Introduction

Types of learning

1. Learning with a teacher: supervised learning.
2. Learning with a critic: reinforcement learning.
3. Learning on your own: unsupervised learning.

- How we study learning in neural networks:

  Fixed architecture, alter synapses/connection strengths

Classification

Input vectors **x** associated with output vectors **y**.
Learn mapping: **x** $\Rightarrow$ **y**.
Generalise to data not seen during learning. ("Training set" vs "test set" and also "validation set").

Approaches to classification

1. Logistic regression (binary outputs). Applied Statistics.
2. Naive Bayes. Machine Learning / probablistic modelling.
3. Multi-layer perceptron. Neural networks part I.
4. Support vector machines. Kernel methods.
5. Decision Trees and Forests.
6. Neural networks part II.

The perceptron (Rosenblatt 1957)

Single layer perceptron, with activation function

Notation:

- x_i: activity of input unit i (binary or [0,1]).
- y: activity of output unit.
- t: desired output (of use later). (μ superscript denotes training sample.)
- $f(\cdot)$: transfer function

Transfer function

Given total weighted input to neuron, what is its output?

1. identity: $f(z) = z$.
2. threshold: $f(z, \theta) = \begin{cases} 1 & \text{if } z \geq \theta \\ 0 & \text{otherwise} \end{cases}$
3. sigmoidal: $f(z) = \frac{1}{1+\exp(-kz)}$
4. tanh: $f(z) = tanh(z)$
5. rectified linear unit (ReLU): $f(z) = max(0, z)$

How to choose? One key property: differentiable.

Perceptron decisions

Whether a perceptrons output is 0 or 1 depends on whether $\sum_{i=1}^{N} w_i x_i$ is less or greater than θ.

The equation

$$\sum_{i=1}^{N} w_i x_i = \theta$$

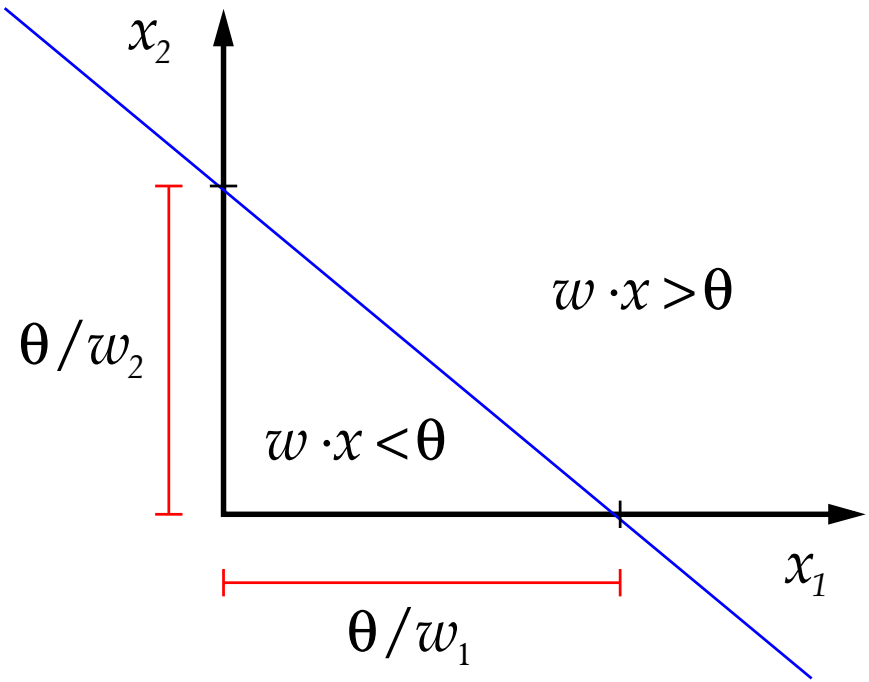
defines a hyperplane in N-dimensional space.

This hyperplane cuts the space in two.

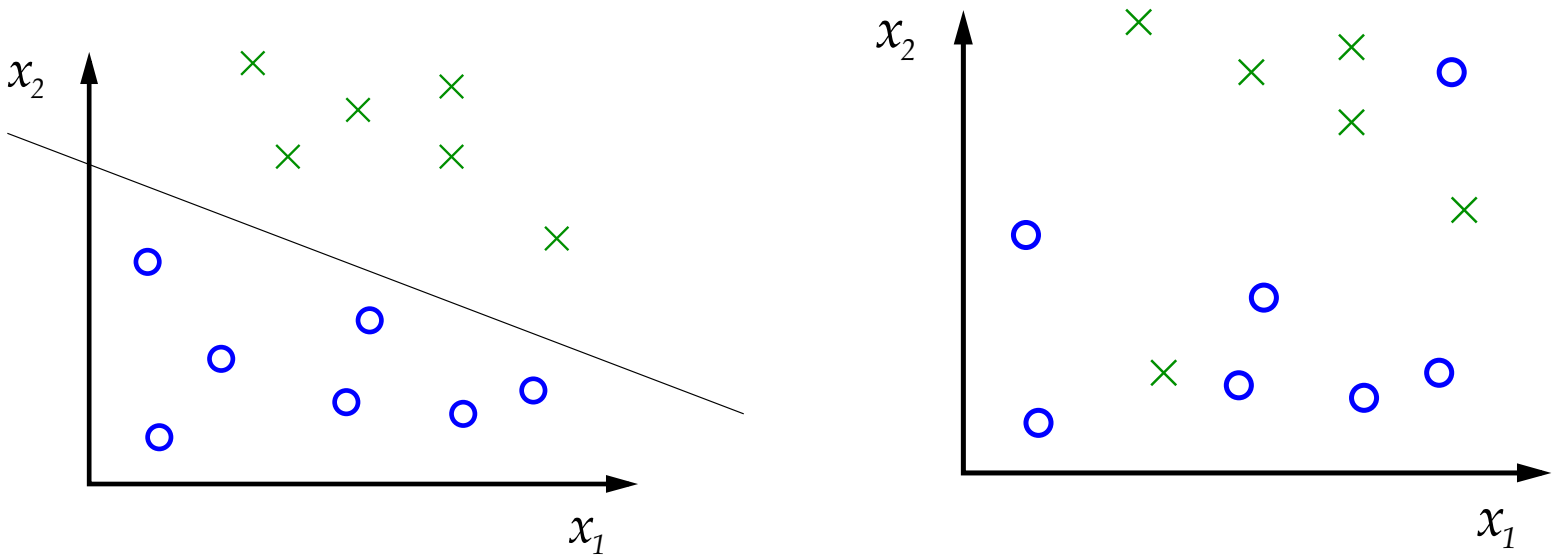
$$w_1 x_1 + w_2 x_2 = \theta$$

$$x_2 = \left(\frac{-w_1}{w_2} \right) x_1 + \frac{\theta}{w_2}$$

This is an equation for a straight line.



