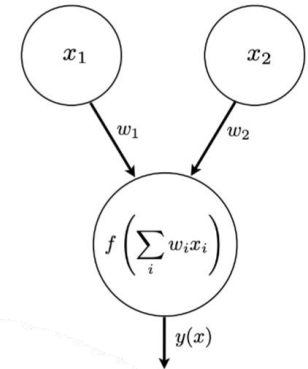
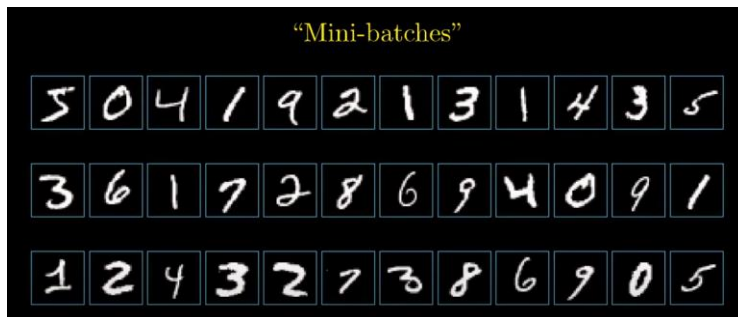
- Need to define error term (cost function):

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

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

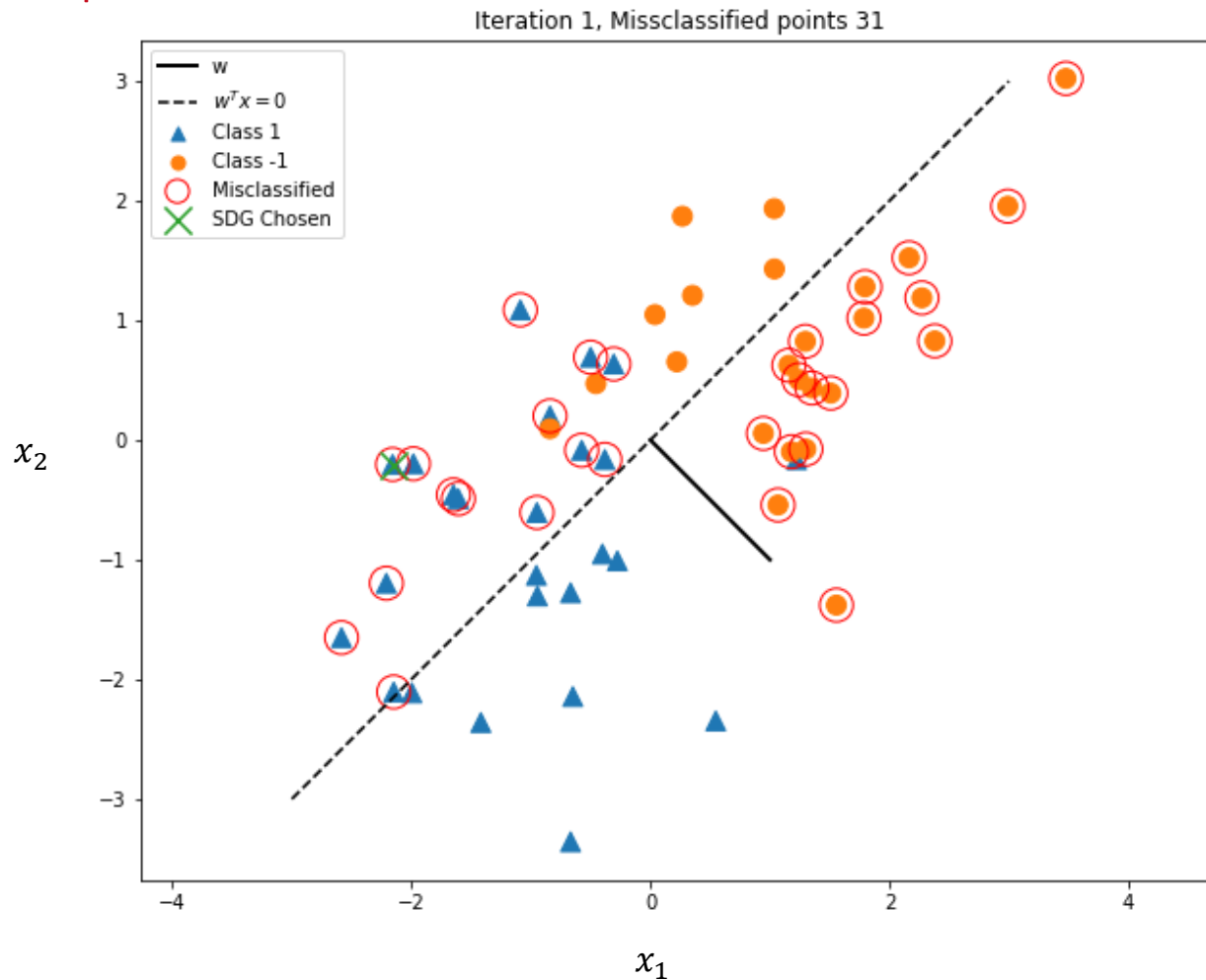
$$E = -\hat{y} y_m = -\mathbf{w}^T \mathbf{x}_m y_m$$

- Stochastic?

