Course: Deep Learning

Deep Neural Networks

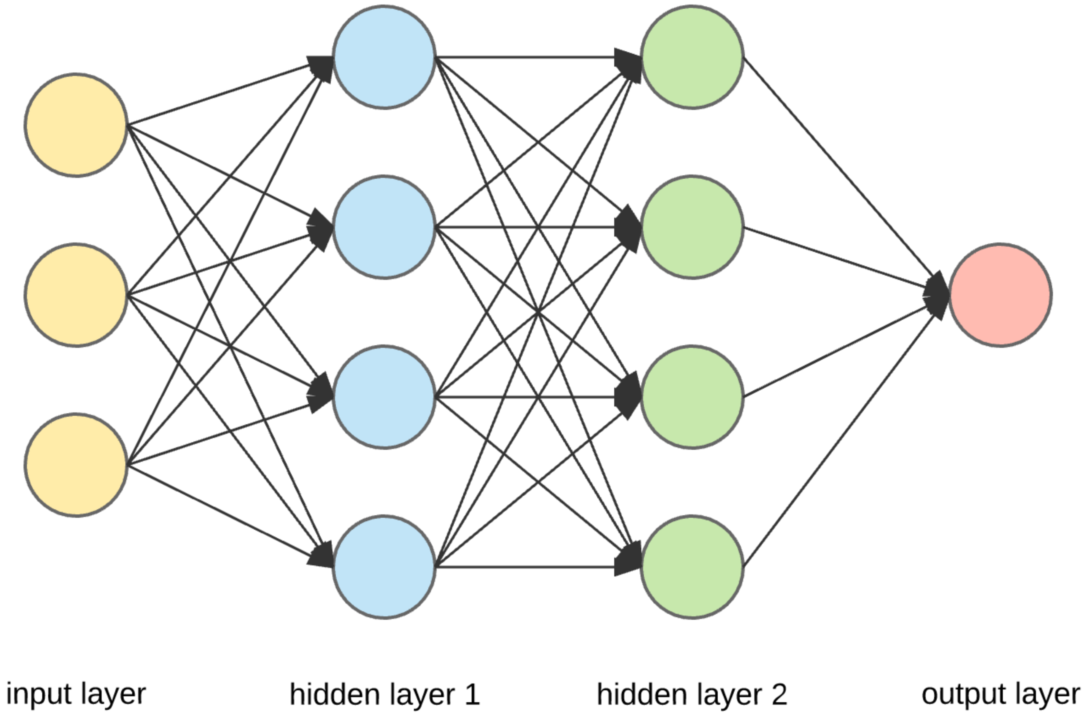
Daniel Manrique

2025

Scaling-up: from shallow to deep

- We could make the one-hidden layer NN more powerful by adding more hidden neurons.
- In fact, one-hidden layer NN can approximate functions with an arbitrarily low error.
  - Just one level of abstraction.
  - Fitting large datasets is very hard with shallow NN.
- We can rather increase the number of layers:
  - Multiple layers of abstraction to progressively extract higher-level features from the raw input and pick out which features improve performance.
  - Deep NN can achieve better accuracy than shallow NN.
- Deep learning comes into play to solve the difficulties arisen from training deep neural networks.