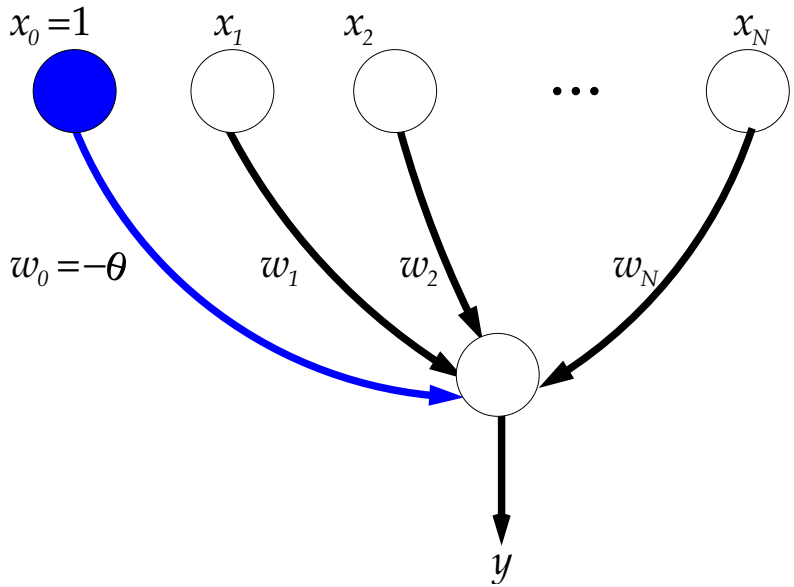
Linearly separable problems and the perceptron



Learning involves adjusting the values of w and θ so that the decision plane can correctly divide the two classes.

Threshold & Bias

The threshold, θ, can be treated as just another weight from a new input unit which always has a value of +1 (or -1). This new input unit is called the **bias** unit.

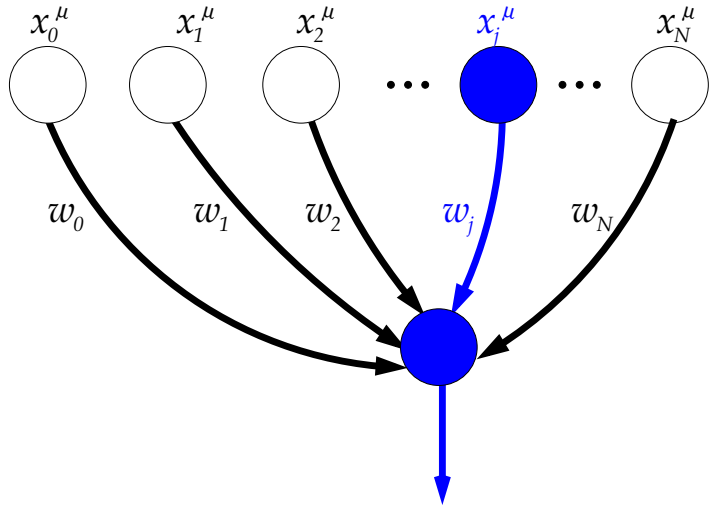


Instead of comparing $\sum_{i=1}^{N} w_i x_i$ with θ, we compare $\sum_{i=0}^{N} w_i x_i$ with 0, or

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^{N} w_i x_i > 0, \\ 0 & \text{if } \sum_{i=0}^{N} w_i x_i \leqslant 0. \end{cases}$$

Now, learning is about jiggling **weights** only.

Intuitively... (positive valued inputs)



$$y^\mu = \text{step}\left(\sum_{i=0}^{N} w_i x_i^\mu\right)$$

Consider w_j's contribution to $\sum_i w_i x_i^\mu$ in different cases:

1. $y^\mu = t^\mu$ Perceptron has classified input μ correctly – change nothing
2. $x_j^\mu = 0$ Changing w_j will not affect the $\sum_i w_i x_i^\mu$ – change nothing
3. $x_j^\mu \neq 0, y^\mu < t^\mu$ The sum $\sum_i w_i x_i^\mu$ is too low – so increase it
4. $x_j^\mu \neq 0, y^\mu > t^\mu$ The sum $\sum_i w_i x_i^\mu$ is too high – so decrease it

The local rule:

$$\boxed{\Delta w_j \propto (t^\mu - y^\mu)\, x_j^\mu}$$

Perceptron learning rule

$$y = f(\mathbf{w} \cdot \mathbf{x}) \qquad \text{e.g.} \quad f(z) = 1/(1+\exp(-z)), \quad f(z) = z$$

$$E = \frac{1}{2}(t-y)^2 \qquad t \text{ is target output}$$

$$\Delta w_j = -\epsilon \frac{\partial E}{\partial w_j} = -\epsilon \frac{\partial E}{\partial y} \frac{\partial y}{\partial w_j} =$$

This is the method of gradient descent with learning-rate parameter ϵ.