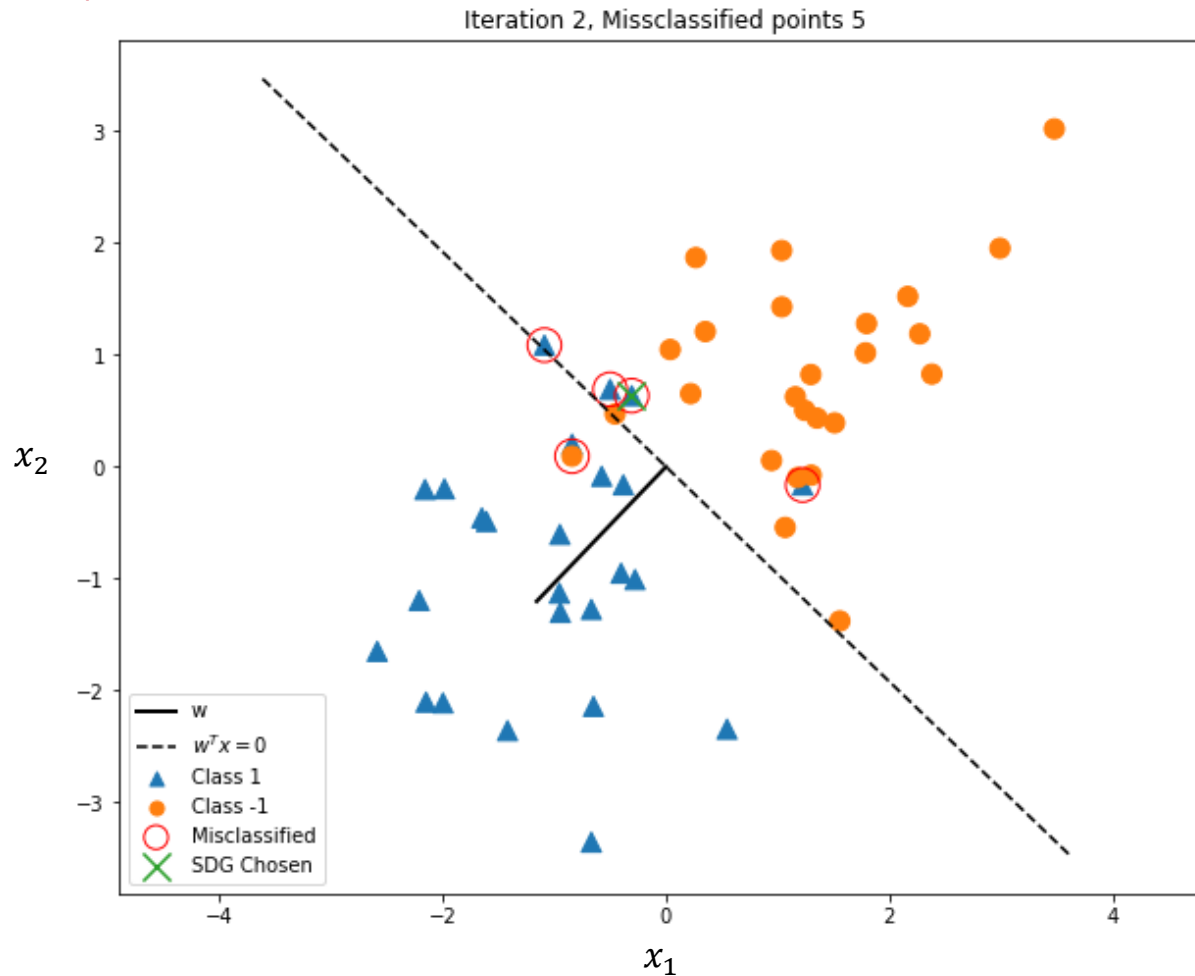


<http://www.its.caltech.edu/~nazizan/papers/SMD.html>

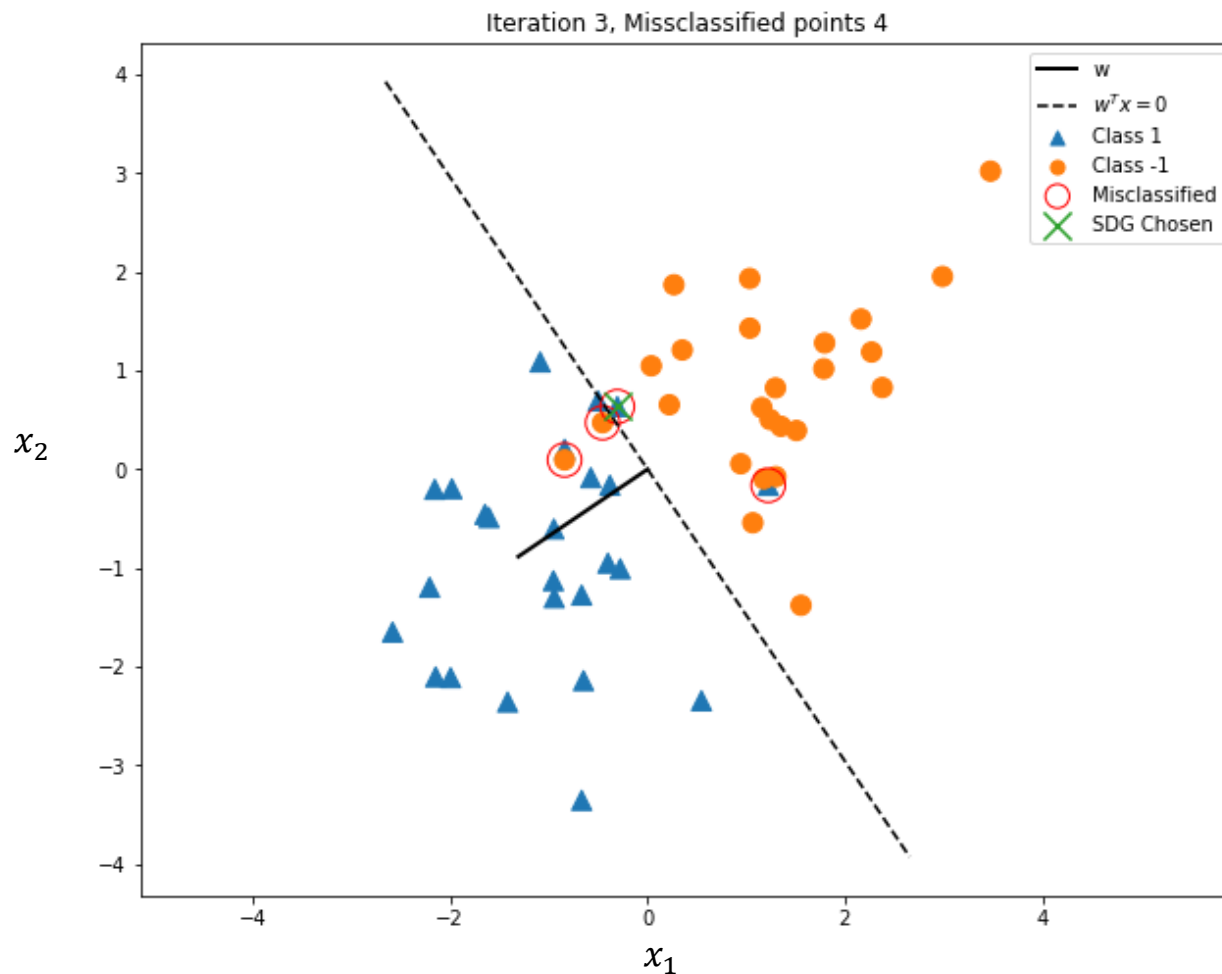
SGD Example Iteration 1



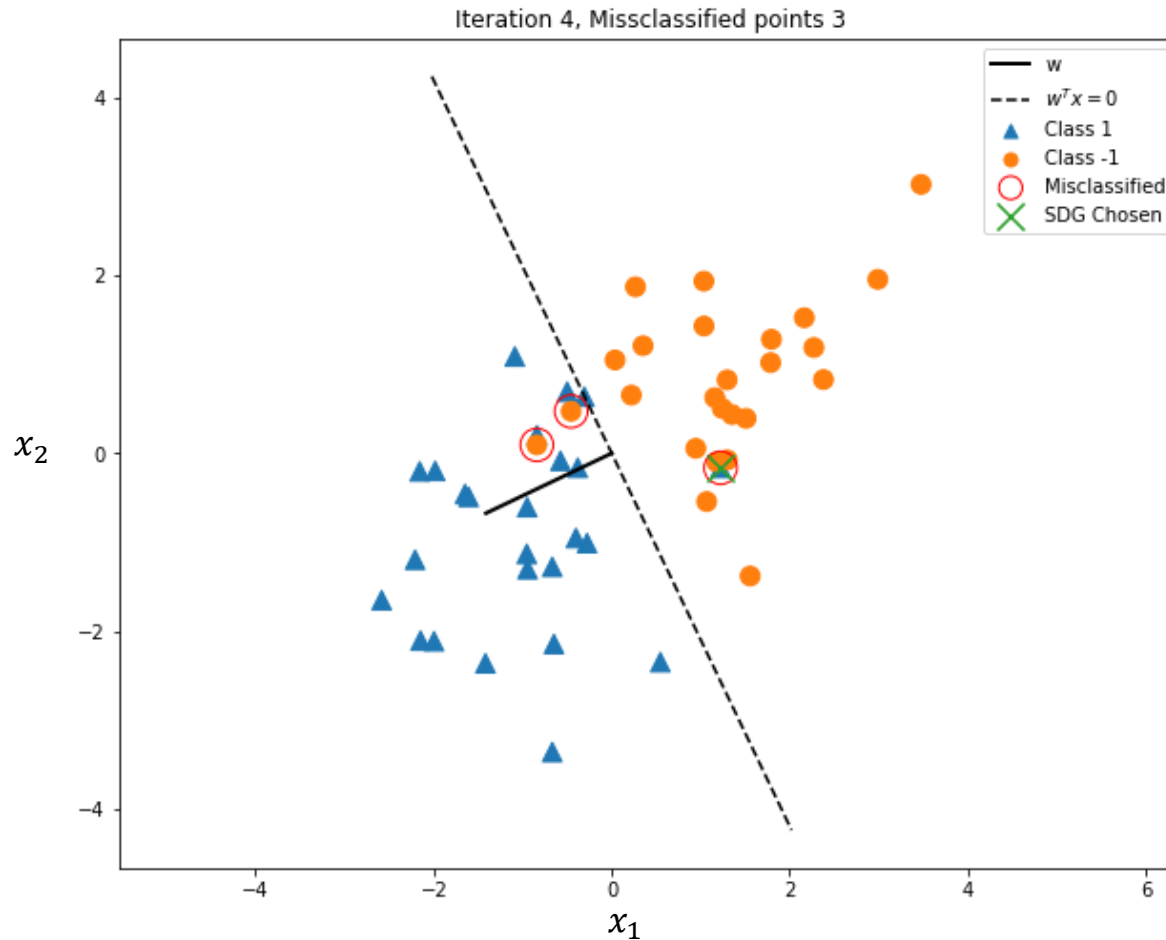
SGD Example Iteration 2



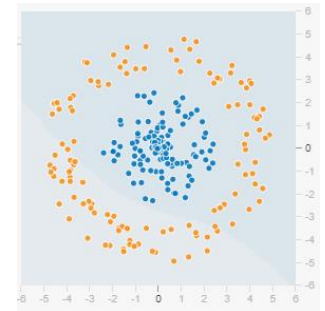
SGD Example Iteration 3



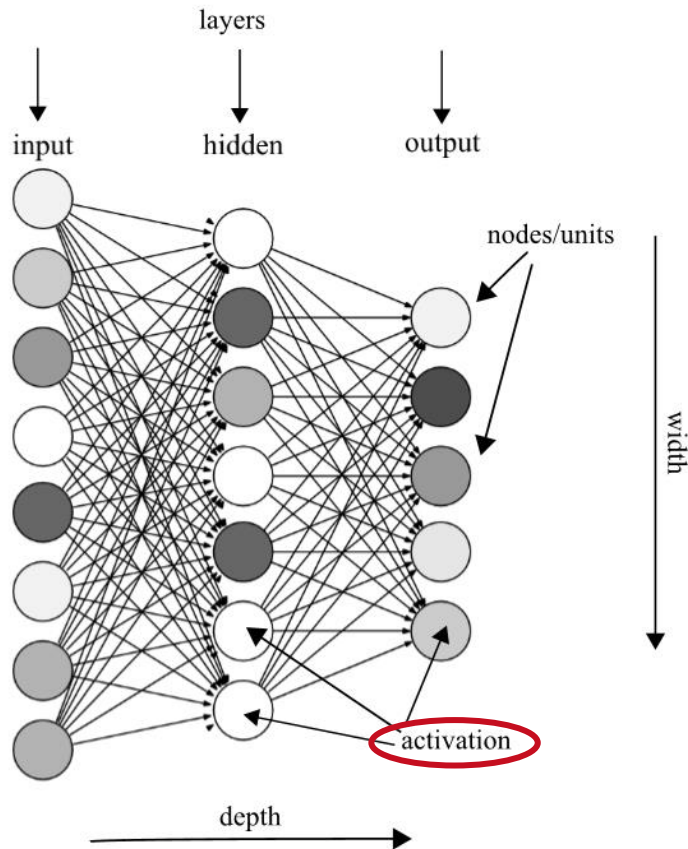
SGD Example Iteration 4



→ Limitations?



