Deep Learning Crash Course

- Single Layer Perceptrons
- Multiple Layer Perceptrons
- Convolutional Neural Nets
Binary vs Multi-Class Classification

Logistic Regression

LeNet
Two classes shown as different colors:

- The label \( y \in \{-1, 1\} \) or \( y \in \{0, 1\} \).
- The samples with label 1 are called positive samples.
- The samples with label -1 or 0 are called negative samples.
Signed Distance

\[ \tilde{x} = [1, x_1, x_2] \]

Notation:
- \( x = [x_1, x_2] \)
- \( \tilde{x} = [1, x_1, x_2] \)

Signed distance:
\[ h = w_0 + w_1 x_1 + w_2 x_2 \]
\[ = \tilde{w} \cdot \tilde{x} \]

- \( h = 0 \): Point is on the line.
- \( h > 0 \): Point in the normal’s direction.
- \( h < 0 \): Point in the other direction.

\[ \tilde{w} = [w_0, w_1, w_2] \text{ with } w_1^2 + w_2^2 = 1 \]
Signed Distance in 3D

\[ x = [x, y, z] \]

\[ \tilde{x} = [1, x, y, z] \]

\[ x \in \mathbb{R}^3, \ 0 = ax + by + cz + d \]

\[ \tilde{w} = [w_0, w_1, w_2, w_3] \]

\[ \tilde{w} \cdot \tilde{x} = 0 \]

Signed distance \( h = \tilde{w} \cdot \tilde{x} \) if \( w_1^2 + w_2^2 + w_3^2 = 1 \).
Signed Distance in N Dimensions

\[ \mathbf{w} = [w_1, \ldots, w_n] \]

\[ \tilde{\mathbf{w}} = [w_0, w_1, \ldots, w_n] \text{ with } \sum_{i=1}^{N} w_i^2 = 1 \]

Notation:
\[ \mathbf{x} = [x_1, \ldots, x_n] \]
\[ \tilde{\mathbf{x}} = [1, x_1, \ldots, x_n] \]

Hyperplane:
\[ \mathbf{x} \in \mathbb{R}^n, \quad 0 = \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} \]
\[ = w_0 + w_1 x_1 + \ldots + w_n x_n \]

Signed distance:
\[ h = \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} \]

h=0: Point is on the decision boundary.
h>0: Point on one side.
h<0: Point on the other side.
Binary Classification in N Dimensions

Hyperplane: \( \mathbf{x} \in R^N, \hat{\mathbf{w}} \cdot \hat{\mathbf{x}} = 0, \) with \( \hat{\mathbf{x}} = [1 | \mathbf{x}] \).

Signed distance: \( \hat{\mathbf{w}} \cdot \hat{\mathbf{x}}, \) with \( \hat{\mathbf{w}} = [w_0 | \mathbf{w}] \) and \( ||\mathbf{w}|| = 1 \).

Problem statement: Find \( \hat{\mathbf{w}} \) such that
- for all or most positive samples \( \hat{\mathbf{w}} \cdot \hat{\mathbf{x}} > 0, \)
- for all or most negative samples \( \hat{\mathbf{w}} \cdot \hat{\mathbf{x}} < 0. \)
Logistic Regression

\[ y(x; \tilde{w}) = \sigma(\tilde{w} \cdot \tilde{x}) \]

\[ = \frac{1}{1 + \exp(-\tilde{w} \cdot \tilde{x})} \]

- When the noise is Gaussian, this is the maximum likelihood solution.
- \( y(x; \tilde{w}) \) can be interpreted at the probability that \( x \) belongs to positive class.

Given a **training** set \( \{(x_n, t_n)_{1 \leq n \leq N}\} \) minimize

\[ - \sum_n (t_n \ln y(x_n) + (1 - t_n) \ln(1 - y(x_n))) \]

with respect to \( \tilde{w} \).

- When the noise is Gaussian, this is the maximum likelihood solution.
- \( y(x; \tilde{w}) \) can be interpreted at the probability that \( x \) belongs to positive class.
Multi-Class Logistic Regression

- K linear classifiers of the form \( y^k(x) = \sigma(w_k^T x) \).
- Assign \( x \) to class \( k \) if \( y^k(x) > y^l(x) \forall l \neq k \).

Because the sigmoid function is monotonic, the formulation is almost unchanged.