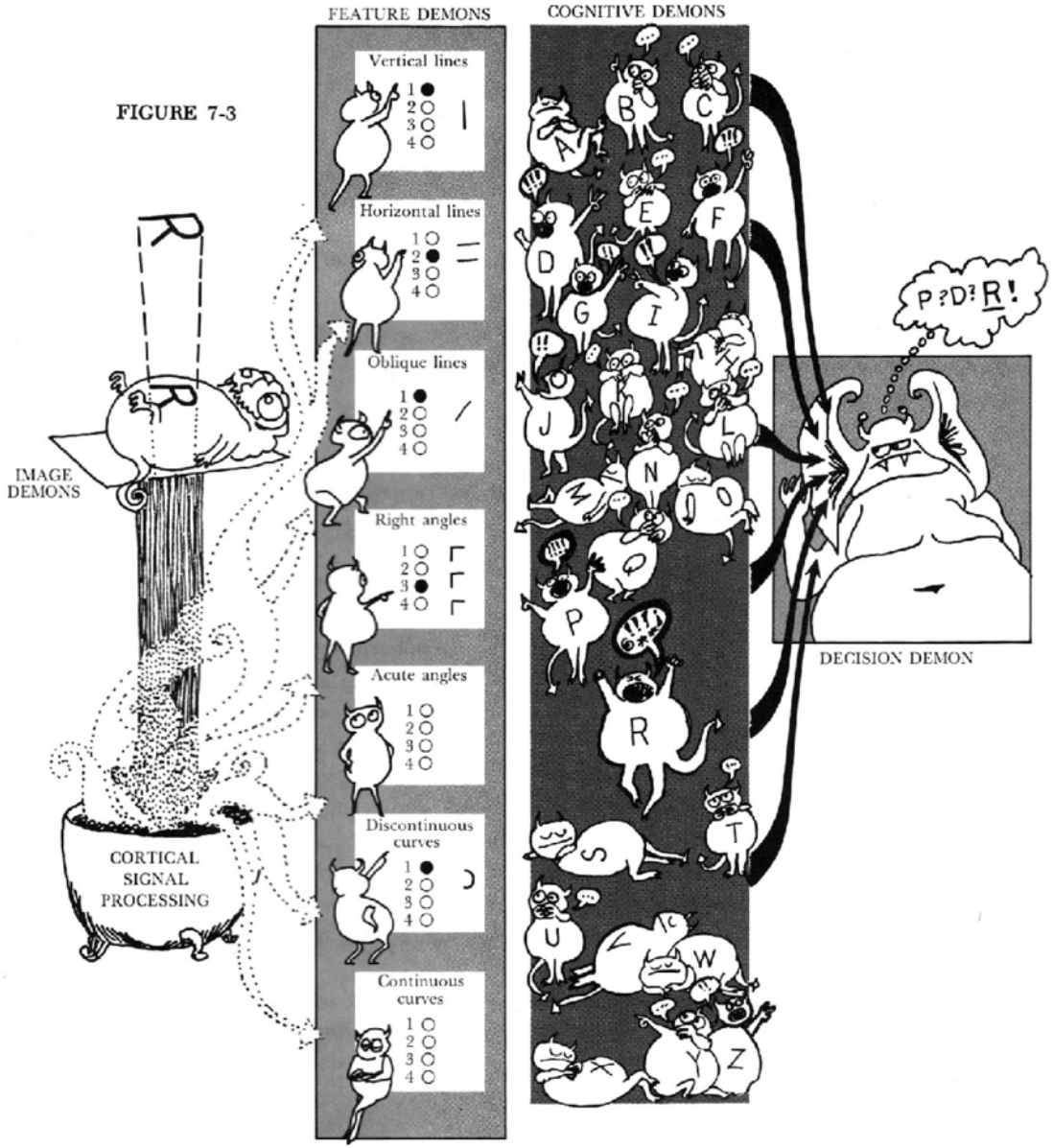
Perceptron pros and cons

- The *perceptron convergence theorem* (Dayan and Abbott, page 327) guarantees that solution will be found iff there is a linearly separable solution.
- Many complex problems are not linearly separable, e.g. XOR problem.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Multi-layer perceptrons (MLPs)

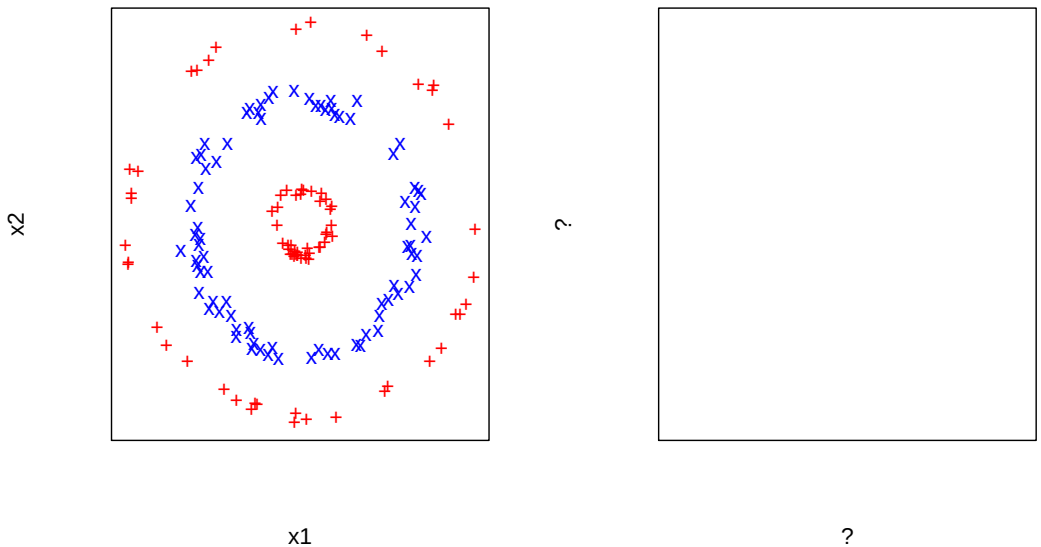
The importance of features / 1



(Lindsay and Norman's view of Selfridge's Pandemonium model, 1959).

The importance of features / 2

- Find the right features to make the task solvable:



- Engineering features by hand is hard.
- Neural networks learn features that they find important.

How many layers of features do you need?

One hidden layer is all you need **in theory** to make a “universal approximator” (Cybenko 1989; Hornik 1991).
With linear transfer functions how many layers do we need?

Neural networks and linear algebra

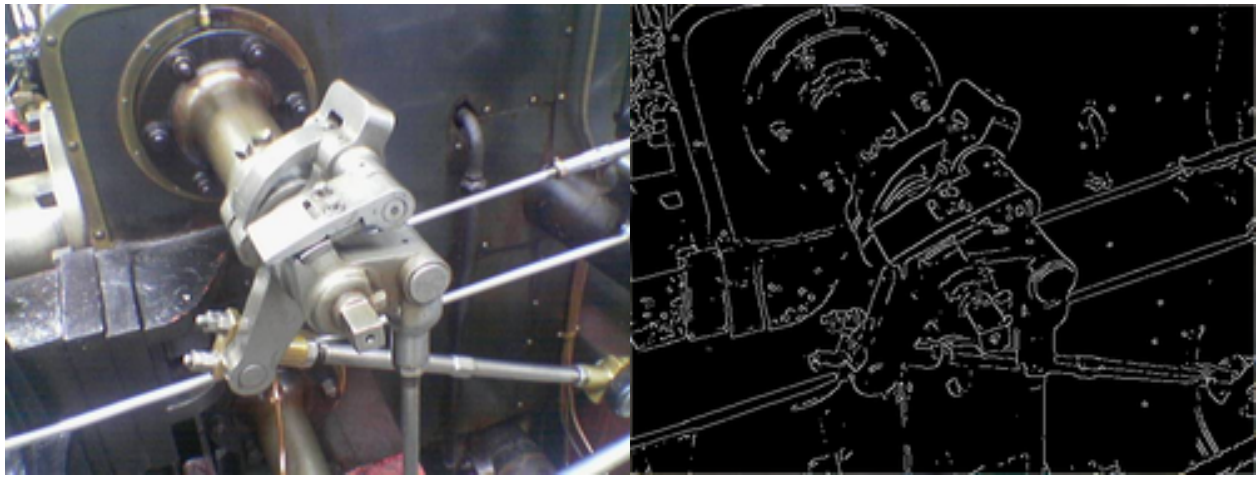
- Activation of a layer of neurons stored in a **vector**.
- Synapses from one layer to another stored in a **weight matrix**: $\mathbf{W}_{ji}$ is strength of connection from unit i in one layer to unit j in the next layer.
- "The single key fact about vectors and matrices is that each vector represents a point located in space, and a matrix moves that point to a different location. Everything else is just details." (Stone 2019, Appx. C).

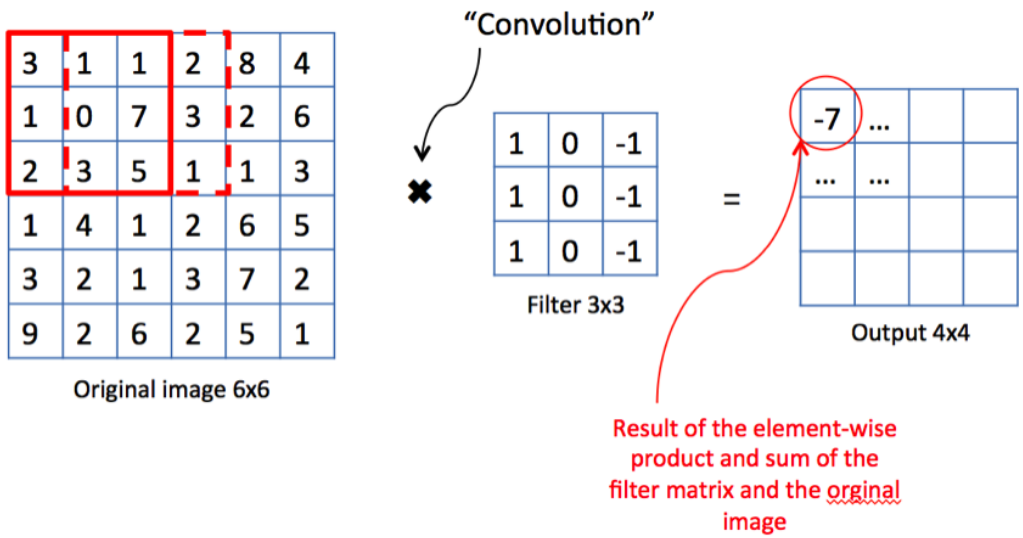
drawing of weight matrices

Images

Convolution

How to do image processing with a neural network?