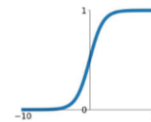
Deep feedforward Networks



Activation Functions

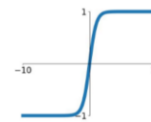
Sigmoid

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



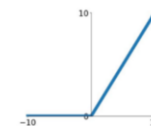
tanh

$$\tanh(x)$$



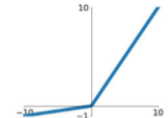
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

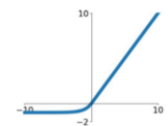


Maxout

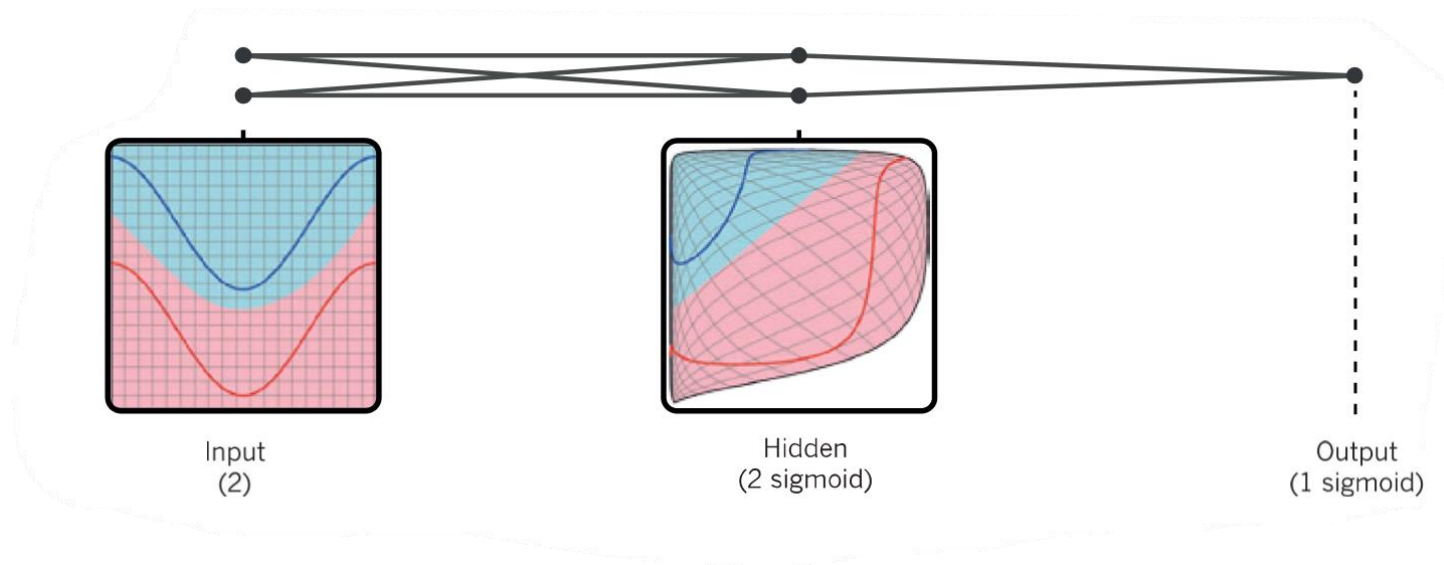
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



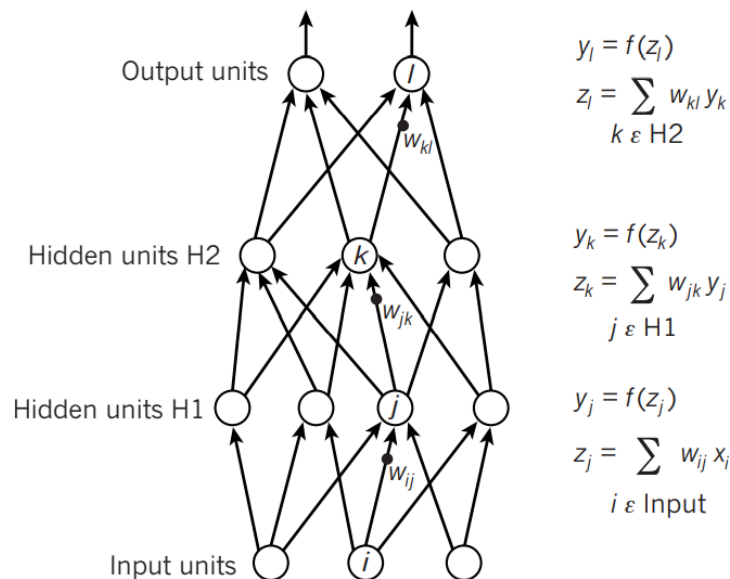
Deep feedforward Networks



Neural Network Training

1 Forward pass: Apply data with current weights

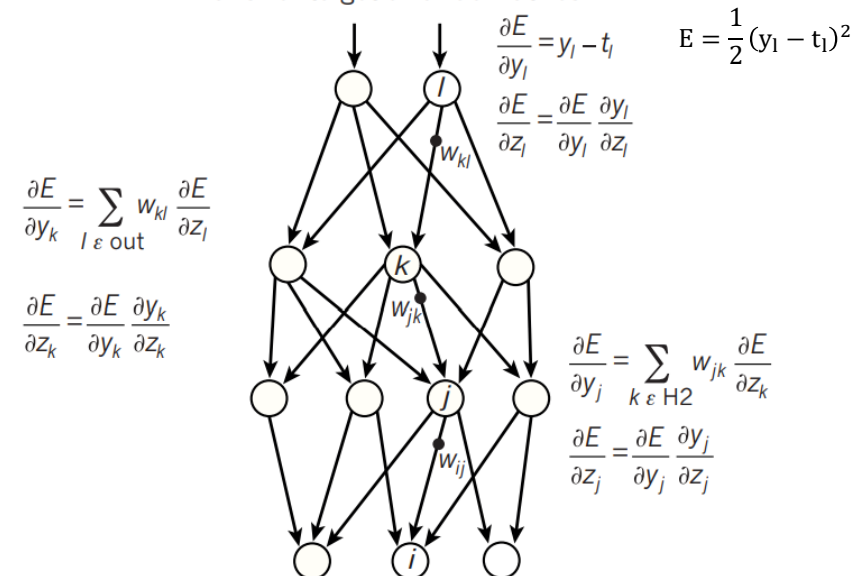
c



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

d

Compare outputs with correct answer to get error derivatives



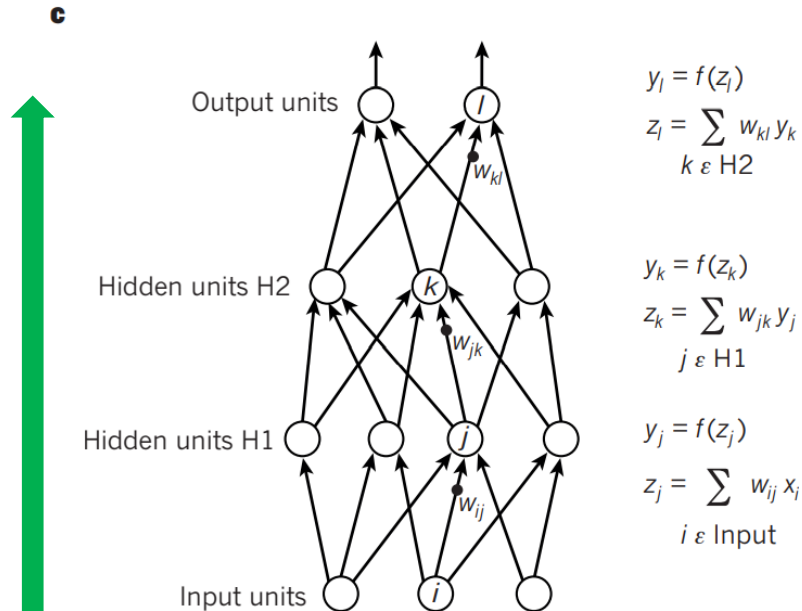
3 Update weights:

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

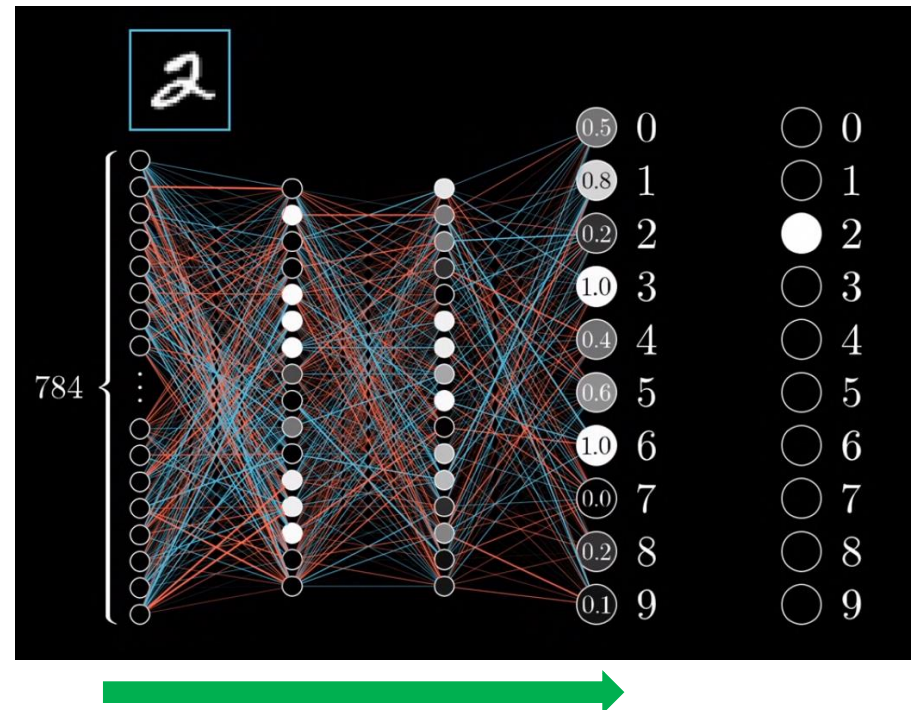
Neural Network Training

1 Forward pass: Apply data with current weights

c



0	2	15	0	0	11	10	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	55	61	32	0	0
0	10	16	115	238	255	244	245	243	250	249	255	222	103	10
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124
2	95	255	228	255	251	254	211	141	116	127	215	251	238	255
13	217	243	255	153	33	226	62	2	0	10	13	232	255	255
16	229	252	254	149	12	0	0	7	7	0	10	237	252	235
6	141	245	255	212	25	11	9	3	0	115	238	243	255	17
0	87	252	250	248	215	60	0	1	172	252	255	248	147	6
0	13	115	255	255	245	255	182	181	248	252	242	208	36	0
1	0	5	117	251	255	241	255	247	255	241	165	17	0	7
0	0	0	4	55	251	255	246	254	253	255	170	11	0	1
0	0	4	107	255	255	255	248	252	255	244	255	182	10	0
0	22	206	252	246	251	241	1100	24	113	255	245	255	194	9
0	111	255	242	255	115	24	0	0	6	39	255	232	230	56
0	218	251	250	117	7	11	0	0	0	2	65	255	250	125
0	173	255	255	101	9	20	0	13	3	15	182	251	245	61
0	107	251	241	255	230	98	55	19	110	217	248	253	255	52
0	18	148	250	255	247	255	255	255	249	255	240	255	129	0
0	0	23	115	215	255	250	248	255	255	248	248	11	14	12
0	0	6	1	0	82	153	233	255	252	147	37	0	0	4
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0



<https://www.youtube.com/watch?v=llg3gGewQ5U&t=740s>

Neural Network Training

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

d

Compare outputs with correct answer to get error derivatives

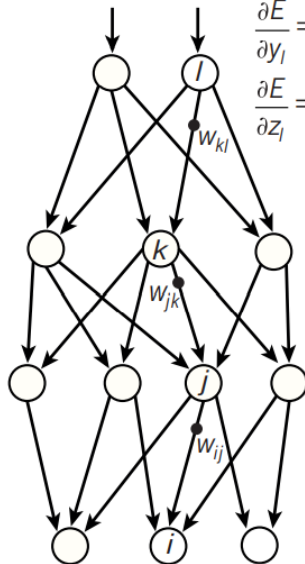
$$E = \frac{1}{2} (y_l - t_l)^2$$

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

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

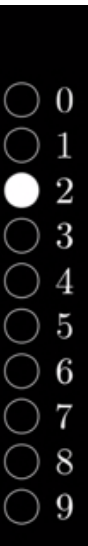
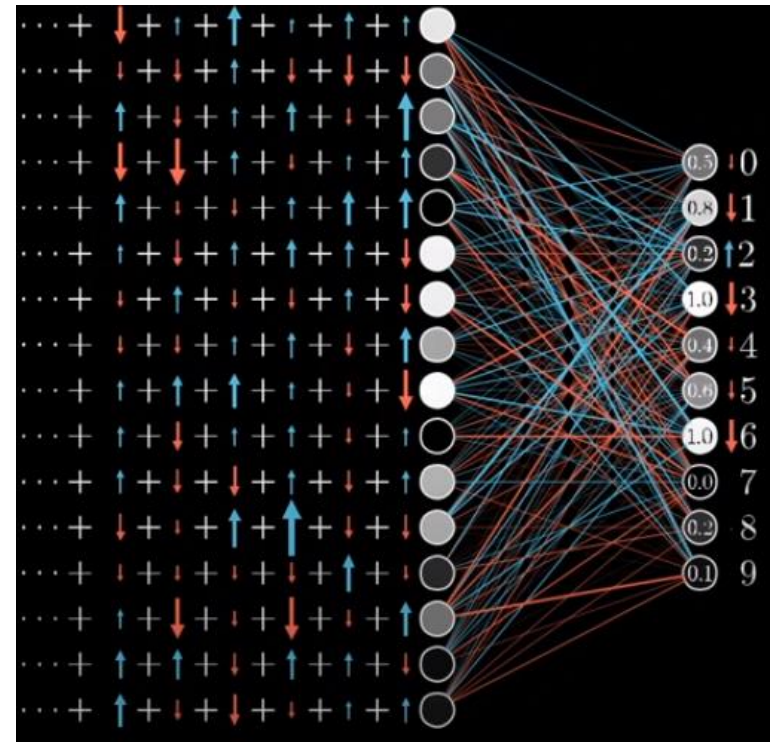
$$\frac{\partial E}{\partial y_k} = \sum_{l \in \text{out}} w_{kl} \frac{\partial E}{\partial z_l}$$

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



$$\frac{\partial E}{\partial y_j} = \sum_{k \in H2} w_{jk} \frac{\partial E}{\partial z_k}$$

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

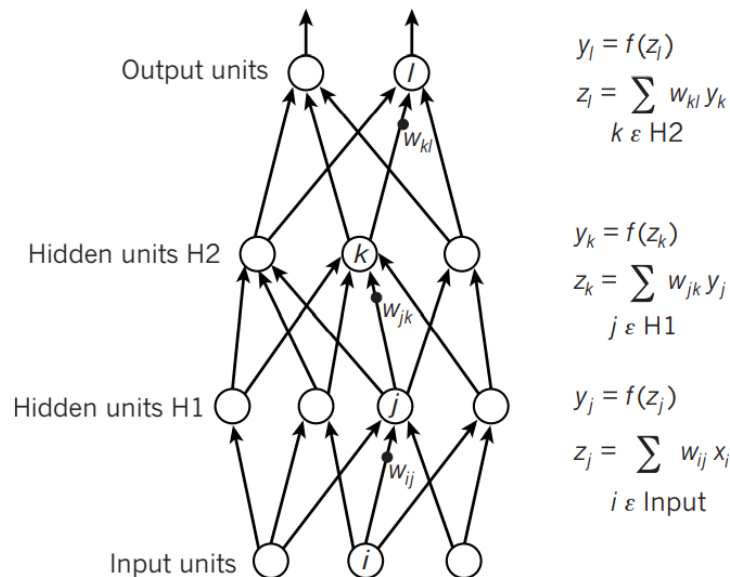


<https://www.youtube.com/watch?v=llg3gGewQ5U&t=740s>

Neural Network Training

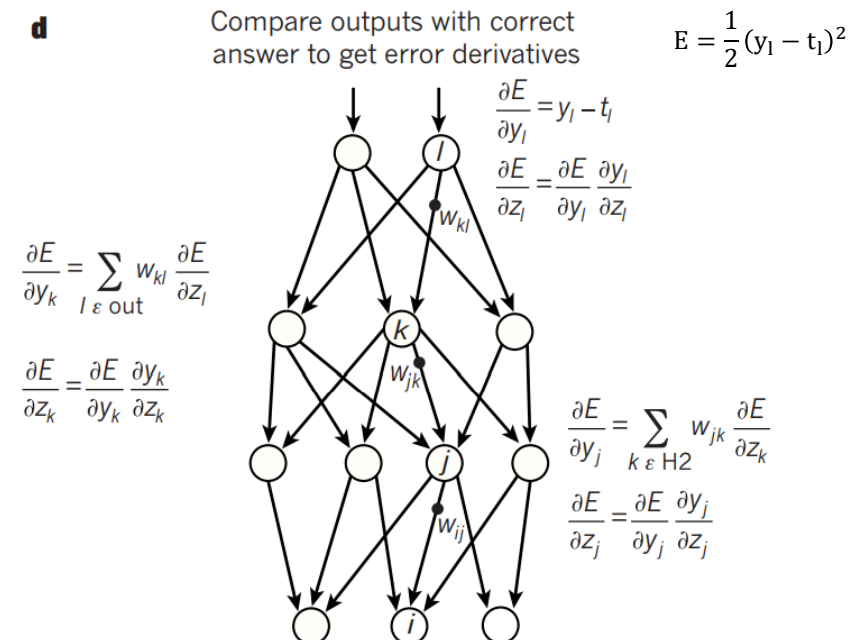
1 Forward pass: Apply data with current weights