- Only the objective function being minimized need to be reformulated.

\[
k = \arg \max_j y_k(x)
\]

\[
\begin{bmatrix}
y_1 \\
\vdots \\
y_K
\end{bmatrix} =
\begin{bmatrix}
\tilde{w}_1^T \\
\vdots \\
\tilde{w}_K^T
\end{bmatrix}
\tilde{x}
\]

\[
k = \arg \max_j y_j
\]

Bishop, Chapter 4.3.4
Non Separable Distribution

Positive: $100(x_2 - x_1^2)^2 + (1 - x_1)^2 < 0.5$

Negative: Otherwise

Logistic regression can handle a few outliers but not a complex non-linear boundary.

How can we learn a function $y$ such that $y(x; \tilde{w})$ is close to 1 for positive samples and close to 0 or -1 for negative ones?

$\rightarrow$ Use LOTS of hyperplanes.
Reformulating Logistic Regression

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

\[ x = [x_1, x_2, \ldots, x_n]^T \]

\[ w = [w_1, w_2, \ldots, w_n]^T \]
Repeating the Process

\[ h_1 = \sigma(w_1 \cdot x + b_1) \]
\[ w_1 = [w_{11}, w_{12}, w_{13}, w_{14}]^T \]

\[ h_2 = \sigma(w_2 \cdot x + b_2) \]
\[ w_2 = [w_{21}, w_{22}, w_{23}, w_{24}]^T \]

\[ \vdots \]
\[ \vdots \]

\[ h_H = \sigma(w_H \cdot x + b_H) \]
\[ w_H = [w_{H1}, w_{H2}, w_{H3}, w_{H4}]^T \]
Repeating the Process

\[ h = \sigma(Wx + b), \]

with \( W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_H \end{bmatrix} \)

and \( b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_H \end{bmatrix} \).
Multi-Layer Perceptron

\[ h = \sigma_1(W_1x + b_1) \]
\[ y = \sigma_2(W_2h + b_2) \]

- The process can be repeated several times to create a vector \( h \).
Multi-Layer Perceptron

- The process can be repeated several times to create a vector $h$.
- It can then be done again to produce an output $y$.

$\rightarrow$ This output is a **differentiable** function of the weights.

\[
\begin{align*}
    h &= \sigma_1(W_1 x + b_1) \\
    y &= \sigma_2(W_2 h + b_2)
\end{align*}
\]
ReLU

\[ h = \sigma_1(W_1x + b_1) \]
\[ y = \sigma_2(W_2h + b_2) \]

\[ \sigma(x) = \max(0, x) \]

- Each node defines a hyperplane.
- The resulting function is piecewise linear affine and continuous.
Sigmoid and Tanh

\[ h = \sigma(W_1 x + b_1) \]
\[ y = \sigma(W_2 h + b_2) \]

- Each node defines a hyperplane.
- The resulting function is continuously differentiable.
Binary Case

\[ h = \sigma(W_1 x_n + b_1) \]
\[ y = \sigma(w_2 h + b_2) \]

In this case \( w_2 \) is vector.
ReLu Behavior

\[
W_1 = \begin{bmatrix}
  \frac{1}{2} & -\frac{1}{2} \\
  -\frac{1}{2} & \frac{1}{2}
\end{bmatrix}
\]

\[
h = \text{ReLu}(W_1x)
\]

\[
y = w_2^T h + b_2
\]
Binary Case

• Let the training set be \( \{(x_n, t_n)_{1 \leq n \leq N}\} \) where \( t_n \in \{0,1\} \) is the class label and let us consider a neural net with a 1D output.

• We write

\[
y_n = \sigma(w_2(\sigma(W_1 x_n + b_1)) + b_2) \in [0,1] .
\]

• We want to minimize the binary cross entropy

\[
E(W_1, w_2, b_1, b_2) = \frac{1}{N} \sum_{n=1}^{N} E_n(W_1, w_2, b_1, b_2) ,
\]

\[
E_n(W_1, w_2, b_1, b_2) = - (t_n \ln(y_n) + (1 - t_n)\ln(1 - y_n)) ,
\]

with respect to the coefficients of \( W_1, w_2, b_1, \) and \( b_2. \)

• E can be minimized using a gradient-based technique.
Binary Case

Given a training set \(\{x_n, t_n\}_{1 \leq n \leq N}\) where \(t_n \in \{0, 1\}\), minimize

\[
E(W, b) = -\frac{1}{N} \sum_{n=1}^{N} E_n(W, b)
\]

\[
E_n(W, b) = t_n \log(y_n) + (1 - t_n) \log(1 - y_n),
\]

\[
y_n = f(x_n) = \sigma(W_2(\sigma(W_1x_n + b_1)) + b_2),
\]

