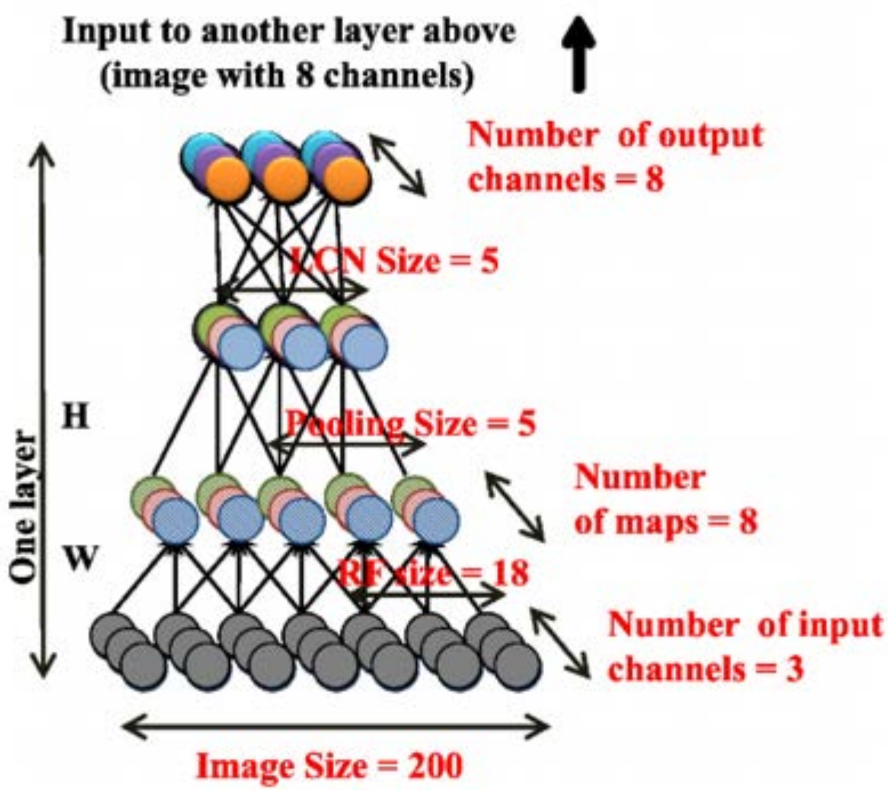
DEEP LEARNING CRASH COURSE

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



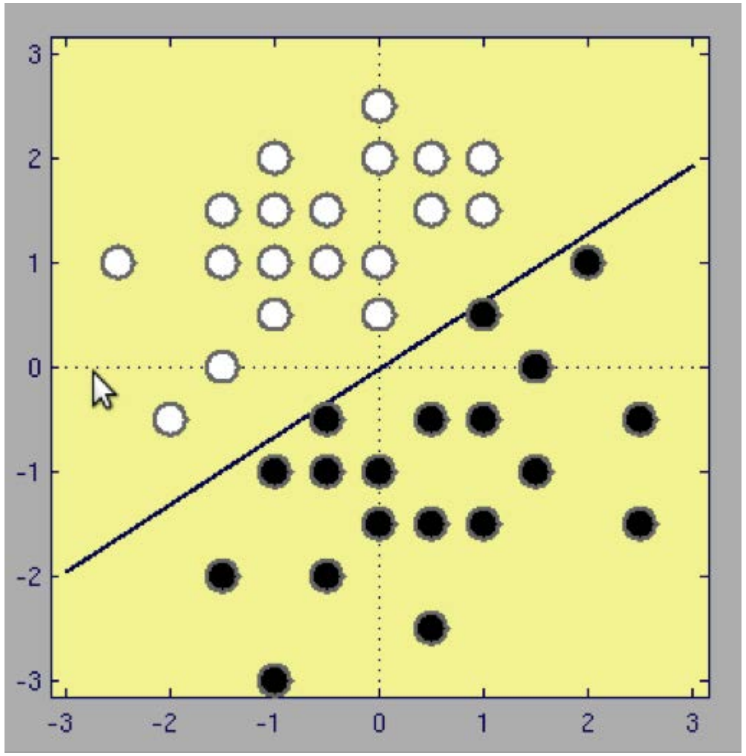
M.A. Nielsen. Neural Networks and Deep Learning, 2015
http://neuralnetworksanddeeplearning.com/

