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/
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

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

Step function

Sigmoid function
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 $l$,

$$a^l_j = \sigma \left( b^l_j + \sum_k w^l_{j,k} a^{l-1}_k \right),$$

where $a$ is the activation of the node.
BINARY LOSS FUNCTION

In the binary case,

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

where \( \hat{y}_n = f_{w,b}(x_n) \).
MULTI-CLASS LOSS FUNCTION

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.
PERCEPTRON WITH RELU

\[
f(x) = \sigma(W \cdot x + B)
\]

For each node \( j \) in layer \( l \),
\[
a^l_j = \max \left( b^l_j + \sum_k w_{j,k}^l a^{l-1}_k, 0 \right),
\]
where \( a \) is the activation of the node.

- Each node defines a hyperplane.
- The resulting function is piecewise linear affine and continuous.
ONE SINGLE HYPERPLANE

\[ y = \max(\mathbf{w}^\top \mathbf{x} + b, 0) \]

\[ y = 0 \]

\[ \mathbf{w}^\top \mathbf{x} + b = 0 \]

\[ y = \mathbf{w}^\top \mathbf{x} + b \]
\[
\begin{aligned}
\{ & h = \max(W_1 x + b_1, 0) \\
& y = w^\top h \\
\text{with dim}(h) = 2
\end{aligned}
\]
THREE HYPERPLANES

\[
\begin{align*}
\{ & \quad h = \max(Wx + b, 0) \\
& \quad y = w^\top h \\
& \text{with } \dim(h) = 3
\end{align*}
\]
PERCEPTRON WITH TANH

- Each node defines a hyperplane.
- The resulting function is piecewise smooth and continuous.

\[ f(x) = \sigma(W \cdot x + B) \]

For each node \( j \) in layer \( l \),

\[ a_j^l = \tanh \left( b_j^l + \sum_k w_{j,k}^l a_{k}^{l-1} \right), \]

where \( a \) is the activation of the node.
INTERPOLATING A SURFACE

\[ z = 100 \times (y - x^2)^2 + (1 - x)^2 \]

3-node hidden layer
INTERPOLATING A SURFACE

\[ z = 100 \times (y - x^2)^2 + (1 - x)^2 \]

4-node hidden layer

loss: 1.089789e+00
Adding more nodes:

\[
z = 100 \times (y - x^2)^2 + (1 - x)^2
\]

- 2 nodes -> loss 3.02e-01
- 3 nodes -> loss 2.08e-02
- 4 nodes -> loss 8.27e-03
Adding more nodes

\( z = \sin(x) \sin(y) \)

2 nodes -> loss 2.61e-01

3 nodes -> loss 2.51e-04

4 nodes -> loss 3.07e-07
MORE COMPLEX SURFACE

\[ I = f(x, y) \]
$I = f(x, y)$
A feedforward network with a linear output layer and at least one hidden layer with any 'squashing' activation function (e.g. logistic sigmoid) can approximate any Borel measurable function (from one finite-dimensional space to another) with any desired nonzero error.

Any continuous function on a closed and bounded set of $\mathbb{R}^n$ is Borel-measurable.

$\rightarrow$ In theory, any reasonable function can be approximated by a two-layer network as long as it is continuous:

[Hornik et al, 1989; Cybenko, 1989]
IN PRACTICE

• It may take an exponentially large number of parameters for a good approximation.
• The optimisation problem becomes increasingly difficult.

--> The one hidden layer perceptron may not converge to the best solution!

Telgarsky, JMLR’16
The descriptive power of the net increases with the number of layers.

In the case of a 1D signal, it is roughly proportional to $\prod_{n} W_n$ where $W_n$ represents the width of a layer.

Telgarsky, JMLR’16
\[
\begin{aligned}
\begin{cases}
    h = \max(Wx + b, 0) \\
    h' = \max(W'h + b', 0) \\
    y = w''^\top h'
\end{cases}
\end{aligned}
\]

with \( \text{dim}(h) = 2 \),

\[
\begin{bmatrix}
0 \\
0
\end{bmatrix}
\]

\[
h = \begin{bmatrix}
0 \\
0
\end{bmatrix}
\]

\[
h' = \max(b', 0) \\
y = w'' \max(b', 0) = \text{cst}
\]
**ADDING A SECOND LAYER**

\[ I = f(x, y) \]

1 Layer: 125 nodes -> loss 2.40e-01
2 Layers: 20 nodes -> loss 8.31e-02

501 weights in both cases
$I = f(x, y)$

2 Layers: 20 nodes -> loss 8.31e-02  
3 Layers: 14 nodes -> loss 7.55e-02

501 weights  
477 weights
ADDING A THIRD LAYER

\[ I = f(x, y) \]  

3 Layers: 15 nodes -> loss 5.93e-02  
3 Layers: 19 nodes -> loss 4.38e-02

541 weights  
837 weights
MULTILAYER PERCEPTRONS

• Adding layers often yields better convergence properties.
• In current practice, deeper is usually better.
SIMPLER WAY TO INTERPOLATE

Original 51x51 image: 2601 gray level values.

Scaled 24x24 image: 576 gray level values.

MLP 10/20/10 Interpolation: 471 weights.

Simpler but not necessarily better!
Further improvements in the convergence properties have been obtained by adding a bypass, which allows the final layers to only compute residuals.
IMPROVING THE NETWORK

Original 51x51 image: 2601 gray level values.