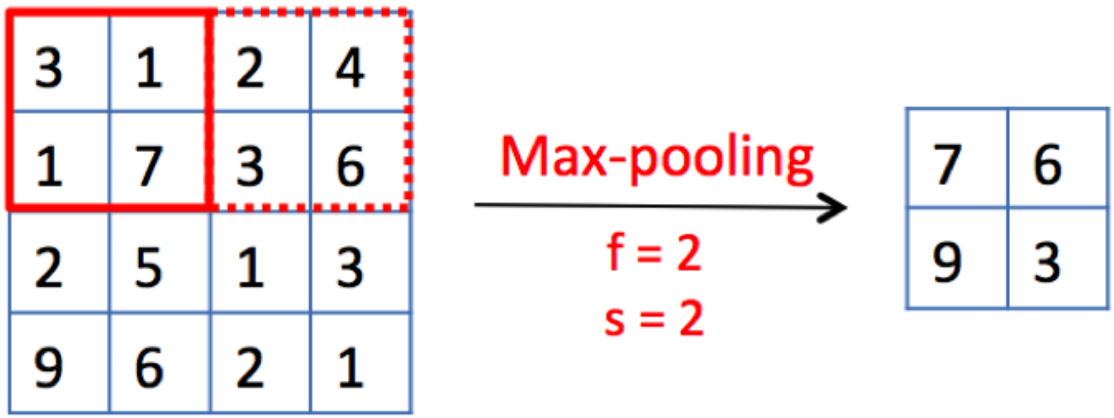




https://en.wikipedia.org/wiki/Canny_edge_detector

https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524

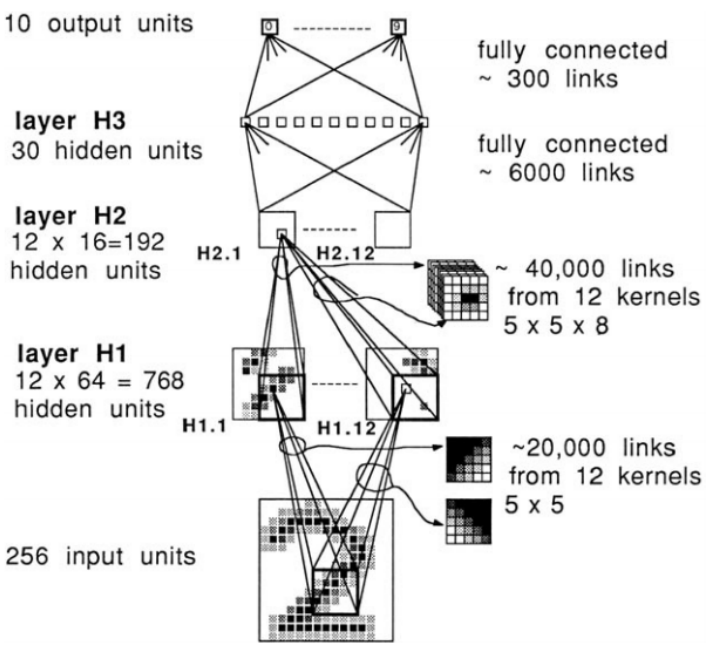
Pooling

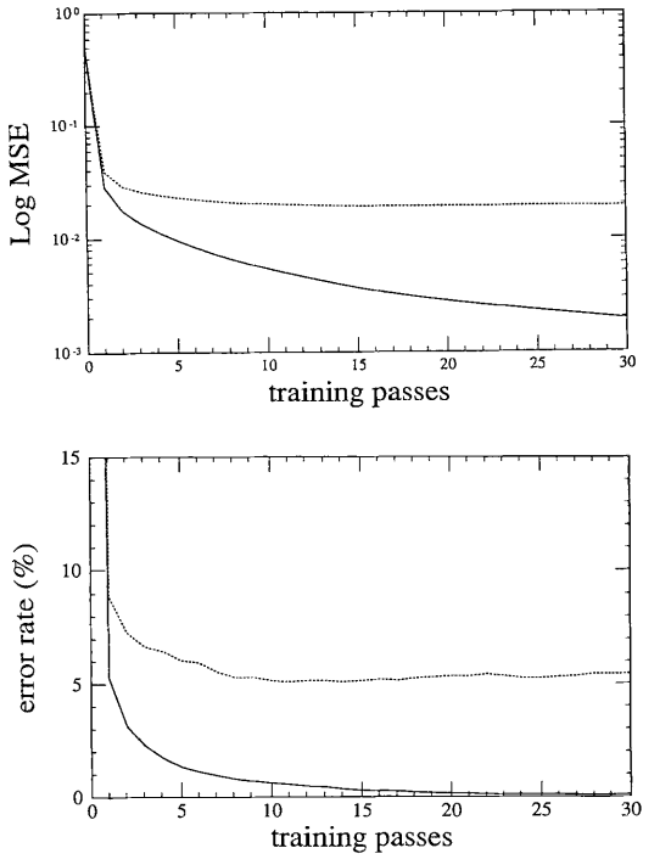


https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524

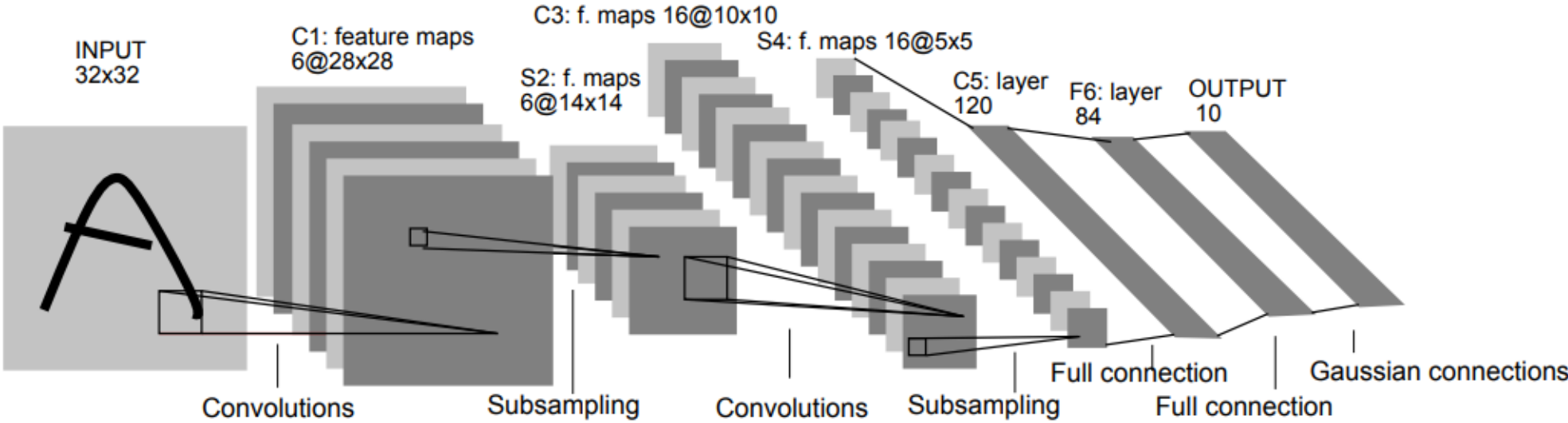
LeCun et al (1989)

LeCun Y et al (1989) Backpropagation Applied to Handwritten Zip Code Recognition. Neural Comput 1:541–551.