c



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

d

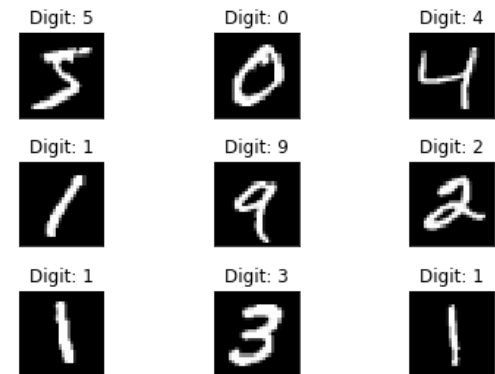
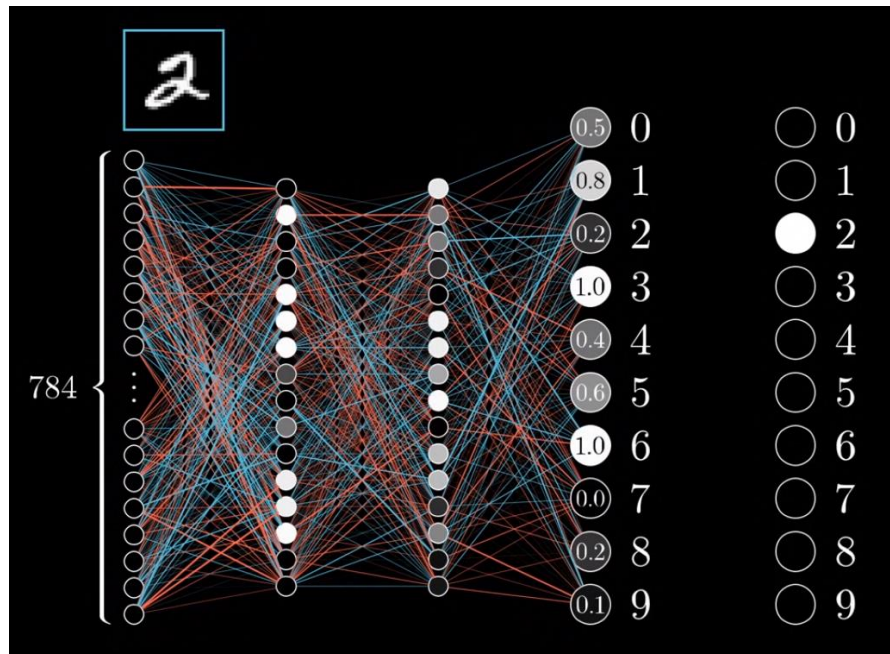


3 Update weights:

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

Neural Network – MNIST

→ MNIST: The hello world of Neural Networks



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

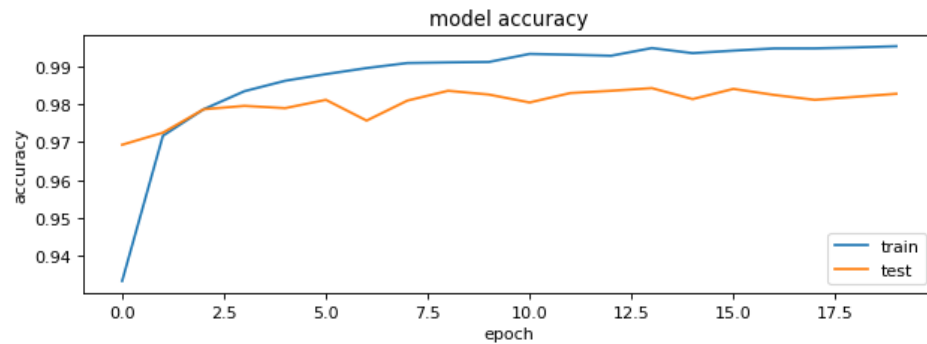
1st time: **784** **784** **10**

2nd time: **10** **10**



Neural Network – MNIST

1st Model – 1568 hidden layer nodes



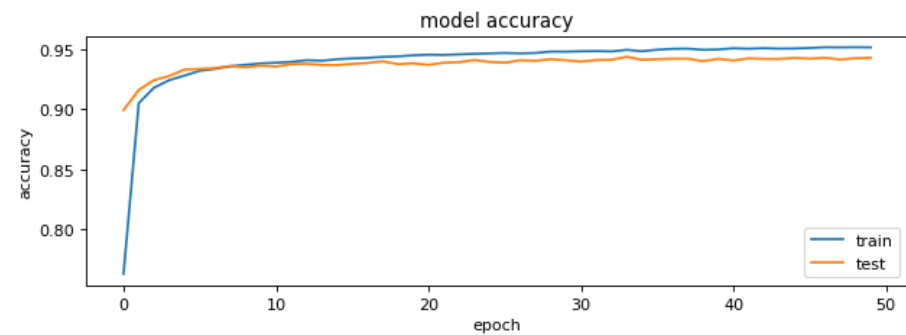
Predicted: 4, Truth: 4 Predicted: 9, Truth: 9 Predicted: 5, Truth: 5



Predicted 9, Truth: 4 Predicted 9, Truth: 2 Predicted 6, Truth: 4



2nd Model – 10 hidden layer nodes



Predicted: 4, Truth: 4 Predicted: 9, Truth: 9 Predicted: 9, Truth: 9



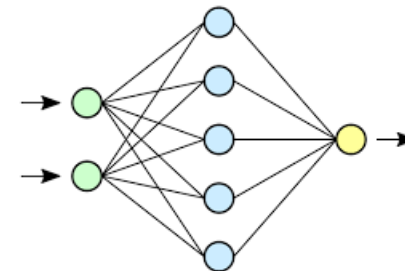
Predicted 6, Truth: 5 Predicted 4, Truth: 6 Predicted 5, Truth: 3



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/>



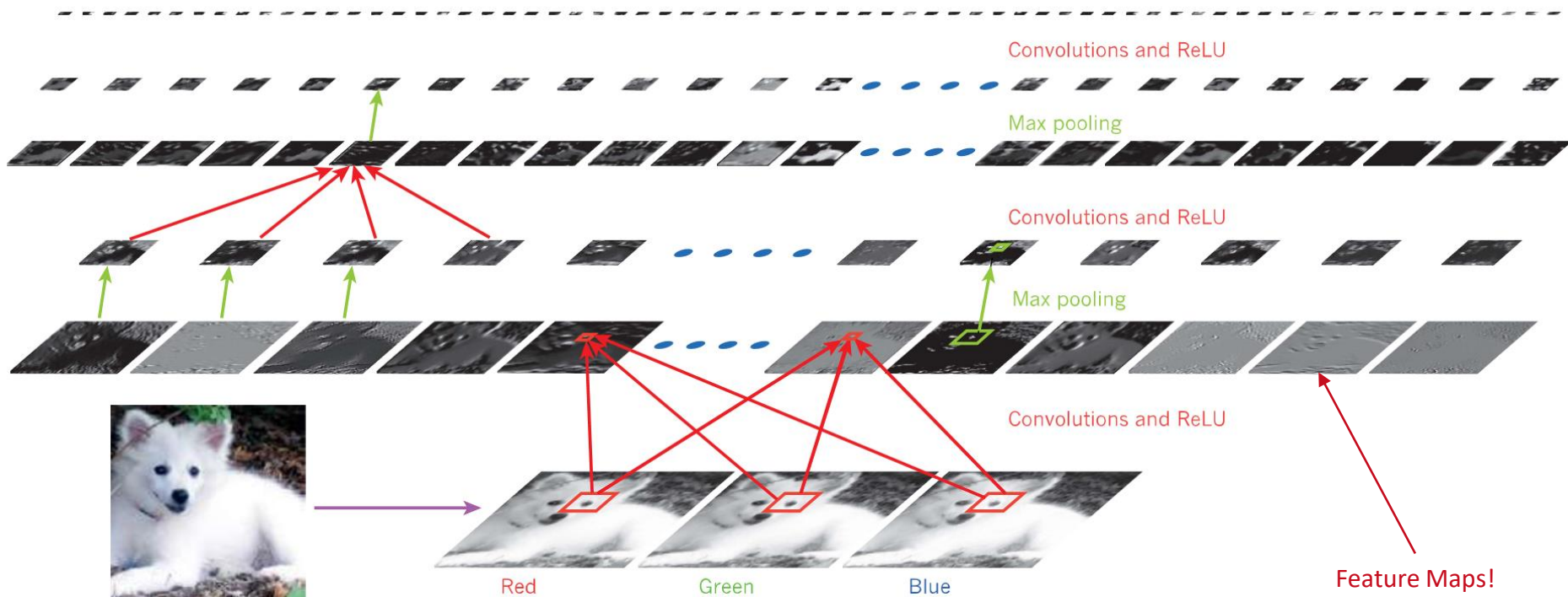
“(...) no one knows why exactly this works, but the idea is that by pre-training you start from more favorable regions of feature space.”