MLP 10/20/10 Interpolation: 471 weights, loss 6.43e-02.

MLP 10/20/10/10 Interpolation: 581 weights, loss 5.30e-2.
IMPROVING THE NETWORK

- Relatively small improvement in this case.
- The problem is probably too small.

$\rightarrow$ Networks can behave very differently for small and large problems!
TANH vs ReLU

Original 51x51 image: 2601 gray level values.

MLP 10/20/10 Interpolation:
- **Tanh**, loss 6.43e-02.

MLP 10/20/10 Interpolation:
- **ReLU**, loss 3.07e-1.

TANH vs ReLU
- Tanh, 1 layers
- Tanh, 2 layers
- Tanh, 3 layers
- ReLU, 3 layers
- Tanh, 4 layers

In the graph, the x-axis represents the number of weights, and the y-axis represents the loss. The different lines correspond to the different configurations of Tanh and ReLU with varying numbers of layers.
TANH vs ReLU

- Tanh works better than ReLU in this case.
- ReLU is widely credited with eliminating the vanishing gradient problem in large networks.

--> There is no substitute for experimentation!
To summarize roughly the evolution of convnets for image classification:

- standard ones are extensions of LeNet5,
- everybody loves ReLU,
- ......
The function learned by a DNN using either the ReLU or Tanh operators is:
• piecewise affine or smooth;
• continuous because it is a composition of continuous functions.

Each region created by a layer is split into smaller regions:
• The equations for each one are correlated in a complex way.
• This may explain why deeper networks generalize better than larger networks for a given number of parameters.
STRENGTHS AND LIMITATIONS

- Powerful regressors but require many parameters, and therefore large training databases.
- Excellent at interpolation but less good at extrapolation. The training data must cover all cases of interest.
In a typical image, the values of neighboring pixels tend to be more highly correlated than those of distant ones. An image filter should be translation invariant.

These two properties can be exploited to drastically reduce the number of weights required by CNNs using so-called convolutional layers.
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
**POOLING LAYER**

- Reduce the number of inputs by replacing all activations in a neighborhood by a single one.
- Can be thought as asking if a particular feature is present in that neighborhood while ignoring the exact location.
ADDING THE POOLING LAYERS

The output size is reduced by the pooling layers.
ADDING A FULLY CONNECTED LAYER

- 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.
MNIST

- 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.
LeNet (1989-1999)
IS MAX POOL REQUIRED?

Accuracy | Train  | Test  
---|---|---
Conv 5x5, stride 1 Max pool 2x3 | 99.58 | 98.77 
Conv 5x5, stride 2 | 99.42 | 98.31 
Conv 5x5, stride 1 Conv 3x3, stride 2 | 99.38 | 98.57 

Springenberg et al., ICLR’15
AlexNet (2012)

Task: Image classification
Training images: Large Scale Visual Recognition Challenge 2010
Training time: 2 weeks on 2 GPUs

Major Breakthrough: Training large networks has now been shown to be practical!!
AlexNet RESULTS
Some of the convolutional masks seem very similar to oriented Gaussian or Gabor filters!

Much ongoing work to better understand this.
HIGHER ORDER DERIVATIVES
FILTER BANKS

Hand-Designed

Learned
“It was demonstrated that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet challenge dataset can be achieved using a conventional ConvNet architecture.”

Simonyan & Zisserman, ICLR’15
HAND POSE ESTIMATION (2015)

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

Oberweger et al., ICCV’15
Network parameters are found by minimizing an 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,
- ............
ROC HUNTING

DeepFace
Taigman et al. 2014

Deep Edge Detection
Shen et al. 2015
DEEPER AND DEEPER

He et al., CVPR’16
MONOCULAR POSE ESTIMATION

Tekin et al., ICCV’17
# IMAGE CLASSIFICATION TAXONOMY

- **LSTM** (Hochreiter and Schmidhuber, 1997)
- **LeNet5** (LeCun et al., 1989)
- **Highway Net** (Srivastava et al., 2015)
- **AlexNet** (Krizhevsky et al., 2012)
- **Overfeat** (Sermanet et al., 2013)
- **VGG** (Simonyan and Zisserman, 2014)
- **GoogLeNet** (Szegedy et al., 2015)
- **ResNet** (He et al., 2015)
- **Net in Net** (Lin et al., 2013)
- **MLPConv**
- **Bigger + ReLU + dropout**
- **Inception modules**
- **Batch Normalization**
- **BN-Inception** (Ioffe and Szegedy, 2015)
- **ResNeXt** (Xie et al., 2016)
- **Aggregated channels**
- **Wide ResNet** (Zagoruyko and Komodakis, 2016)
- **DenseNet** (Huang et al., 2016)
- **Wide ResNeXt** (He et al., 2016)
- **Inception-ResNet** (Szegedy et al., 2016)
ANOTHER POINT OF VIEW

To summarize roughly the evolution of convnets for image classification:

• standard ones are extensions of LeNet5,
• everybody loves ReLU,
• state-of-the-art networks have 100s of channels and 10s of layers,
• they can (should?) be fully convolutional,
• pass-through connections allow deeper “residual” nets,
• bottleneck local structures reduce the number of parameters,
• aggregated pathways reduce the number of parameters.
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
XKCD’S VIEW ON THE MATTER

THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.

https://xkcd.com/
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.