LeCun et al (1998)



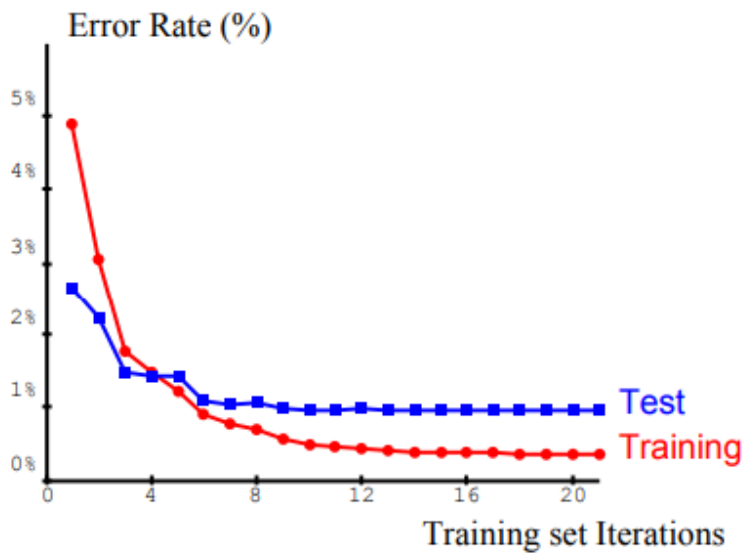
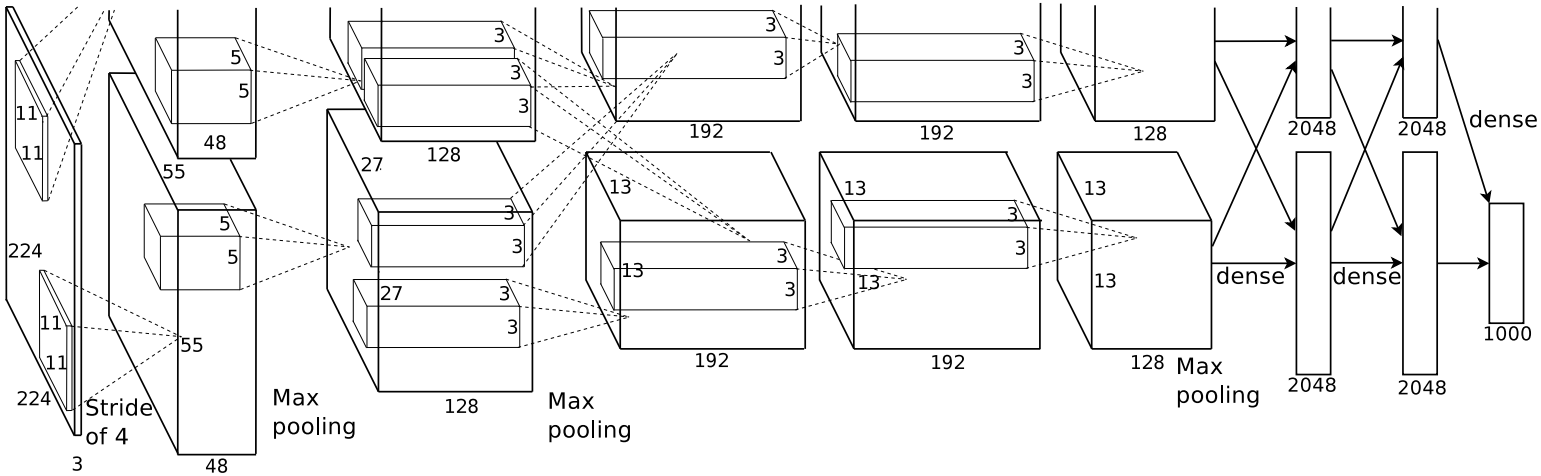


Image classification (Krizhevsky et al 2012)

- Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet Classification with Deep Convolutional Neural Networks. In: Advances in Neural Information Processing Systems 25, Burges CJC, Bottou L, Weinberger KQ, eds), pp 1097–1105.
- Annual competition. 1,000 examples of 1,000 classes.
- Rotating/translating of training images to make 'new' samples.
- Top-1 (37.5%) and top-5 (17.0%) error rate exceed previous state of the art. Won 2012 competition with top-5 test error rate of 15%; second had error rate of 26%.

ImageNet architecture (Krizhevsky et al. 2012)



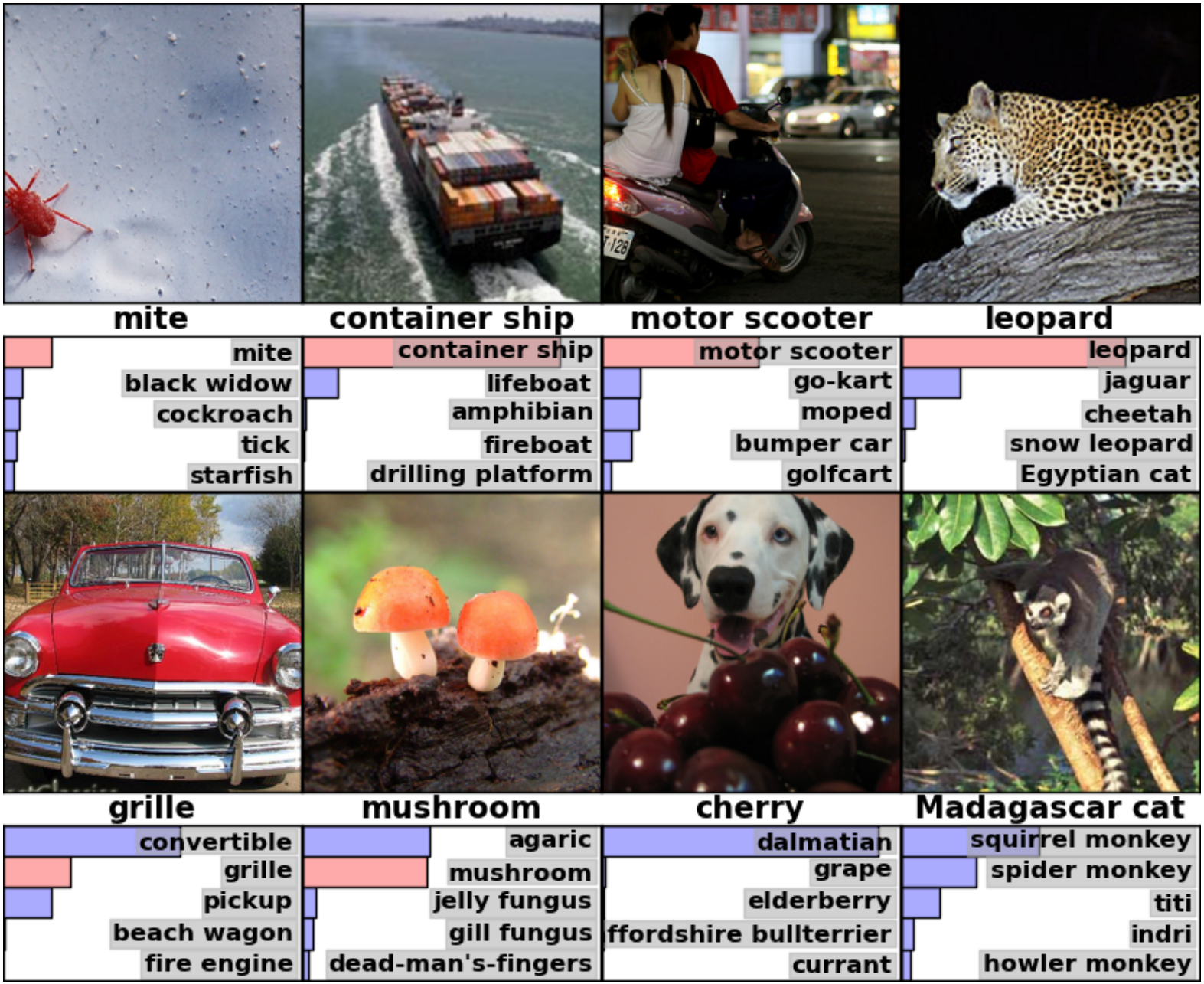
Split on 2 GPUs.