ARTIFICAL INTELLIGENCE

1997: Deep Blue beats chess World Champion

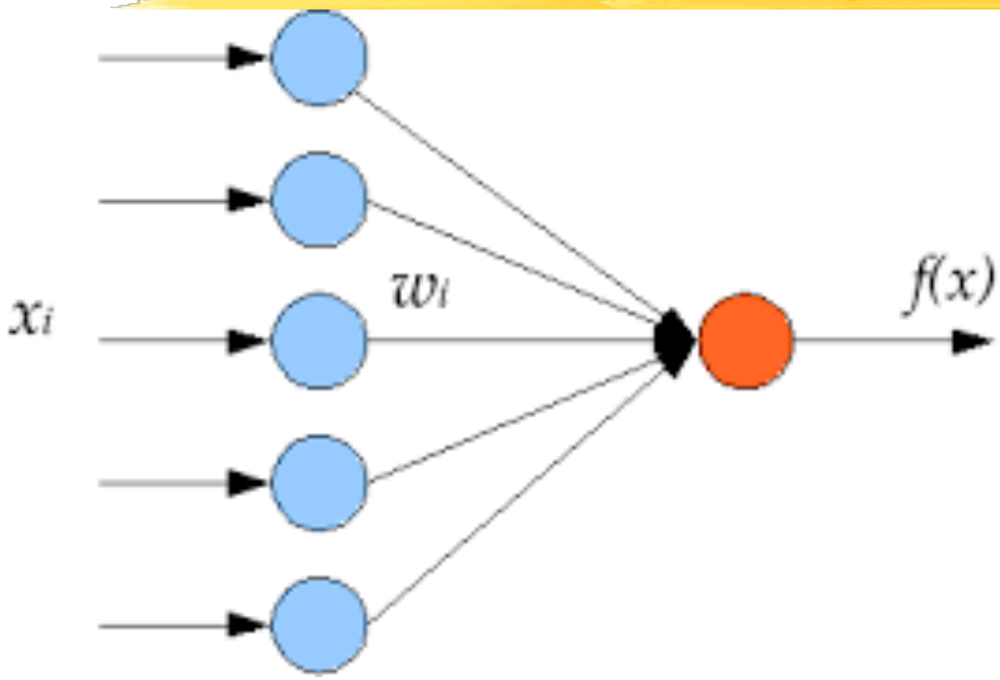
2016: AlphaGo beats go world champion

LINEAR CLASSIFICATION



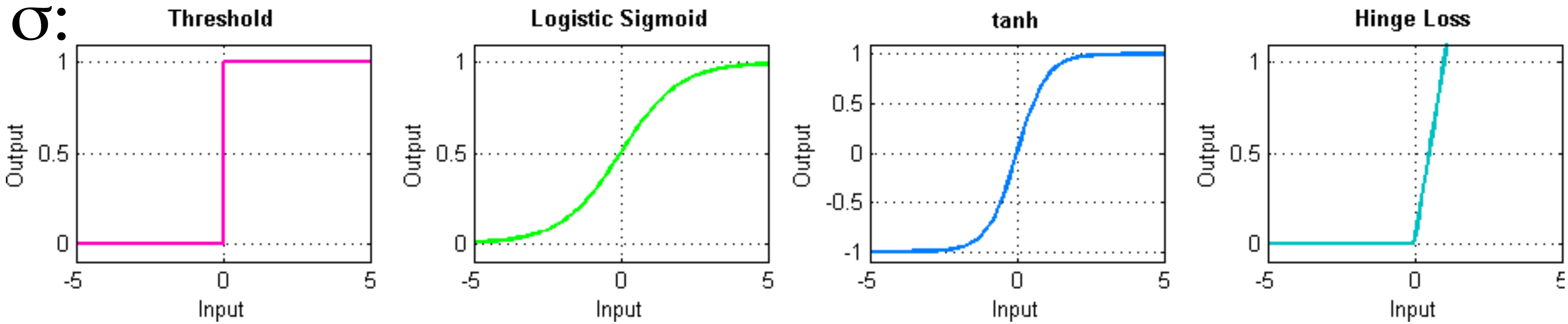
$$f(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{w} \cdot \mathbf{x} + \mathbf{b} \geq 0, \\ 0 \text{ otherwise.} \end{cases}$$