Shallow and deep



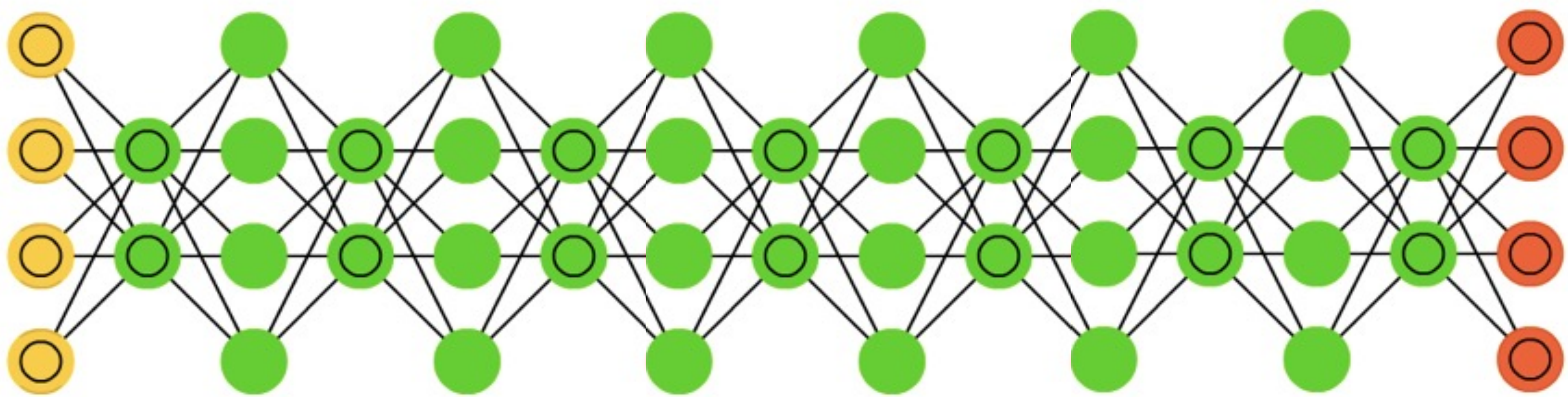
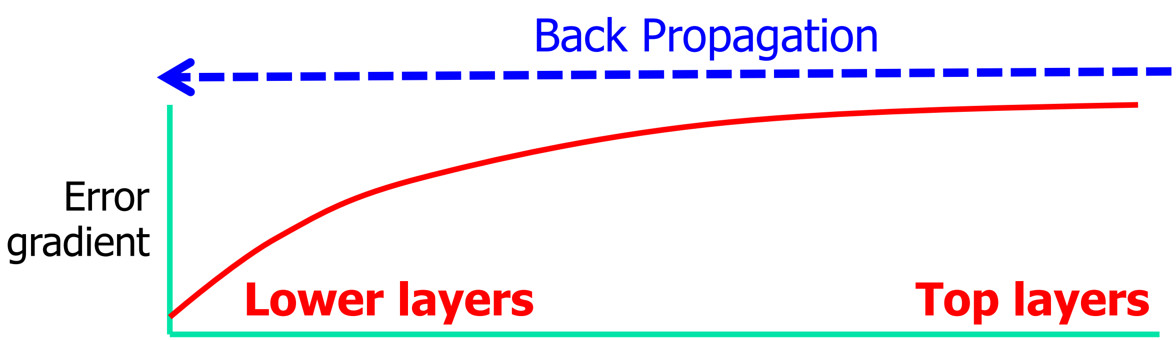


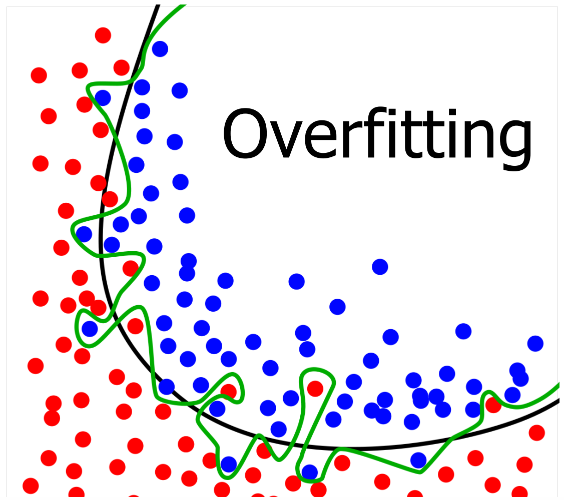
Image by http://www.asimovinstitute.org/neural-network-zoo/

Deep learning

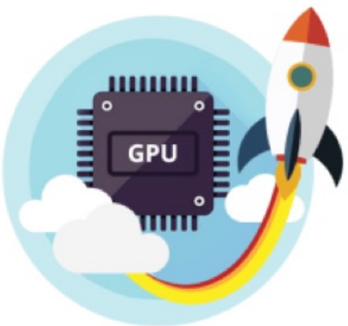
- Deep learning comes into play to solve the difficulties arisen from training deep neural networks:

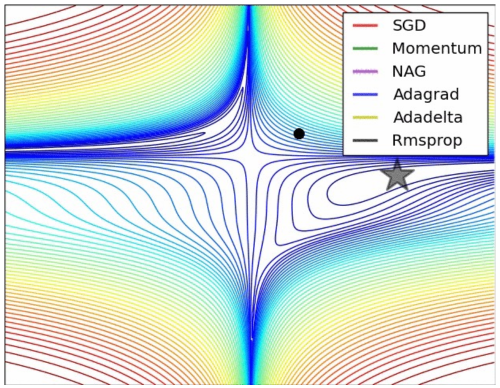
The vanishing gradients problem

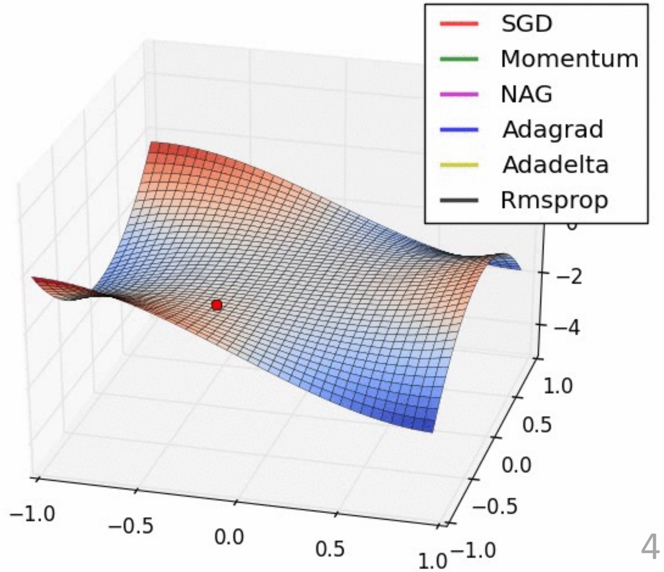




Intensive vector computation: speed up training.