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

What about the computer vision community? → Convolutional neural networks

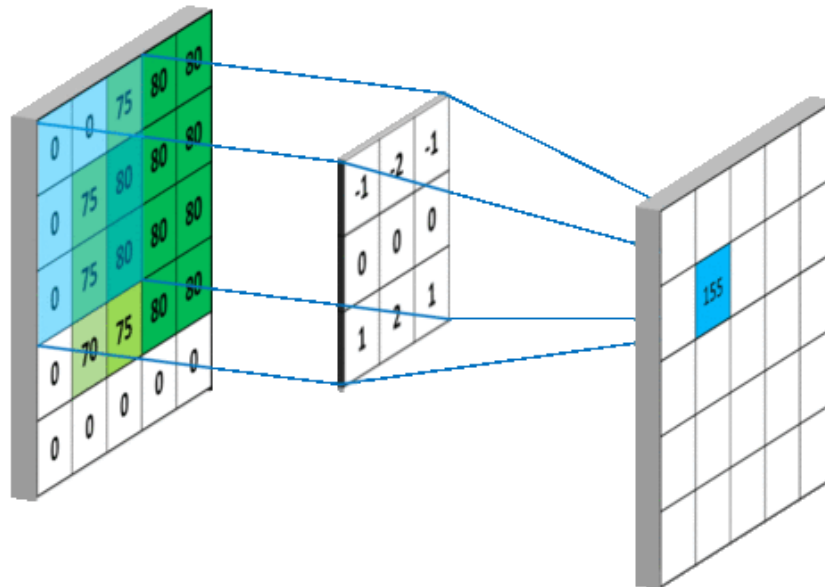
Convolutional Neural Network

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic fox (1.0); Eskimo dog (0.6); white wolf (0.4); Siberian husky (0.4)



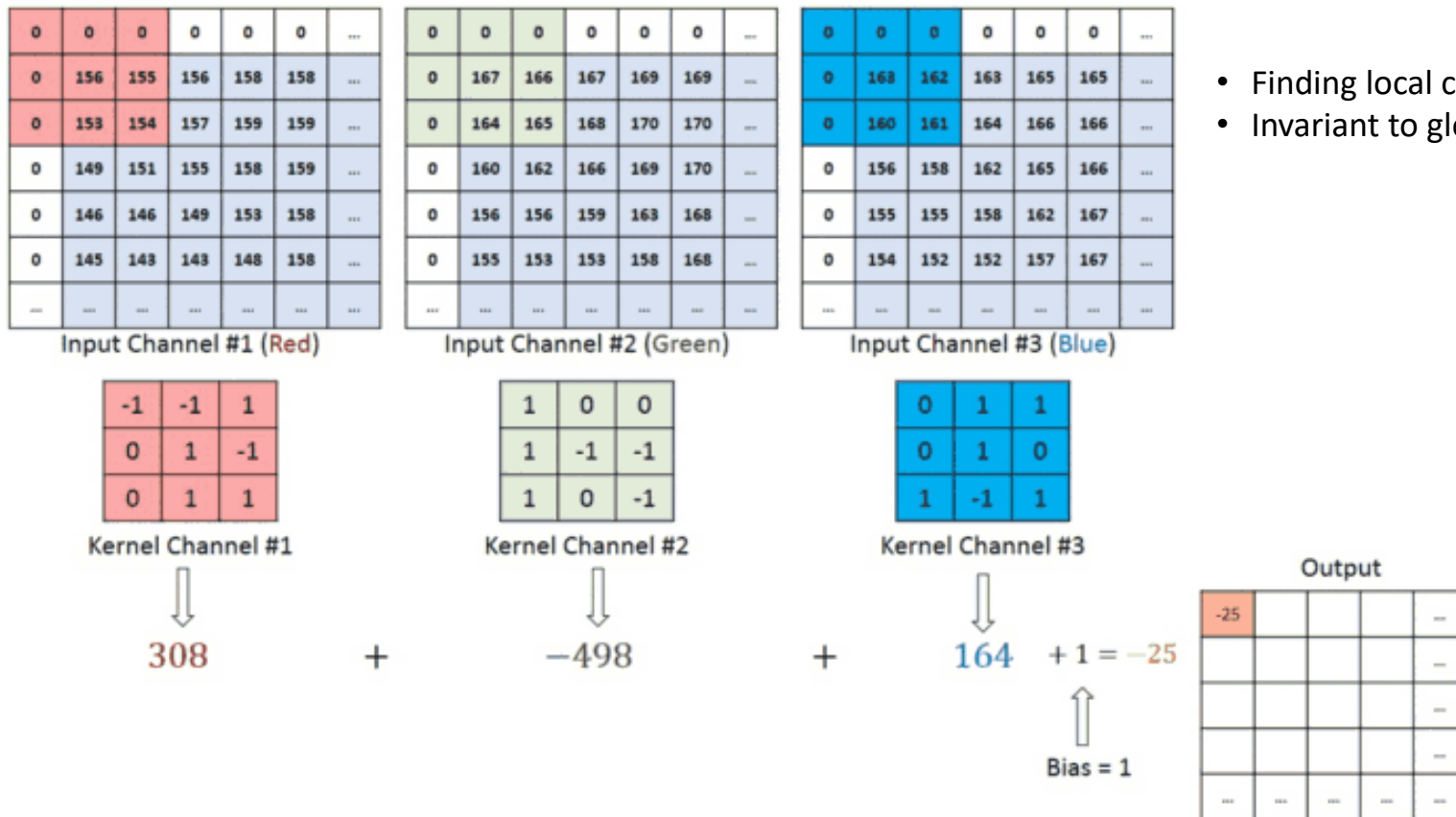
Convolutional Neural Network

Convolution with a 3x3 filter:



Convolutional Neural Network

Convolution with a 3x3x3 Filter:

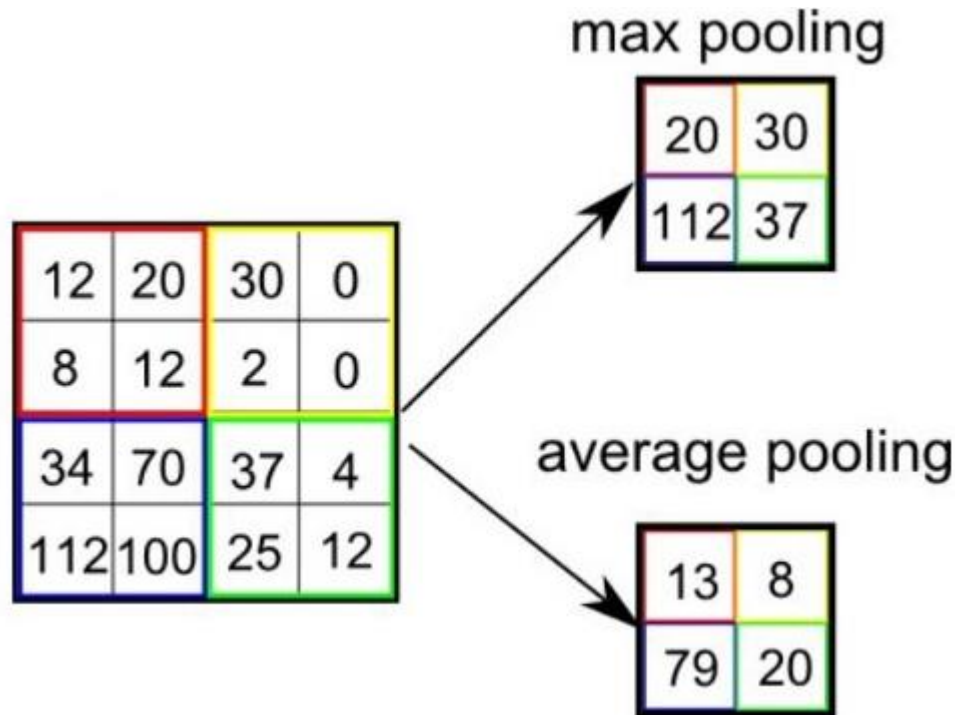


- Finding local connections
- Invariant to global location

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

Convolutional Neural Network

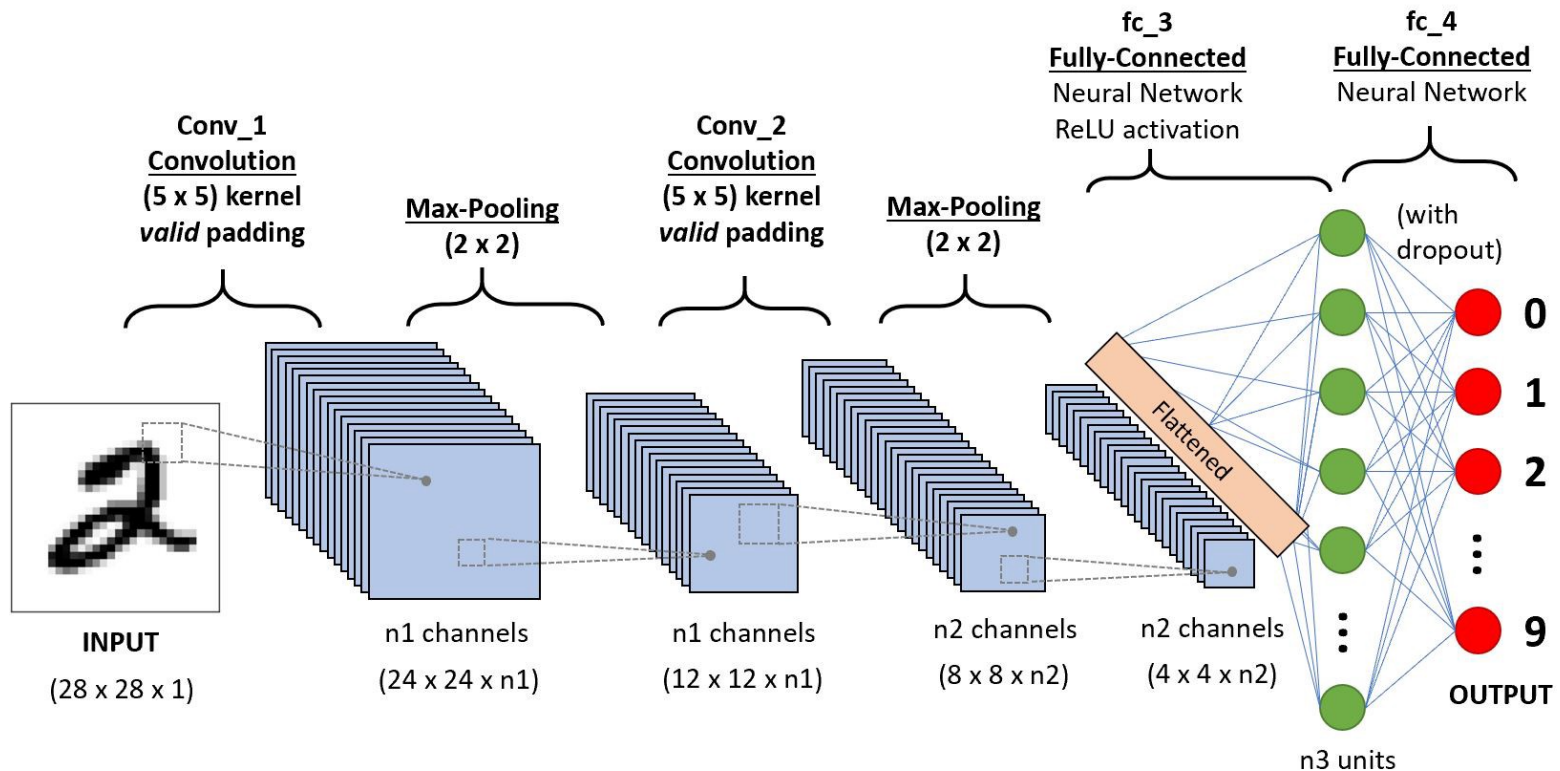
Max and Average Pooling:



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

Convolutional Neural Network

Fully connected Layer:



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

Convolutional Neural Network

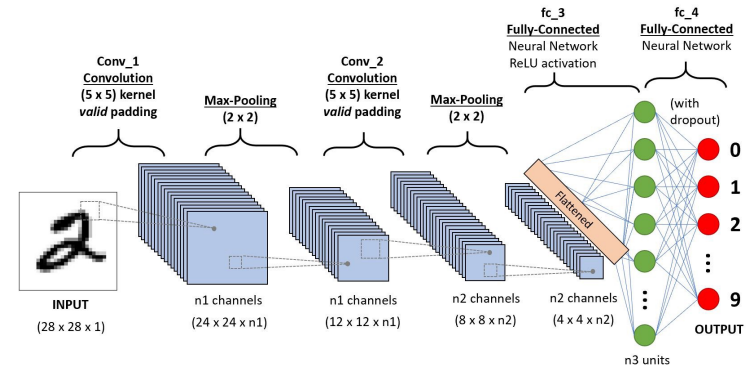
Why does it work so well?

- Higher level features are obtained by composing lower level one
- Inspired by visual neuroscience

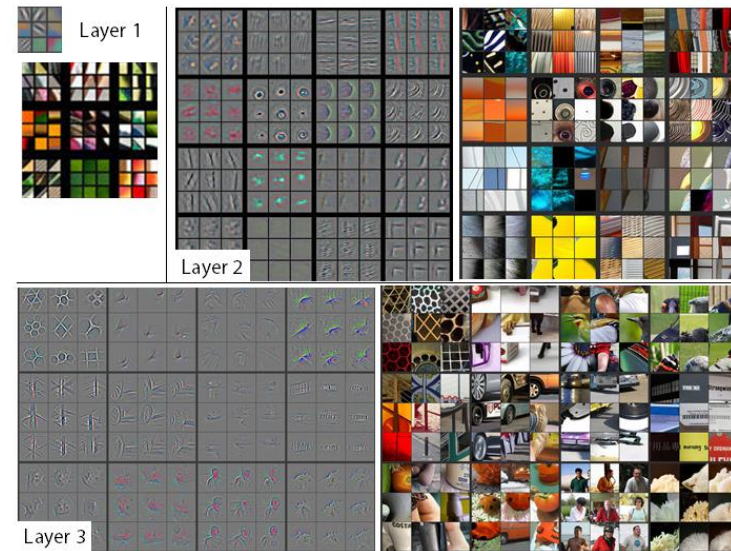
Breakthrough?

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

→ What about natural language processing?