Since \(E\) is a differentiable function of \(W\) and \(b\), this can be done using a gradient-based technique.
One Single Hyperplane

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

\[ y = 0 \]

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

\[ y = \mathbf{w}^\top \mathbf{x} + b \]
Two Hyperplanes

\[ h = \max(\mathbf{Wx} + \mathbf{b}, 0) \] with \( \mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \end{bmatrix} \) and \( \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \)

\[ y = \mathbf{w'}^T \mathbf{h} + b' \]
Three Hyperplanes

\[ \begin{align*}
\{ & \quad \mathbf{h} = \max(\mathbf{Wx} + \mathbf{b}, 0) \\
& \quad y = \mathbf{w}^T \mathbf{h} \\
\text{with dim}(\mathbf{h}) = 3
\end{align*} \]
Multi-Class Case

\[ h = \sigma(W_1x_n + b_1) \]
\[ y = \sigma(W_2h + b_2) \]

In this case, \( W_2 \) is a matrix.
Multi-Class Case

Let the training set be \( \{(x_n, [t_n^1, \ldots, t_n^K])_{1 \leq n \leq N}\} \) where \( t_n^k \in \{0,1\} \) is the probability that sample \( x_n \) belongs to class \( k \).

- We write

\[
y_n = \sigma(W_2(\sigma(W_1x_n + b_1)) + b_2) \in R^K
\]

\[
p_n^k = \frac{\exp(y_n[k])}{\sum_j \exp(y_n[j])}
\]

- We minimize the cross entropy

\[
E(W_1, W_2, b_1, b_2) = \frac{1}{N} \sum_{n=1}^{N} E_n(W_1, W_2, b_1, b_2),
\]

\[
E_n(W_1, W_2, b_1, b_2) = - \sum t_n^k \ln(p_n^k),
\]

with respect to the coefficients of \( W_1, W_2, b_1, \) and \( b_2 \).
Non-Linear Binary Classification

Positive: \(100(x_2 - x_1^2)^2 + (1 - x_1)^2 < 0.5\)
Negative: Otherwise

\(y(x; \tilde{w})\) is now a non-linear function implemented by the network.

**Problem statement:** Find \(\tilde{w}\) such that
- for all or most positive samples \(y(\tilde{x}; \tilde{w}) > 0.0\),
- for all or most negative samples \(y(\tilde{x}; \tilde{w}) < 0.0\).
Non-Linear Regression

Problem statement: Given \( \{x_1, z_1\}, \ldots, \{x_n, z_n\} \), minimize

\[
\sum_i (z_i - f(x_i, \tilde{w}))^2
\]

w.r.t. \( \tilde{w} \).

\[
z = f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2
\]
Classification / Regression

Classification can be understood as finding \( \tilde{\mathbf{w}} \) such that

\[
y(\mathbf{x}; \tilde{\mathbf{w}}) \approx f(\mathbf{x})
\]

Positive: \( f(\mathbf{x}) < 0.5 \)
Negative: Otherwise

\( y(\mathbf{x}; \tilde{\mathbf{w}}) \) is now a non-linear function implemented by the network.
From Classification to Regression

Minimize $\sum_i [t_i \log(\text{sigm}(f(x_i, y_i))) + (1 - t_i) \log(1 - \text{sigm}(f(x_i, y_i)))$ with respect to $W_1, w_2, b_x, b_y, b_z$.

Minimize $\sum_i (z_i - f(x_i, y_i))^2$, with respect to $W_1, w_2, b_x, b_y, b_z$. 
Approximating a Surface

\[ z = f(x, y) \]
\[ \quad = w_2 \sigma(W_1 \begin{bmatrix} x \\ y \end{bmatrix} + b_1) + b_2 \]

Given \( \{x_1, z_1\}, \ldots, \{x_n, z_n\} \), minimize
\[ \sum_i (z_i - f(x_i))^2 \]
with respect to \( W_1, w_2, b_1, b_2 \).
Interpolating a Surface

\[ z = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \]

3-node hidden layer

loss: 2.192508e+01
Interpolating a Surface

\[ z = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \]

4-node hidden layer
Adding more Nodes

\[ z = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \]
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

\[ I = f(x, y) \]
More Complex Surface

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

- 50 nodes \(\rightarrow\) loss 3.65e-01
- 100 nodes \(\rightarrow\) loss 2.50e-01
- 125 nodes \(\rightarrow\) loss 2.40e-01
- 300 nodes \(\rightarrow\) loss 1.92e-01
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.

\[ \rightarrow \text{In theory, any reasonable function can be approximated by a one-hidden layer network as long as it is continuous.} \]

[Hornik et al, 1989; Cybenko, 1989]
In Practice

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

→ The one hidden layer perceptron may not converge to the best solution!
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.
MNIST Results

- $n_{In} = 784$
- $n_{Out} = 10$
- $20 < \text{hidden layer size} < 120$

- MLPs have many parameters.
- This has long been a major problem.
  —> Was eventually solved by using GPUs.