150K pixels (224 x 224 x 3) into 8 layer network:

253K-186K-65K-65K-43K-4K-4K-1K.

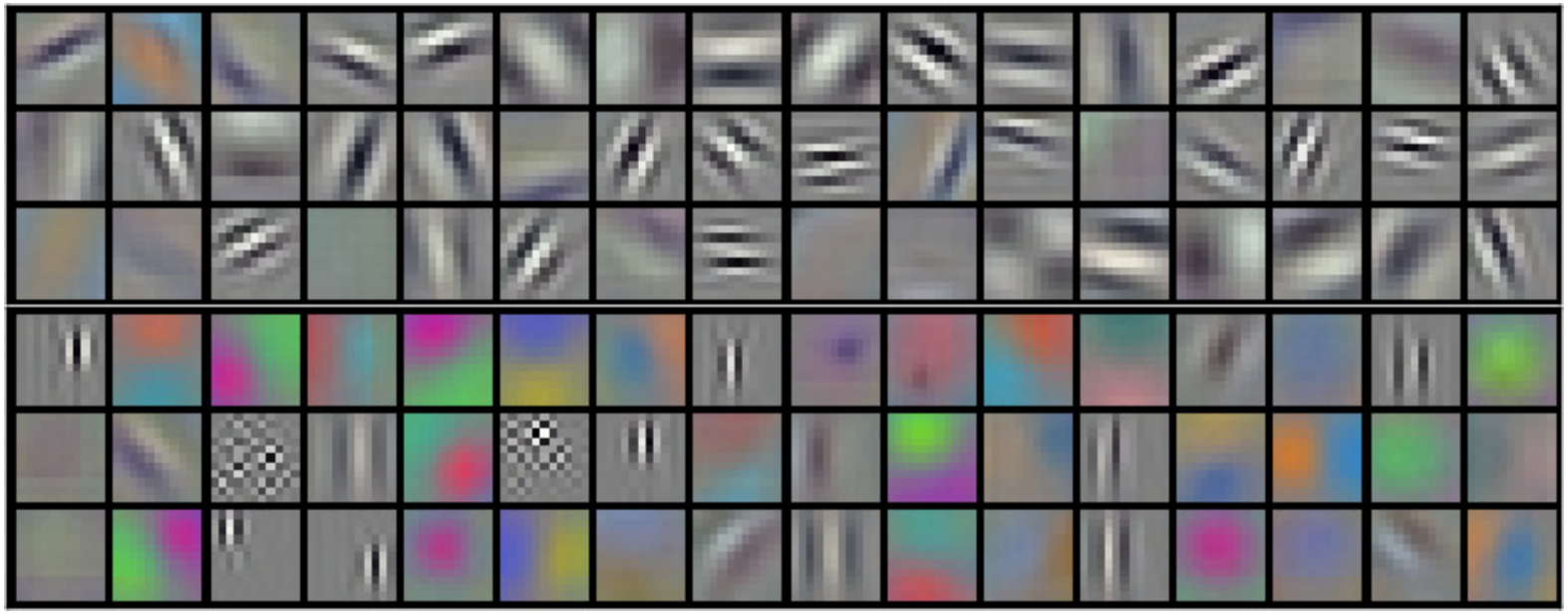
i.e. 650,000 neurons (excluding input), 60 million parameters.

ImageNet performance



ImageNet close-matches

ImageNet RFs



“It is notable that our network’s performance degrades if a single convolutional layer is removed. For example, removing any of the middle layers results in a loss of about 2% for the top-1 performance of the network. So the depth really is important for achieving our results.”

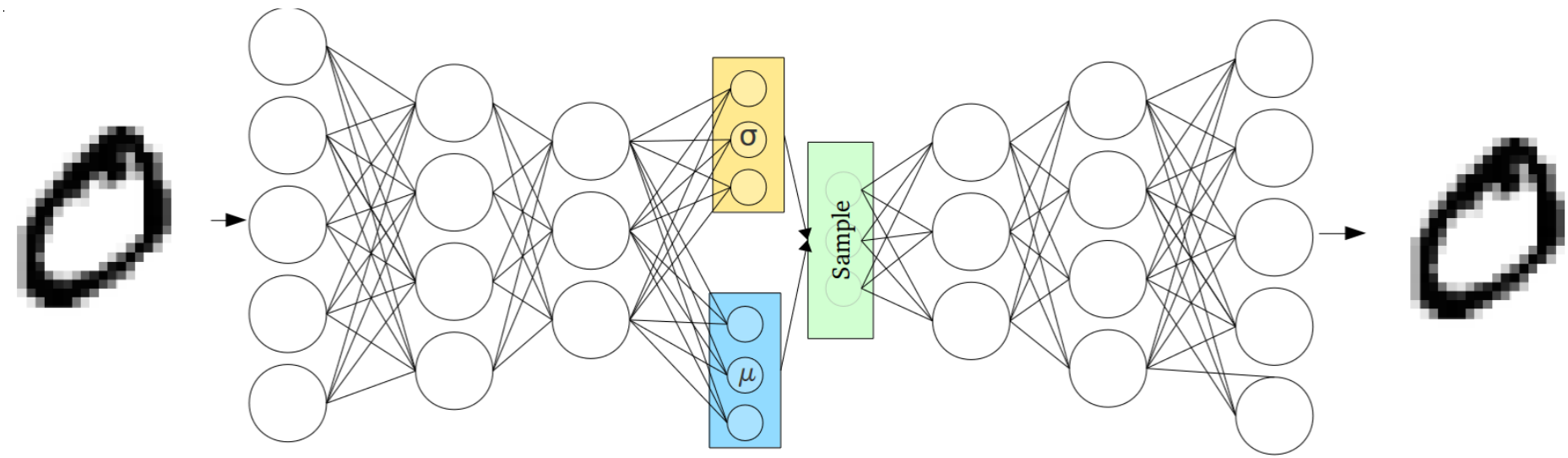
Life since AlexNet

1. 2012: AlexNet (Krizhevsky et al 2012). Top-5 error of 17% (top-1 of 37%).
2. 2014: GoogLeNet / “Inception” (Szegedy et al 2014). Top-5 error of 6.67%.
3. 2015: ResNet (He et al 2015). First to beat human, top-5 error of 3.75%. 152 layers.
4. ResNet with 1001 layers (He et al 2016).