Enhanced ANNs + Big Data + Computational Resources
Deep learning

Enhanced algorithms: relatively small tweaks with huge positive impact.

UC Irvine Machine Learning Repository

UCI Machine Learning Repository

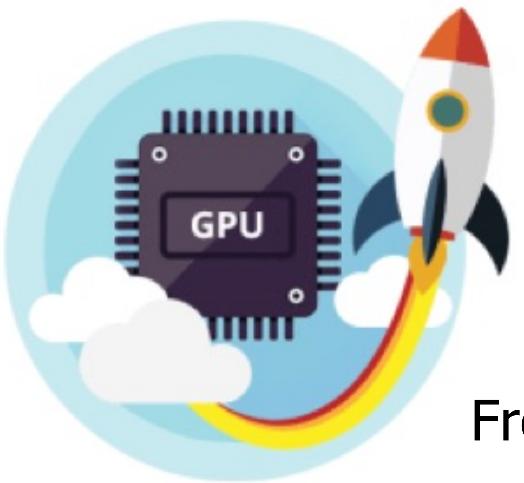
Access to big data: internet.

kaggle

OpenAI

OpenAI Gym BETA

Gymnasium



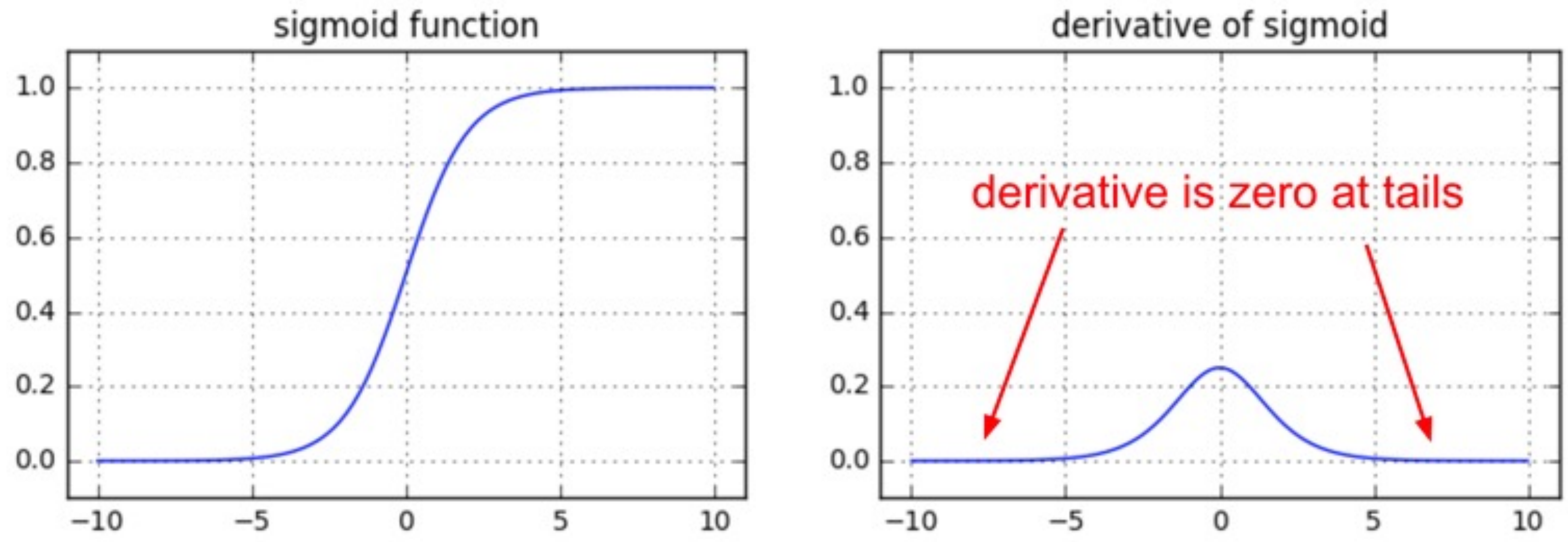
From 10^5 calculations per second per 1000$ to 10^{15}

The vanishing gradients problem

- Cumulative back-propagated error signals shrink rapidly. They decay exponentially in the number of layers. The result is that the final trained network converges to a poor local minimum.
- Different layers learn at very different speeds.