SINGLE LAYER PERCEPTRON

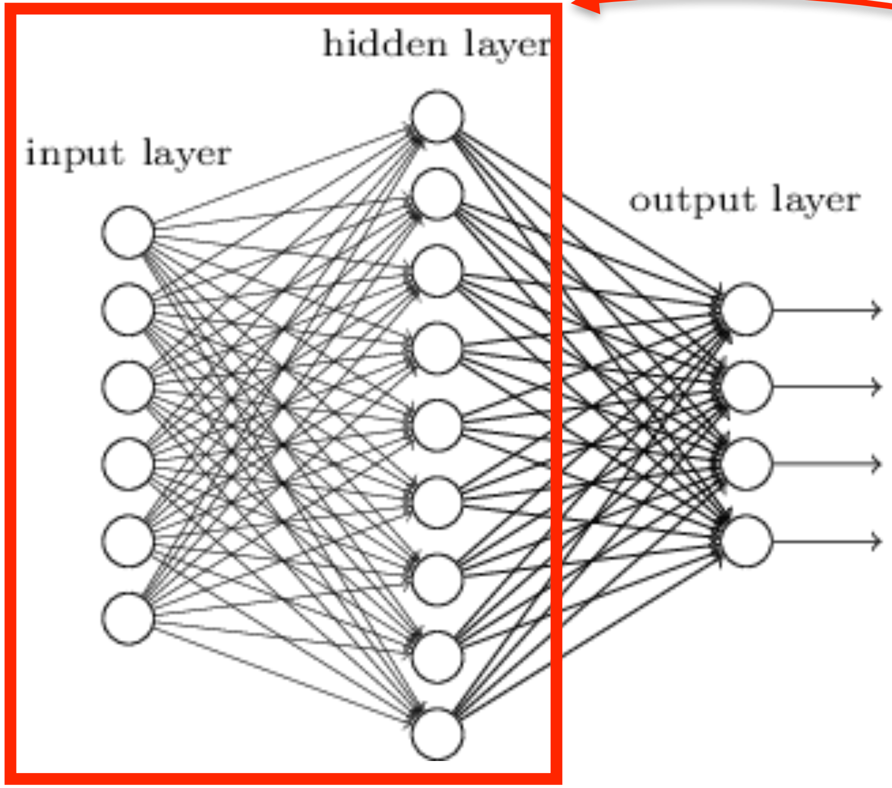


$$f(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + \mathbf{b})$$

σ:



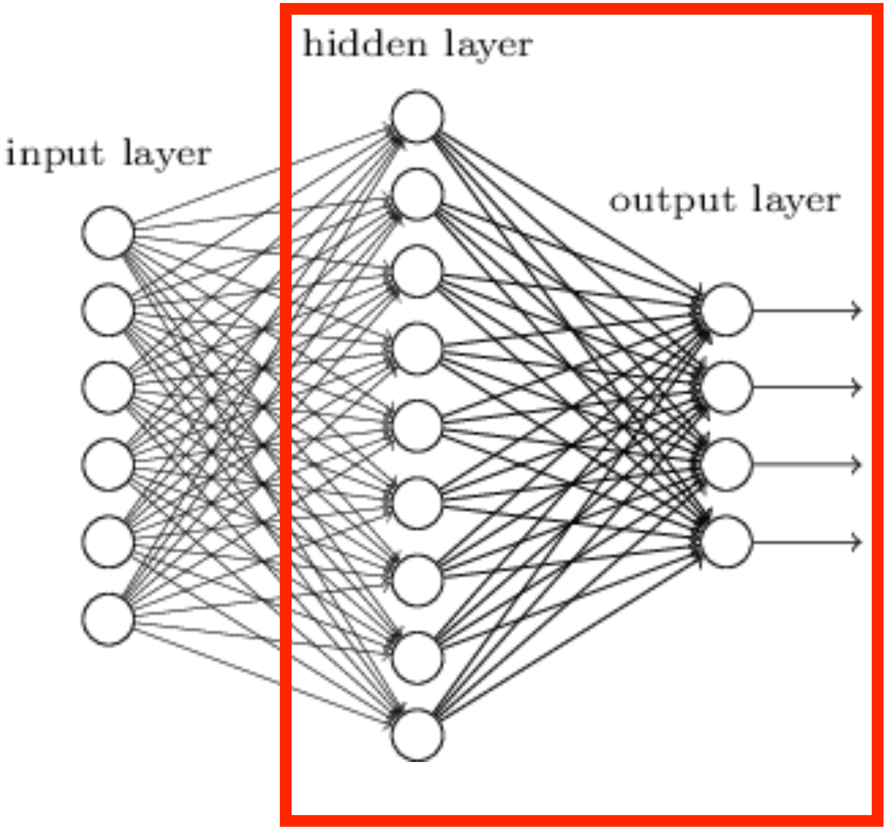
MULTILAYER PERCEPTRON



$$\boxed{\mathbf{h} = \sigma_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)}$$
$$\mathbf{y} = \sigma_2(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2)$$

- The process can be repeated several times to create a vector **h**.