- SVM: 98.6
- Knn: 96.8

- Around 2005, SVMs were often felt to be superior to neural nets.
- This is no longer the case ....
In the case of a 1D signal, it is roughly proportional to $w_n$ where $w_n$ is the width of layer $n$.

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 $w_n$ is the width of layer $n$. 

$h_1 = \sigma_1(W_1x_1 + b_1)$

$h_2 = \sigma_2(W_2h_2 + b_2)$

$\ldots$

$y = \sigma_n(W_nh_n + b_n)$
One Layer: Two Hyperplanes

$h = \max(Wx + b, 0)$ with $W = \begin{bmatrix} w_1^T \\ w_2^T \end{bmatrix}$ and $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

$y = w'^T h + b'$

$h = \begin{bmatrix} 0 \\ w_2^T x + b_2 \end{bmatrix}$

$h = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$h = \begin{bmatrix} w_1^T x + b_1 \\ 0 \end{bmatrix}$

$h = \begin{bmatrix} w_1^T x + b_1 \\ w_2^T x + b_2 \end{bmatrix}$
Two Layers: Two Hyperplanes

\[ h = \max(Wx + b, 0) \]

\[ h' = \max(W'h + b', 0) \]

\[ y = w''^T h' + b'' \]
Graphical Interpretation

Hyperplanes at every level of the network split the space.
Graphical Interpretation

Hyperplanes at every level of the network split the space.
Graphical Interpretation

The splits are combined by the hierarchical nature of the network.
Graphical Interpretation

The splits are combined due to the sequential nature of the network.
The function learned by an MLP using the ReLU, Sigmoid, 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:

- Their boundaries are correlated in a complex way.
- Their descriptive power is larger than shallow networks for the same number of parameters.
Second Layer for Approximation

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

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

501 weights in both cases
Adding a Third Layer

$$I = f(x, y)$$

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
Multi Layer Perceptrons

- Adding layers tends to deliver better convergence properties.
- In current practice, deeper is usually 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.

Linear 2 -> 10
Linear 10 -> 20
Linear 20 -> 10
Linear 10 -> 10
Linear 10 -> 10
Linear 10 -> 1
A MxN image can be represented as an MN vector, in which case neighborhood relationships are lost.

By contrast, treating it as a 2D array preserves neighborhood relationships.
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.
- 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.
Convolutional Layer

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

28 × 28 input neurons

first hidden layer: 3 × 24 × 24 neurons

Filters:
Filters

Derivatives

Learned filters
Pooling Layer

- Reduces 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.
LeNet (1989-1999)
Lenet Results

- Ln5: 99.05
- SVM: 98.6
- Knn: 96.8
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!!

Krizhevsky, NIPS’12
AlexNet Results

- At the 2012 ImageNet Large Scale Visual Recognition Challenge, AlexNet achieved a top-5 error of 15.3%, more than 10.8% lower than the runner up.
- Since 2015, networks outperform humans on this task.
Feature Maps

• Some of the convolutional masks are very similar to oriented Gaussian or Gabor filters.
• The trained neural nets compute oriented derivatives, which the brain is also believed to do.
Reminder: Discrete 2D Convolution

Convolution mask $m$, also known as a *kernel*.

$$m_{11} \ldots m_{1w}$$
$$\vdots \quad \vdots \quad \vdots$$
$$m_{w1} \ldots m_{ww}$$

$$m**f(x, y) = \sum_{i=0}^{w} \sum_{j=0}^{w} m(i, j)f(x - i, y - j)$$
Reminder: 3X3 Masks

x derivative

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1 \\
\end{bmatrix}
\text{and}
\begin{bmatrix}
0 & 0 & 0 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

Prewitt operator

y derivative

\[
\begin{bmatrix}
-1 & -1 & -1 \\
0 & 0 & 0 \\
1 & 1 & 1 \\
\end{bmatrix}
\text{and}
\begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{bmatrix}
\]

Sobel operator
Filter Banks

Derivatives of order 0, 1, and 2.