The vanishing gradients problem: sigmoid and tanh

- If near output layers are saturated at -1, 0, or 1, the asymptotes of the tanh (sigmoid) function, near input layers have gradients of nearly 0. Derivatives are almost 0 in the asymptotes. This may occur during the early stages of training. The final trained network converges to a poor local optimum.



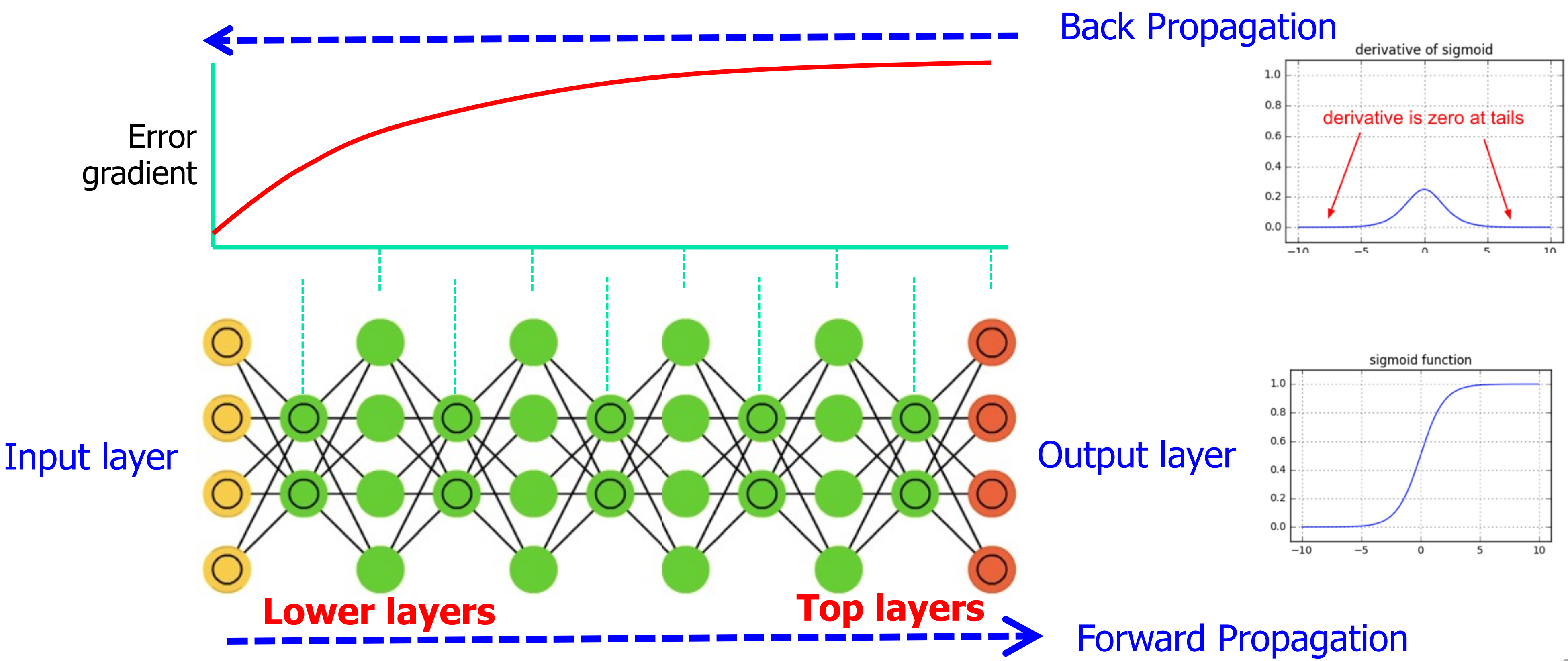
Graphically

$\delta_i^{(p)} = (t_i^{(p)} - y_i^{(p)}) \cdot f'(net_i^{(p)})$. If $net_i^{(p)}$ is high, $\delta_i^{(p)} \to 0$

$\delta_i^{(p)} = f'(net_i^{(p)}) \sum_k \delta_k^{(p)} \cdot w_{ki}$. if $\delta_k^{(p)} \to 0$ and $net_i^{(p)}$ is high, then $\delta_i^{(p)} \approx 0$

D. Manrique. (2021): "From artificial cells to deep learning". Archivo Digital UPM.

- When inputs become large at top layers, the logistic activation function saturates at 0 or 1, being its derivative close to zero.
- When BP starts, it has no gradient to propagate back.
- Still, that little gradient back-propagates, getting diluted even more as approaching lower layers, where there is nothing left to update weights.



http://www.asimovinstitute.org/neural-network-zoo/

Deep ANN tanh results

Class	t
Cheap	100
Averaged	010
Expensive	001

Shallow NN don't work when they become deep

Hyperparameters:

Stop condition: 600 epochs

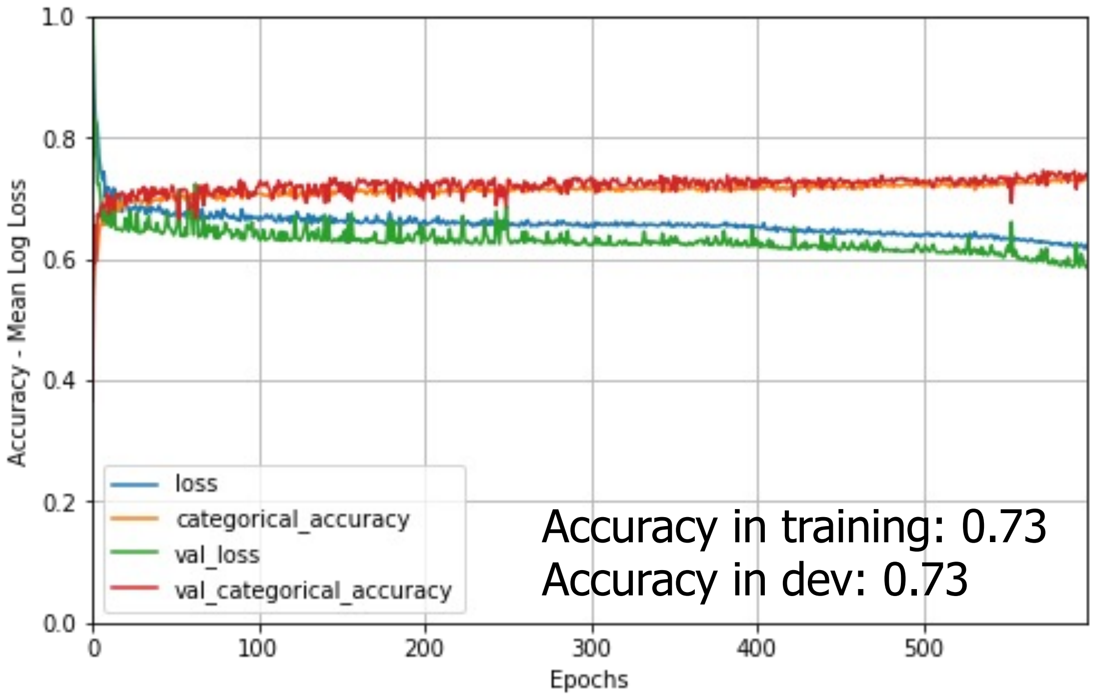
Learning rate α=0.1

Mini-batch size = 512

Hidden layers = 10,000-10,000-10,000

Activation functions: tanh - Softmax

16342 samples for training;
2043 for development.

