DEEP LEARNING CRASH COURSE

- Single Layer Perceptron
- Multiple Layer Perceptron
- Convolutional Neural Net

M.A. Nielsen. Neural Networks and Deep Learning, 2015
http://neuralnetworksanddeeplearning.com/
ARTIFICIAL INTELLIGENCE

1997: Deep Blue beats chess World Champion

2016: AlphaGo beats go world champion
LINEAR CLASSIFICATION

\[ f(x) = \begin{cases} 
1 \text{ if } w \cdot x + b \geq 0, \\
0 \text{ otherwise.}
\end{cases} \]
SINGLE LAYER PERCEPTRON

\[ f(x) = \sigma(w \cdot x + b) \]

\( \sigma \): Step function and Sigmoid function graphs.
The network can be trained to produce a desired output given a specific input.

In practice, this means learning the $b$ and $w$ parameters by minimizing a loss function on a training set.

Often done on GPUs, which are much faster.

$$f(x) = \sigma(W \cdot x + B)$$

For each node $j$ in layer $l$,

$$a_j^l = \sigma \left( b_j^l + \sum_k w_{j,k}^l a_k^{l-1} \right),$$

where $a$ is the activation of the node.
In the binary case,

\[
L(w, b) = -\frac{1}{N} \sum_{1}^{N} [y_n \log(\hat{y}_n) + (1 - y_n) \log(1 - \hat{y}_n)]
\]

where \( \hat{y}_n = f_{w,b}(x_n) \).
In the multiclass case, the probability that input vector $\mathbf{x}$ belongs to class $i$ can be written as

$$P(Y = i | \mathbf{x}, \mathbf{w}, \mathbf{b}) = \frac{f_i(\mathbf{x})}{\sum_j f_j(\mathbf{x})}$$

The class assigned to vector $\mathbf{x}$ is taken to be

$$\hat{y} = \arg \max_i P(Y = i | \mathbf{x}, \mathbf{w}, \mathbf{b})$$

Given a set of $N$ training samples $(\mathbf{x}_n, y_n)_{1 \leq n \leq N}$, the loss function can be written as

$$L(\mathbf{w}, \mathbf{b}) = \sum_n \log(P(Y = y_n | \mathbf{x}_n, \mathbf{w}, \mathbf{b}))$$

$\rightarrow$ $L$ is a differentiable function of $\mathbf{w}$ and $\mathbf{b}$ and can be optimised using back propagation, that is, gradient descent.
The network takes as input 28x28 images represented as 784D vectors.

The output is a 10D vector giving the probability of the image representing any of the 10 digits.

There are 50’000 training pairs of images and the corresponding label, 10’000 validation pairs, and 5’000 testing pairs.
• The descriptive power of the net increases with the number of layers.
• Since every neuron is connected to every other in adjacent layers, the number of parameters to be learned increases quickly.
• Does not take into account the specific topology of images.
CONVOLUTIONAL LAYER

\[ \sigma \left( b + \sum_{x=0}^{n_x} \sum_{y=0}^{n_y} w_{i,j} a_{i+x,j+y} \right) \]
FEATURE MAPS

28 × 28 input neurons

first hidden layer: 3 × 24 × 24 neurons
• Reduce the number of inputs by replacing all activations in a neighbourhood by a single one.
• Can be thought as asking if a particular feature is present in that neighbourhood while ignoring the exact location.
ADDING THE POOLING LAYERS

The output size is reduced by the pooling layers.
- Each neuron in the final fully connected layer is connected to all neurons in the preceding one.
- Deep architecture with many parameters to learn but still far fewer than an equivalent multilayer perceptron.
HAND POSE ESTIMATION

Input: Depth image.  
Output: 3D pose vector.

Oberweger et al., ICCV’15
Network parameters are found by minimizing and objective function of the form

\[
\min_{W_l, B_l} \sum_i \| F(x_i, W_1, \ldots, W_L, b_1, \ldots, b_L) - y_i \|^2
\]

using

- stochastic gradient descent on mini-batches,
- dropout,
- hard example mining,
- ............
FEATURE MAPS LEARNED FOR IMAGE CLASSIFICATION

Some of the convolutional masks seem very similar to oriented Gaussian or Gabor filters!

Much ongoing work to better understand this.
DeepFace
Taigman et al. 2014

Deep Edge Detection
Shen et al. 2015
DEEPER AND DEEPER

He et al., CVPR’16
VISUAL CORTEX
AlphaGo

- Uses Deep Nets to find the most promising locations to focus on.
- Performs Tree based search when possible.
- Relies on reinforcement learning and other ML techniques to train.
ADVERSARIAL IMAGES

Szegedy et al. 2013
IN SHORT

• Deep Belief Networks in general and Convolutional Neural Nets in particular outperform conventional Computer Vision algorithms on many benchmarks.
• It is not fully understood why and unexpected failure cases have been demonstrated.
• They require a lot of manual tuning to perform well and performance is hard to predict.

—> Many questions are still open and there is much work left to do.