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.”
Deeper and Deeper

He et al., CVPR’16

\[ x + l_2(\sigma(l_1(x))) \]
Max pooling can be replaced by Gaussian convolutions with stride > 1.
ResNet to U-Net

ResNet block

\[ x + l_2(\sigma(l_1(x))) \]

\[ \text{Conv Layer} \]

\[ \text{Conv Layer} \]

\[ \sigma \]

\[ x \]

U-Net

Downsampling

Upsampling

Skip connection

Add skip connection to produce an output of the same size as the input.

Ronneberger, MICCAI'15
Training a U-Net

Train Encoder-decoder U-Net architecture using binary cross-entropy

Minimize

\[ L_{bce}(x, y; w) = -\frac{1}{i} \sum_{i=1}^{P} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \]

where

- \( \hat{y} = f_w(x) \),
- \( x \) in an input image,
- \( y \) the corresponding ground truth.

Mosinska et al, CVPR’18.
• Good start but not the end of the story.
• We will discuss this again during the delineation lecture.
• Break up the images into square patches.
• Transform each path into a feature vector.
• Feed to a transformer architecture.
Given $X = [x_1, \ldots, x_I]$:

- $a[x_i, x_j]$ is the attention that $x_i$ gives to $x_j$. It measures the influence of one on the other.
- It can be computed for all $i$ and $j$ using far fewer weights that in a fully connected layer.

$\Rightarrow$ Provides context.
Transformer Layer

\[
X \leftarrow X + Sa(X)
\]

\[
X \leftarrow \text{LayerNorm}(X)
\]

\[
x_i \leftarrow x_i + mlp[x_i] \quad \forall i
\]

\[
X \leftarrow \text{LayerNorm}(X)
\]
Pros: Good at modeling long range relationships.
Cons: Flattening the patches looses some amount of information.
• A CNN produces a low-resolution feature vector.
• A transformer operates on that feature vector.
• The upsampling is similar to that of U-Net

--> Best of both worlds?

Chen et al., TETCI’23
Regression

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

using
- stochastic gradient descent on mini-batches,
- dropout,
- hard example mining,
- ............
Body Pose Estimation
People and their Clothes

- Regress the body pose.
- Drape the garments on the body.
- Enforce consistency.
Crowd Counting

EPFL at lunchtime: The colors denote crowd density.

Liu et al. , CVPR’22
Brains vs Neural Networks

- Neural networks are said to “bio-inspired”.
- An excellent marketing argument but how true is it?
Monkey Cortex

(a) Dorsal pathway
   - Pink: Feed forward.
   - Cyan: Feed back.
   - Yellow: Horizontal

Lamme and Roelfsema, Trends in Neuroscience’00
Human Visual Cortex

Dorsal pathway

Ventral pathway
Recognize And Classify: Animal /No Animal

Subjects must raise their hand if they see an animal:

- 60 images
- 1 image per second

→ Measure their reaction time.

“Shape stimuli are optimally reinforcing each other when separated in time by \(\sim 60\) ms, suggesting an underlying recurrent circuit with a time constant (feedforward + feedback) of 60 ms.”

Drewes et al., *Journal of Neuroscience*, 2016
Adversarial Images

Szegedy et al. 2013
Brains vs Neural Networks

• Neural networks are said to “bio-inspired”.
• An excellent marketing argument but how true is it?

Not that good:

• Much feedback is involved in biological systems.
• We don’t need large databases to learn.
• We are not as susceptible to adversarial examples.

Neural nets are powerful but not the final answer!
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.
Deep Nets in Short

- Deep Neural Networks can handle huge training databases.
- When the objective function can be minimized, the results are outstanding.
- There are failure cases and performance is hard to predict.

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

- 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.

-> Beat the world champion in 2017.
Optional: ChatGPT

- At this point, the transformer layer is fed both the prompt and the already generated text.
- It uses this information to guess the next word.
- The process is then iterated.

Keys to ChatGPT success:
- The network looks back as far as needed.
- It uses a huge corpus.
- Human guided training.

Vaswani’17
Optional: ChatGPT

A little late for that.

That's going to get you killed!!

Correct. You will live.

It is.

What changed? Presumably, enough people complained and the system was re-retrained with correct responses.
Optional: Generative Models Taxonomy 2014 — 2023
What does it mean for Vision?

Two distinct approaches to increasing performance:

- Use ever larger training databases.
  - How do we build them?
  - How do we tame the computational explosion?

- Use existing knowledge to reduce the need for training data.
  - Physics-based knowledge.
  - Geometrical knowledge.

→ Self-supervised methods.