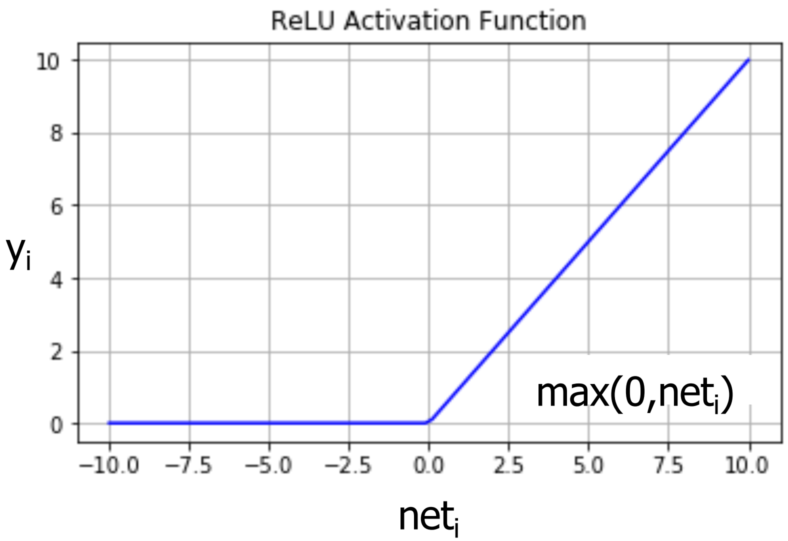


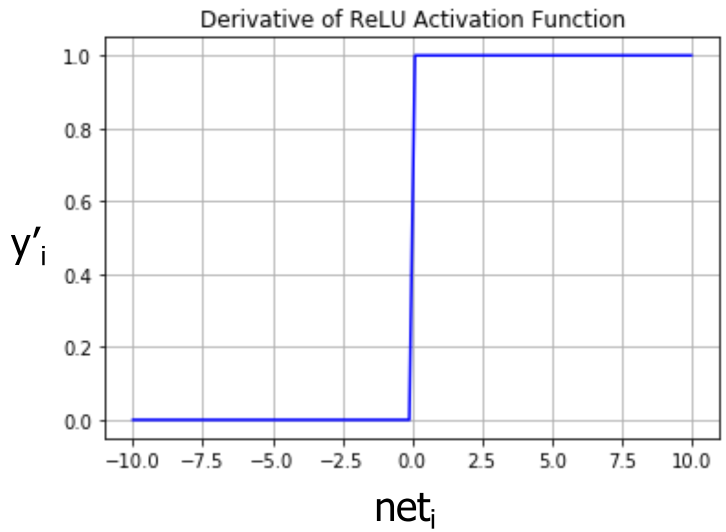
Comparisons: accuracy, time

Approach	Train	Dev	Time
MLP	0.76	0.75	2:40
Deep-tanh	0.73	0.73	80:00

Solution: ReLU

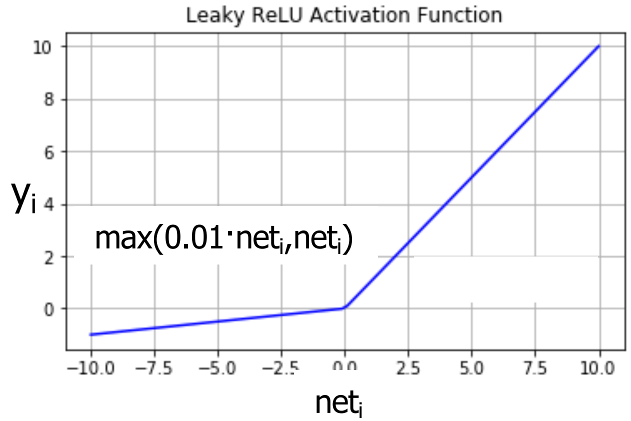
- Activation functions that do not saturate.
- ReLU stands for Rectifier Linear Unit.
- It is the simplest activation function you can think of.
- It is very fast to compute: $\text{ReLU}(net_i) = \max(0, net_i)$





https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/

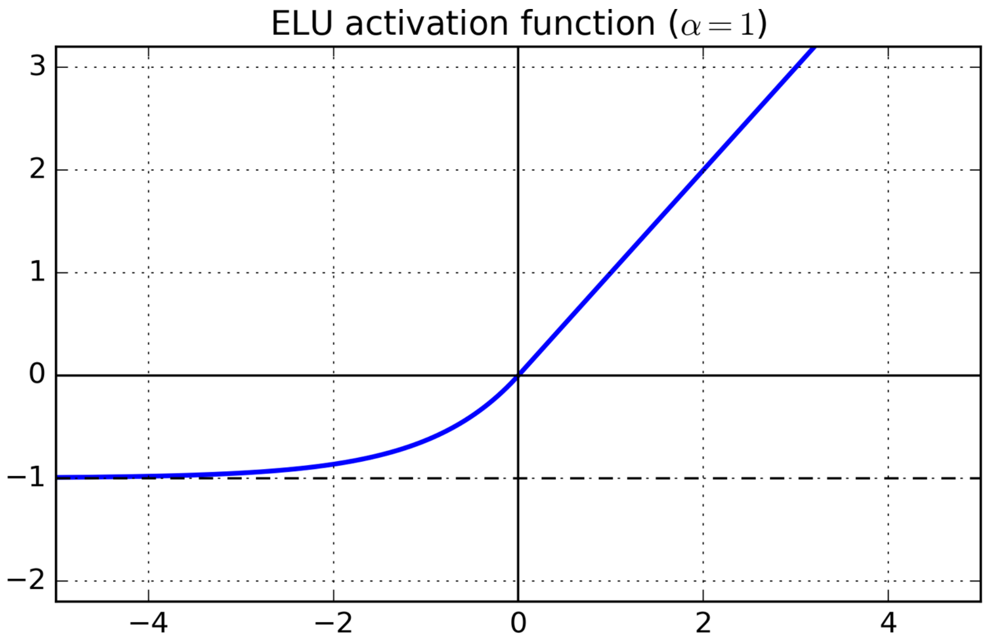
ReLU variants: leaky



- Learning rate α should be small (close to zero) when using ReLU.
- Otherwise, about half of the neurons die:
  - if neti<0 for a neuron, then ReLU will start outputting 0. When this happens, the neuron is unlikely to come back to life since the gradient of the ReLU function is 0 when its input is negative.
- $LeakyReLU_s(net_i) = \max(s \cdot net_i, net_i)$; $s \rightarrow 0$
- s is the slope of function for $net_i<0$, typically $s=0.01$.
- Leaky always outperforms strict ReLU.
- Randomized leaky may reduce overfitting.