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



Distributed representation – Word Vectors

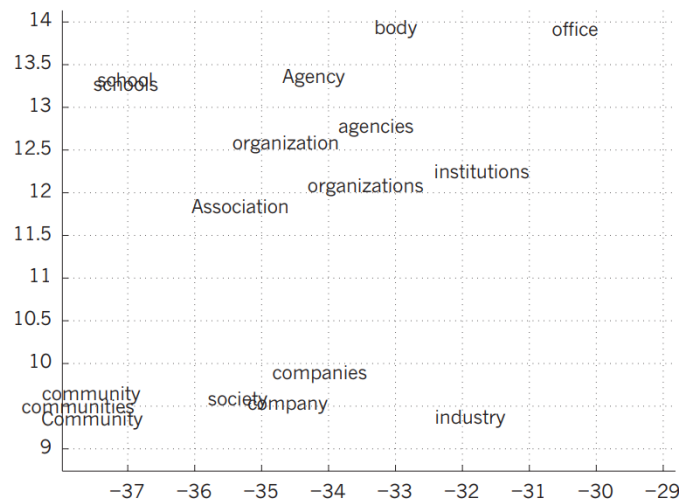
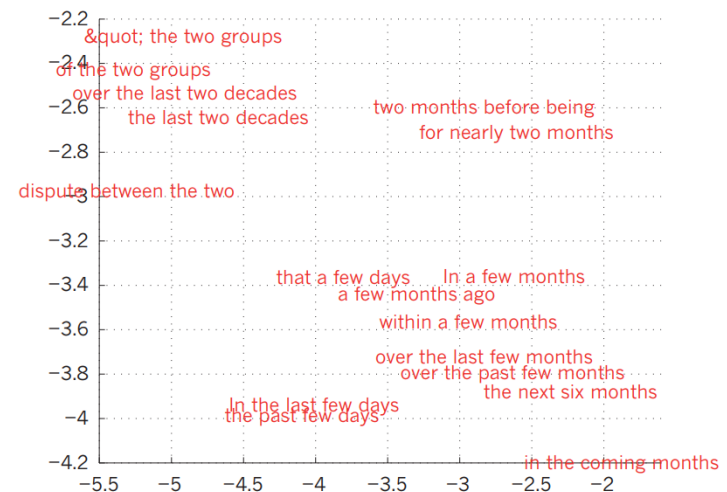
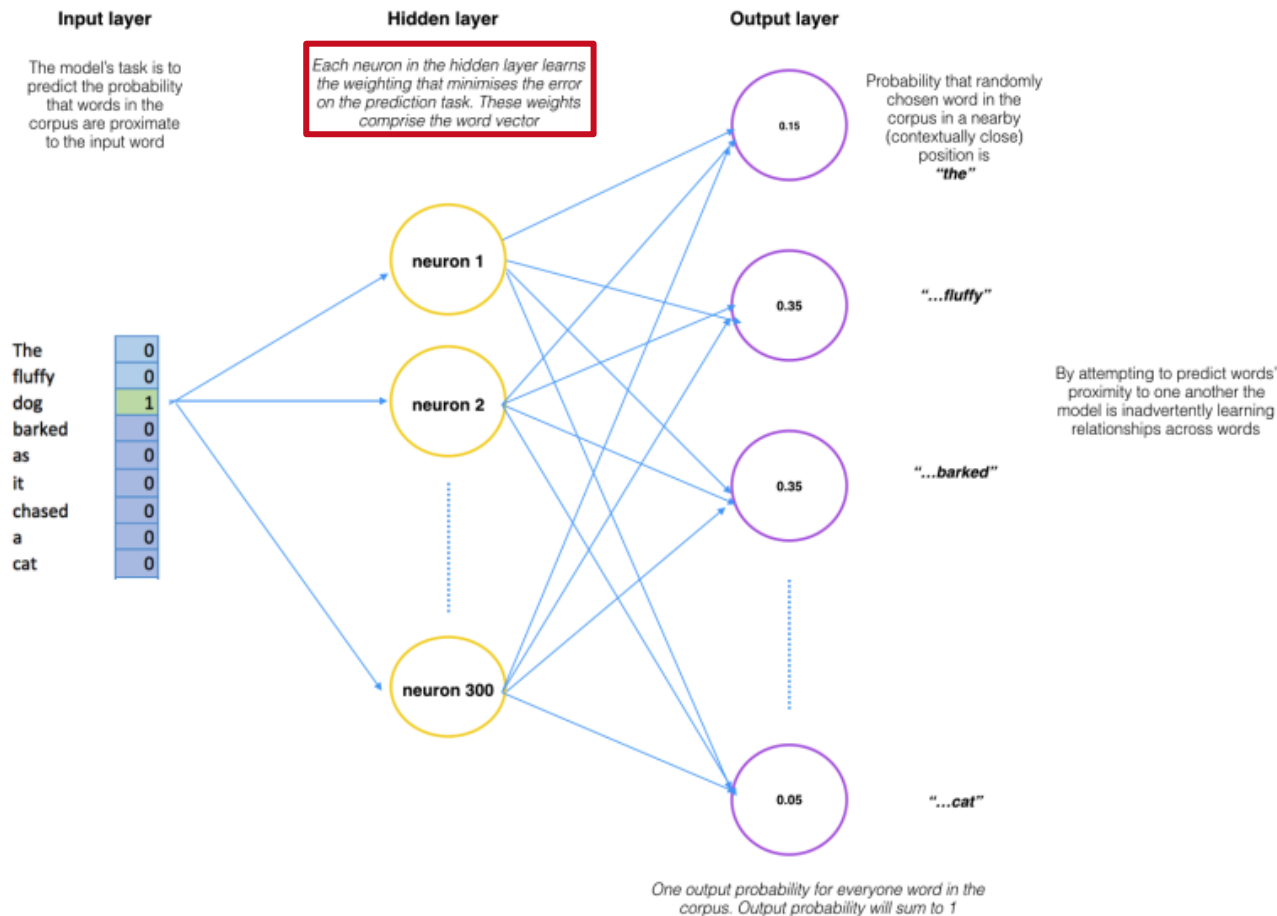


Figure 4 | Visualizing the learned word vectors. On the left is an illustration of word representations learned for modelling language, non-linearly projected to 2D for visualization using the t-SNE algorithm¹⁰³. On the right is a 2D representation of phrases learned by an English-to-French encoder-decoder recurrent neural network⁷⁵. 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

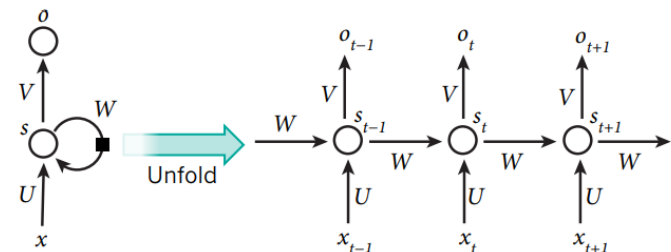


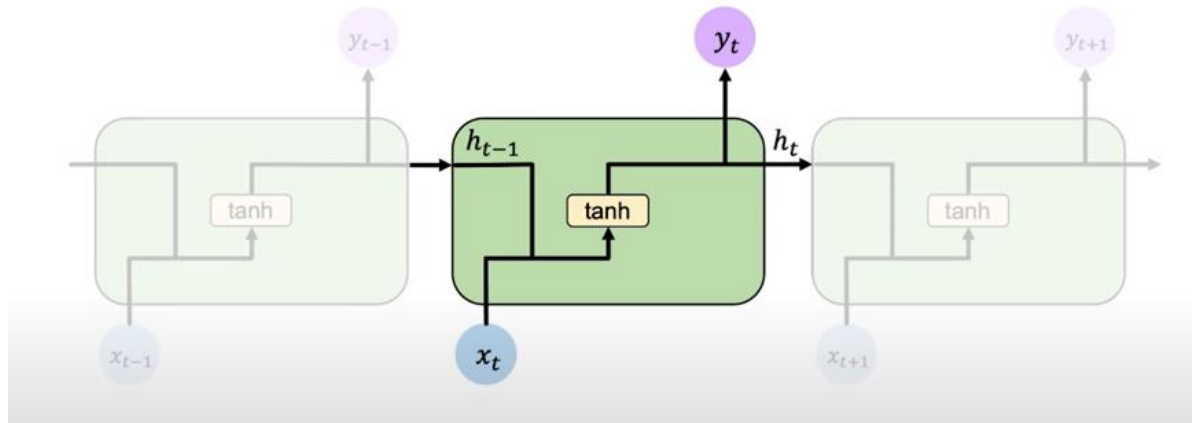
Figure 5 | A recurrent neural network and the unfolding in time of the computation involved in its forward computation. The artificial neurons

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

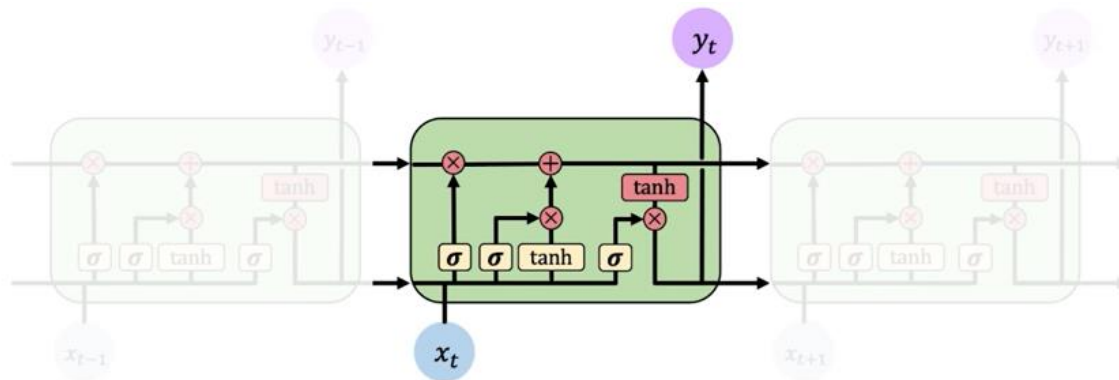
Recurrent Neural Networks - LSTM

→ 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 lesson!**

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

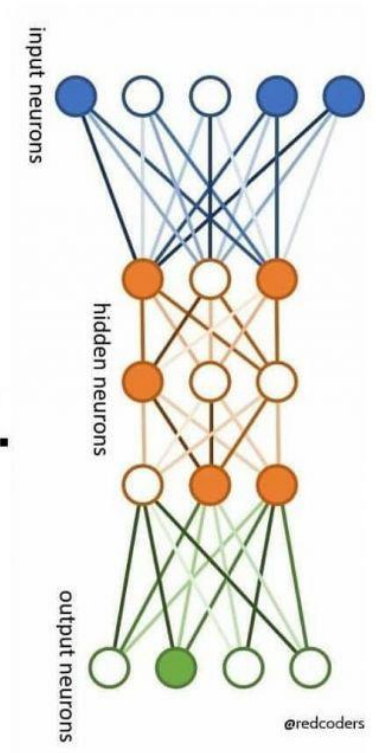
Questions?



**THIS IS A NEURAL
NETWORK.**

**IT MAKES MISTAKES.
IT LEARNS FROM THEM.**

**BE LIKE A NEURAL
NETWORK.**



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