MULTILAYER PERCEPTRON



$$\mathbf{h} = \sigma_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\boxed{\mathbf{y} = \sigma_2(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2)}$$

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

- —> This output is a **differentiable** function of the weights.

BINARY CASE

Given a training set $\{\mathbf{x}_n, t_n\}_{1\leq n\leq N}$ where $t_n \in \{0,1\}$, minimize

$$
\begin{aligned}
E(\mathbf{W},\mathbf{b}) &= -\frac{1}{N}\sum_{1}^{N}\left[t_i \log(y_i) + (1-t_i)\log(1-y_i)\right] , \\
y_i &= f(\mathbf{x}_i) \\
&= \sigma(\mathbf{W}_2(\sigma(\mathbf{W}_1\mathbf{x}_i + \mathbf{b}_1)) + \mathbf{b}_2),
\end{aligned}
$$

that is, minimize the number of misclassified samples.

Since E is a differentiable function of **W** and **b**, this can be done using a gradient-based technique, also known as back propagation.

MULTI-CLASS CASE

In the multi-class case, the probability that input vector $\mathbf{x}$ belongs to class c is taken to be

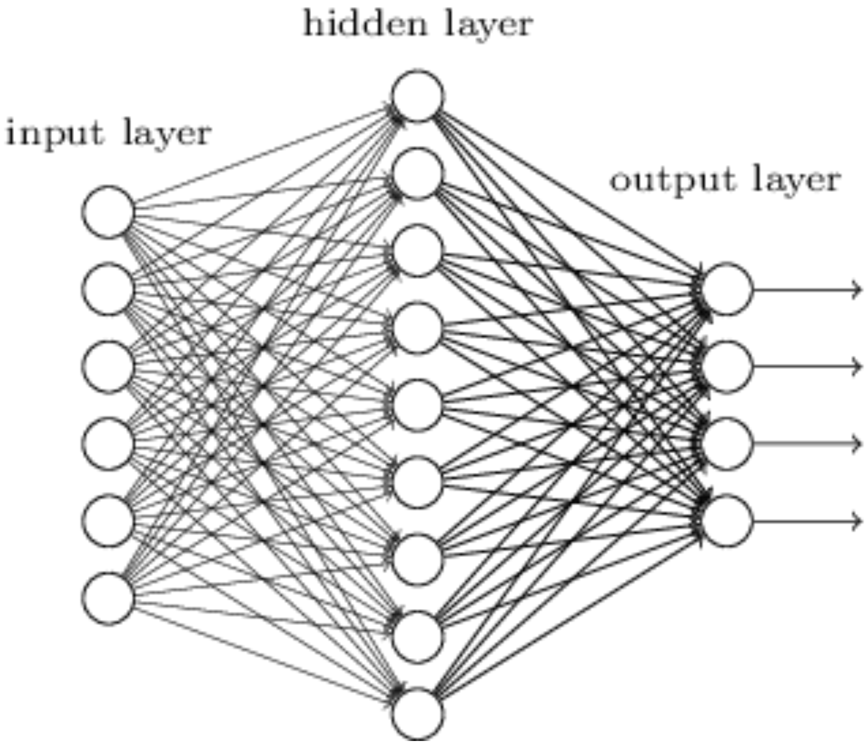
$$P(\mathbf{x} \in c | \mathbf{W}, \mathbf{b}) = \frac{y^c(\mathbf{x}; \mathbf{W}, \mathbf{b})}{\sum_k y^k(\mathbf{x}; \mathbf{W}, \mathbf{b})} \; .$$

Given a training set $\{\mathbf{x}_n, t_n^1, \ldots, t_n^C\}_{1 \leq n \leq N}$ where $t_n^c \in \{0, 1\}$, minimize

$$E(\mathbf{W}, \mathbf{b}) = -\sum_n \sum_c t_n^c \log(P(\mathbf{x}_n \in c | \mathbf{W}, \mathbf{b})) \; .$$

Since E remains a differentiable function of **W** and **b**, this can also be done using a gradient-based technique, also known as back propagation.