ReLU Variants: ELU

- Exponential Linear Unit reduces training time and performs better on the final test set.
- $ELU_\alpha(net_i)=\boldsymbol{\alpha}\cdot(e^{neti}-1)$, if $net_i < 0$; net_i otherwise
- $\boldsymbol{\alpha}$ usually equals 1.



https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/

Deep ANN results with ReLU

Class	t
Cheap	100
Averaged	010
Expensive	001

Hyperparameters:

Stop condition: 600 epochs

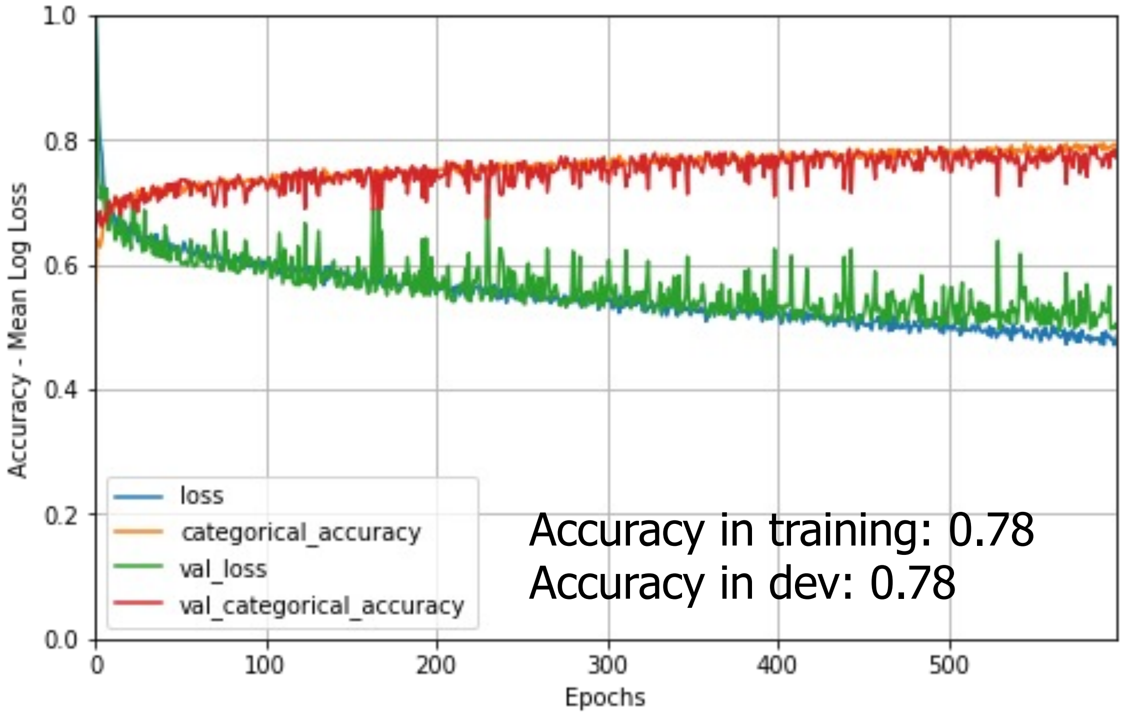
Learning rate α=0.1

Mini-batch size = 512

Hidden layers = 10,000-10,000-10,000

Activation functions: ReLU - Softmax

16342 samples for training;
2043 for development.



Comparisons: accuracy, time

Approach	Train	Dev	Time
MLP	0.76	0.75	2:40
Deep-tanh	0.73	0.73	80:00
ReLU	0.78	0.78	20:00

After tuning hyperparameters

Class	t
Cheap	100
Averaged	010
Expensive	001

Hyperparameters:

Stop condition: 450 epochs

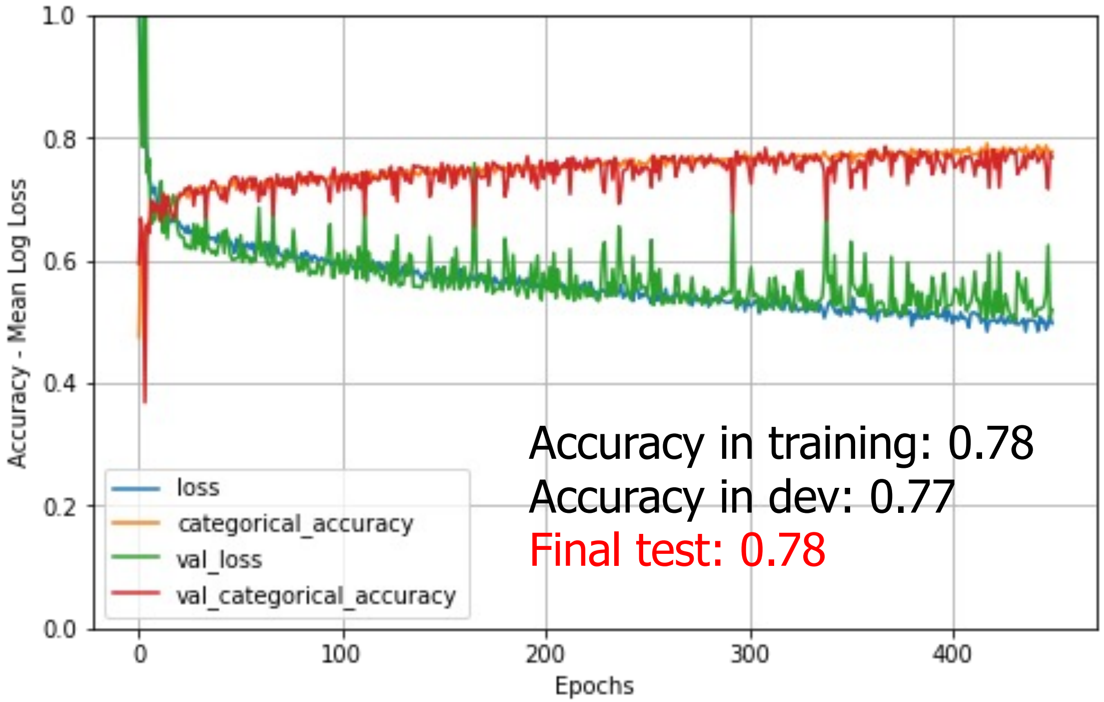
Learning rate α=0.1

Mini-batch size = 512

Hidden layers = 500-250-75-25

Activation functions: ReLU - Softmax

16,342 samples for training;
2,043 for development.
2,043 for final test.



Comparisons: accuracy, time

Approach	Train	Dev	Time
MLP	0.73	0.74	3:30
Deep-tanh	0.73	0.74	80:00
ReLU	0.78	0.78	20:00
Tuned	0.78	0.77	1:20

What if we keep on training?

Class	t
Cheap	100
Averaged	010
Expensive	001

Hyperparameters:

Stop condition: 1,000 epochs

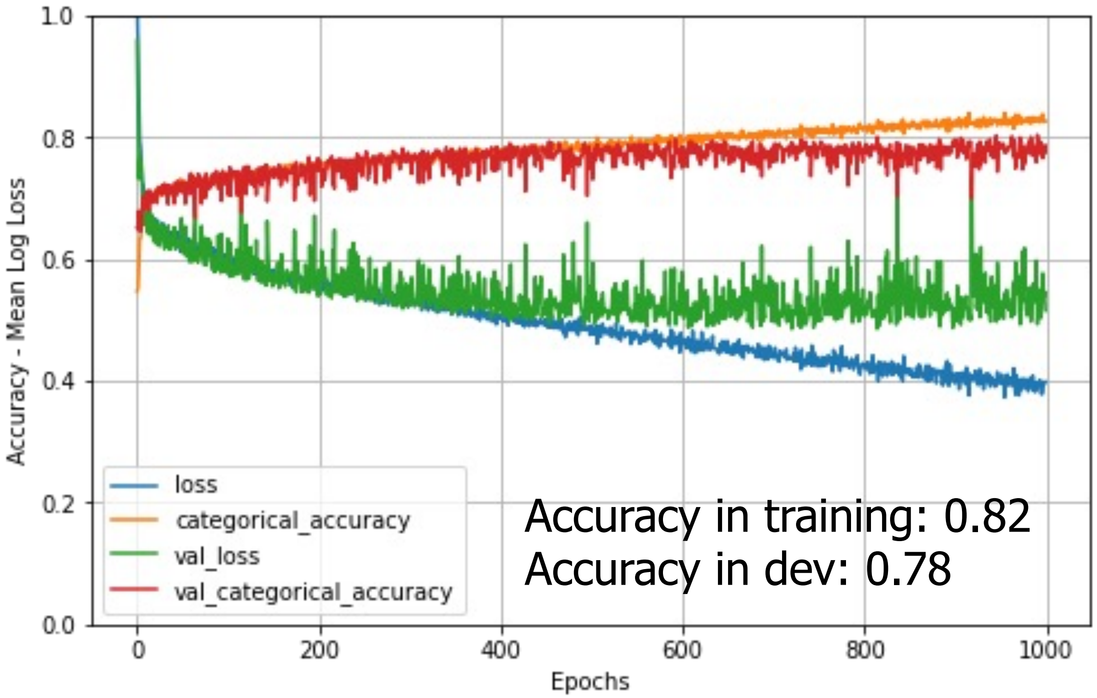
Learning rate α=0.1

Mini-batch size = 512

Hidden layers = 500-250-75-25

Activation functions: ReLU - Softmax

16,342 samples for training;
2,043 for development.



Comparisons: accuracy, time

Approach	Train	Dev	Time
MLP	0.73	0.74	3:30
Deep-tanh	0.73	0.74	80:00
ReLU	0.78	0.78	20:00
Tuned	0.78	0.77	1:20
Overfitted	0.82	0.78	3:20

Lecture slides of the master course “Deep Learning”.
2025 Daniel Manrique

Suggested work citation:

D. Manrique. (2025). *Training Methods for Deep Neural Networks. Deep Neural Networks*. Master course (lecture slides). Department of Artificial Intelligence. Universidad Politécnica de Madrid.