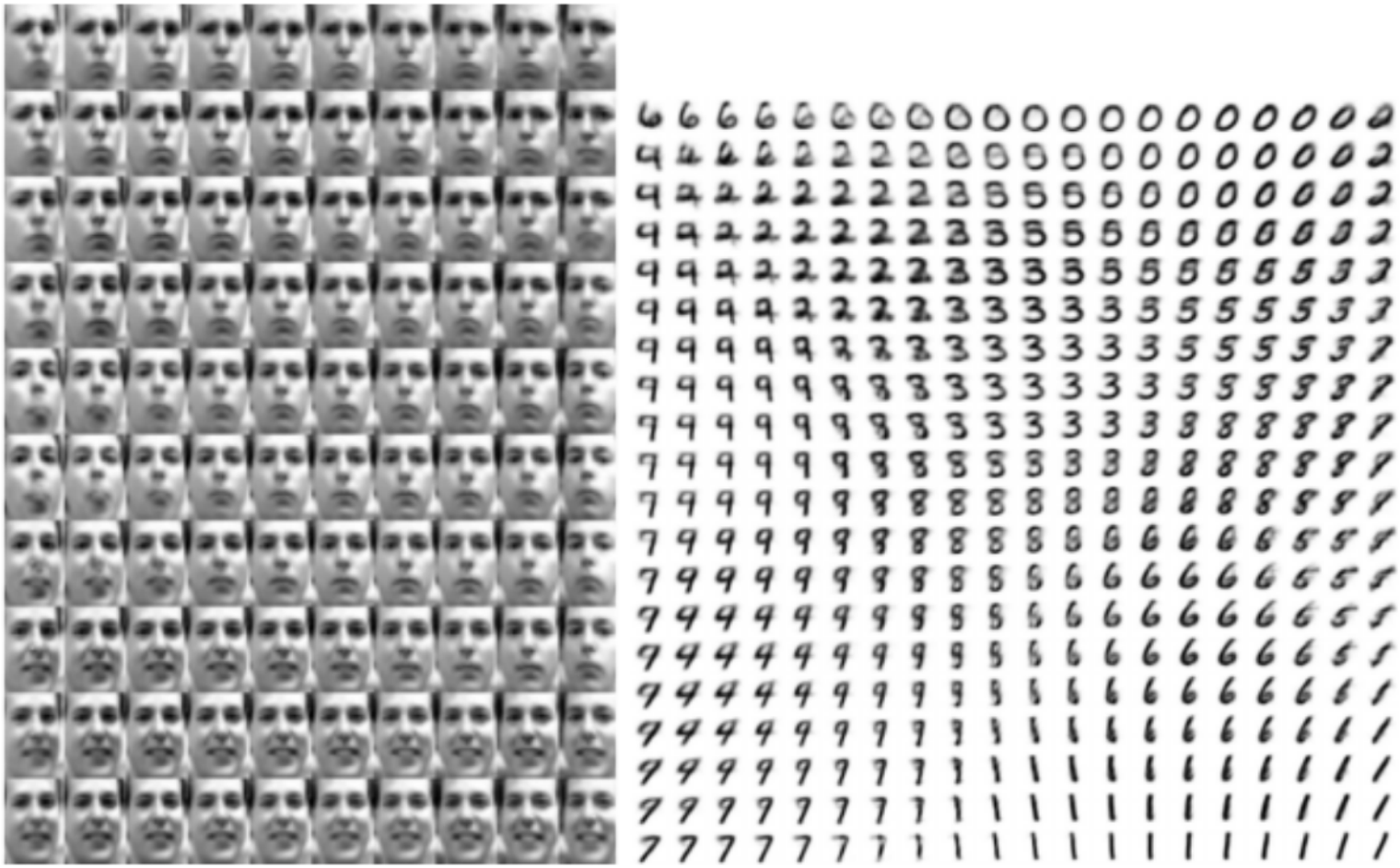
Transfer learning using these networks to provide features: freeze main network and learn layers on top.

Variational autoencoders

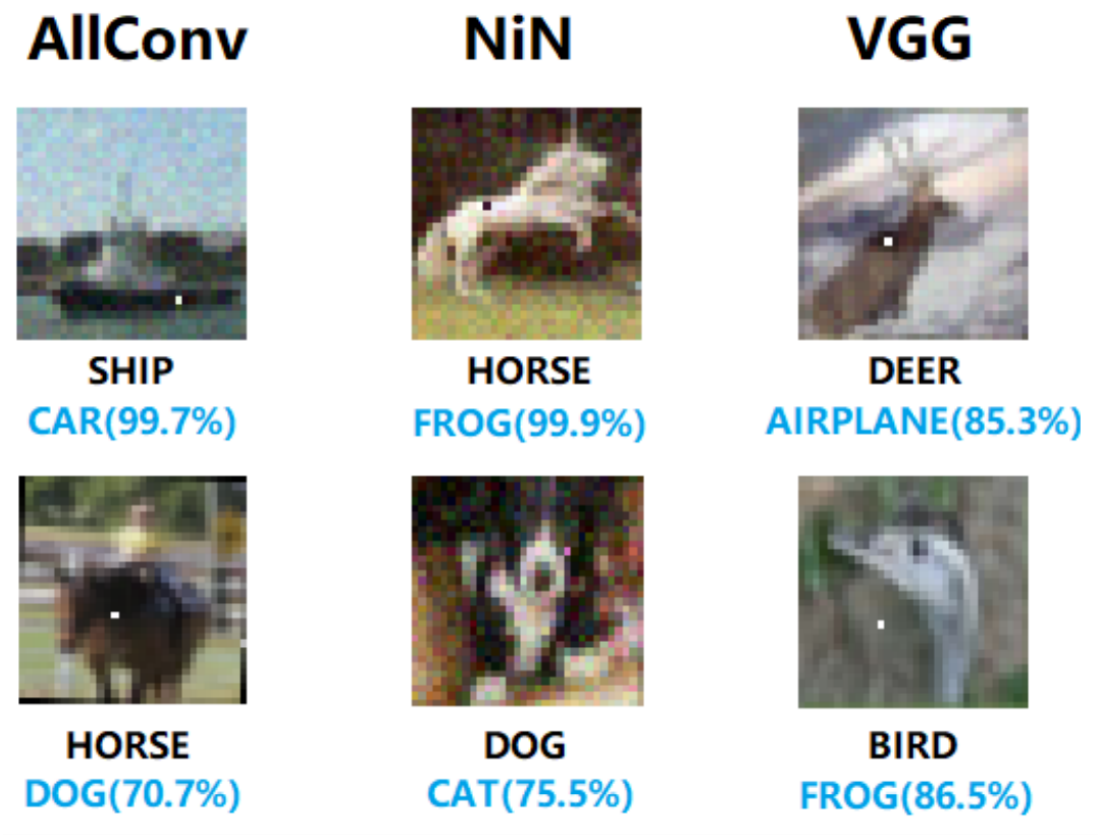
https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf



Sampling the latent space (Goodfellow , Figure 20.6)



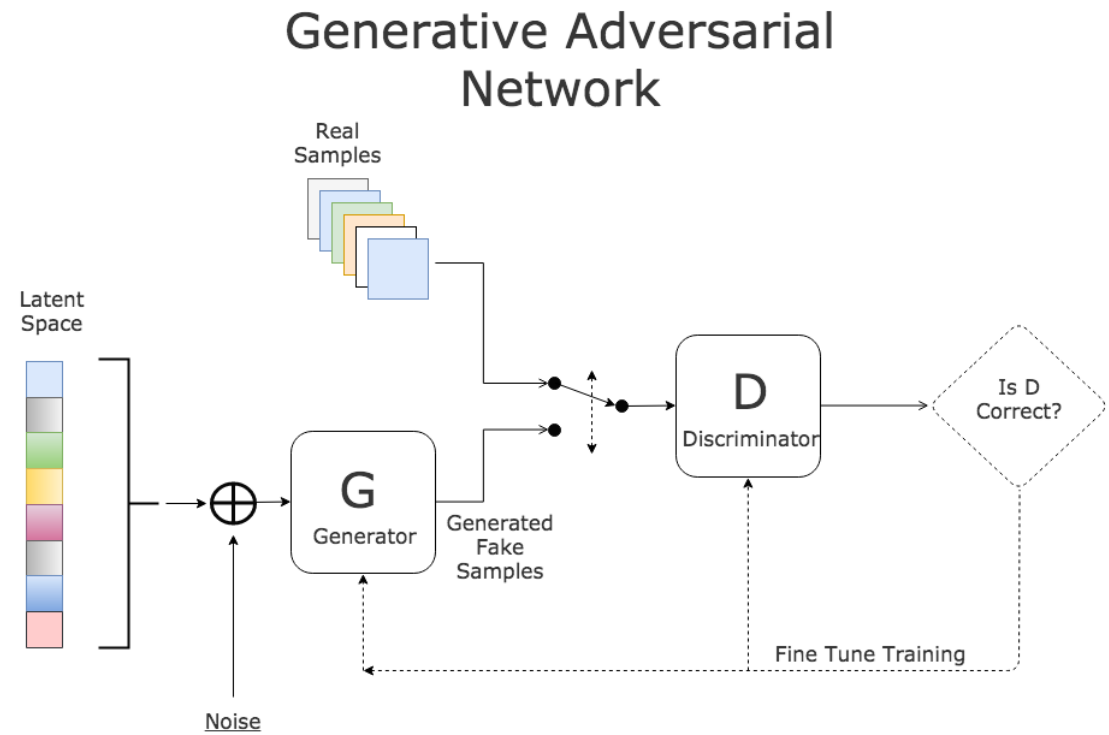
Fooling Deep Networks with adversarial samples



Su et al (2017)

See also `https://arxiv.org/pdf/1707.08945.pdf` for robust attacks on stop signs.

Generative adversarial neworks (Goodfellow et al 2014)



1. **Discriminator** spots real vs fake training samples. Adjust weights to increase discrimination.
2. **Generator** adjusts weights to generate images that are more likely to be classified as training images.

Source: `https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html`

For further information:

`http://bamos.github.io/2016/08/09/deep-completion/#step-1-interpreting-images-as-samples-from-a-probability-distr`

Radford et al. (2015), figure 4

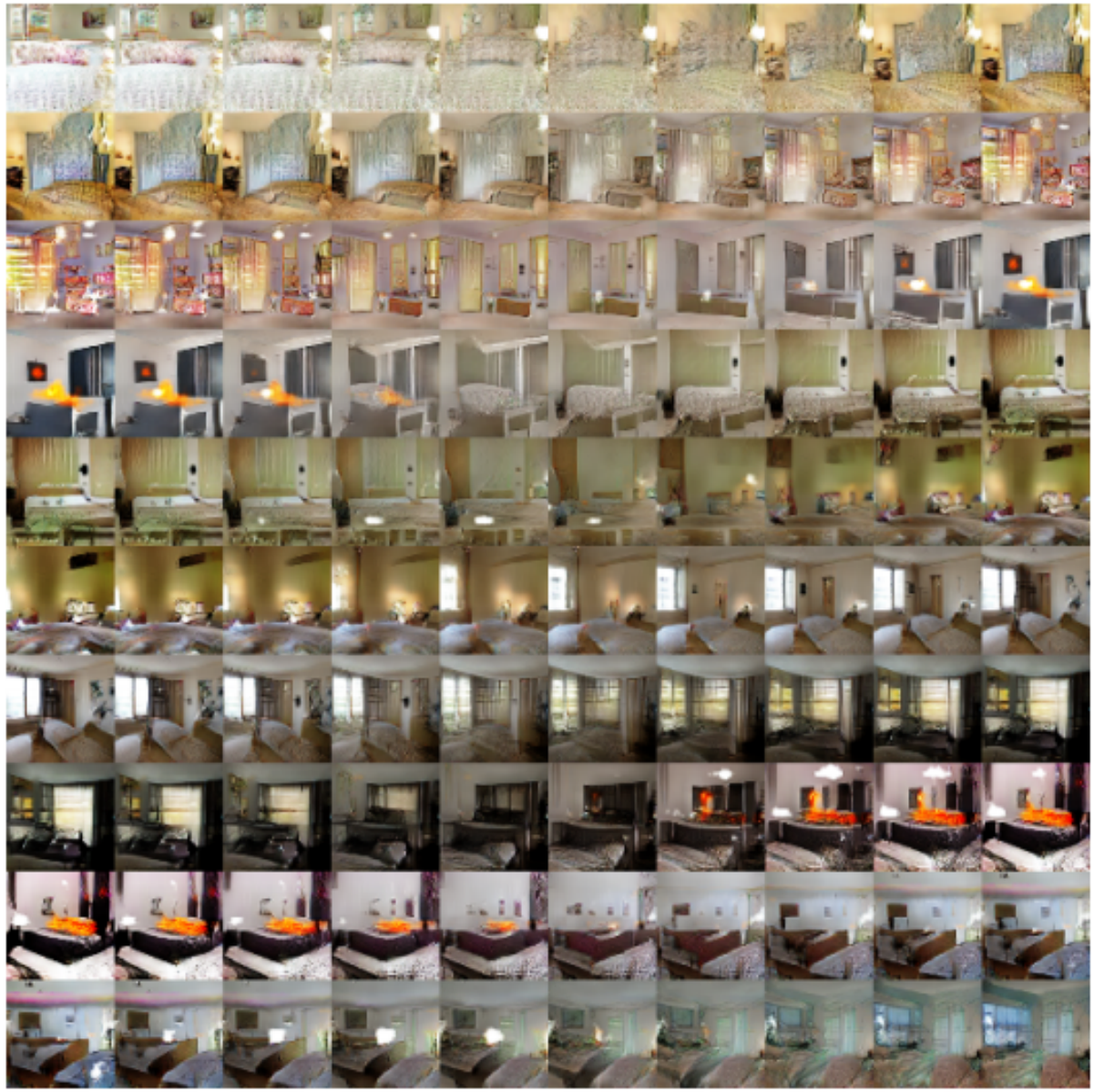


Figure 4: Top rows: Interpolation between a series of 9 random points in Z show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom. In the 6th row, you see a room without a window slowly transforming into a room with a giant window. In the 10th row, you see what appears to be a TV slowly being transformed into a window.