HINGE LOSS OR RELU



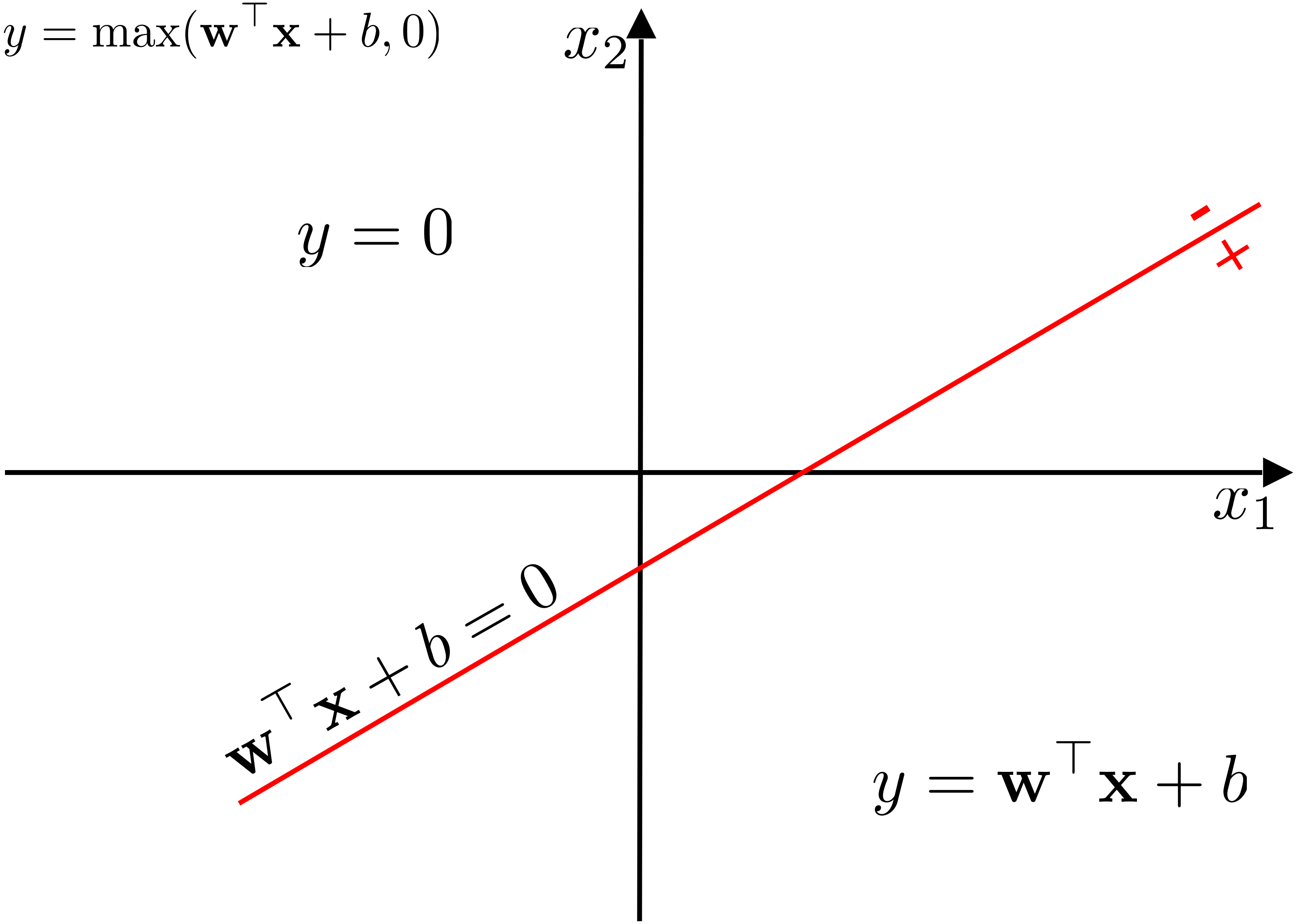
$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{y} = \sigma(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2)$$

$$\sigma(\mathbf{x}) = \max(0,\mathbf{x}).$$

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

ONE SINGLE HYPERPLANE

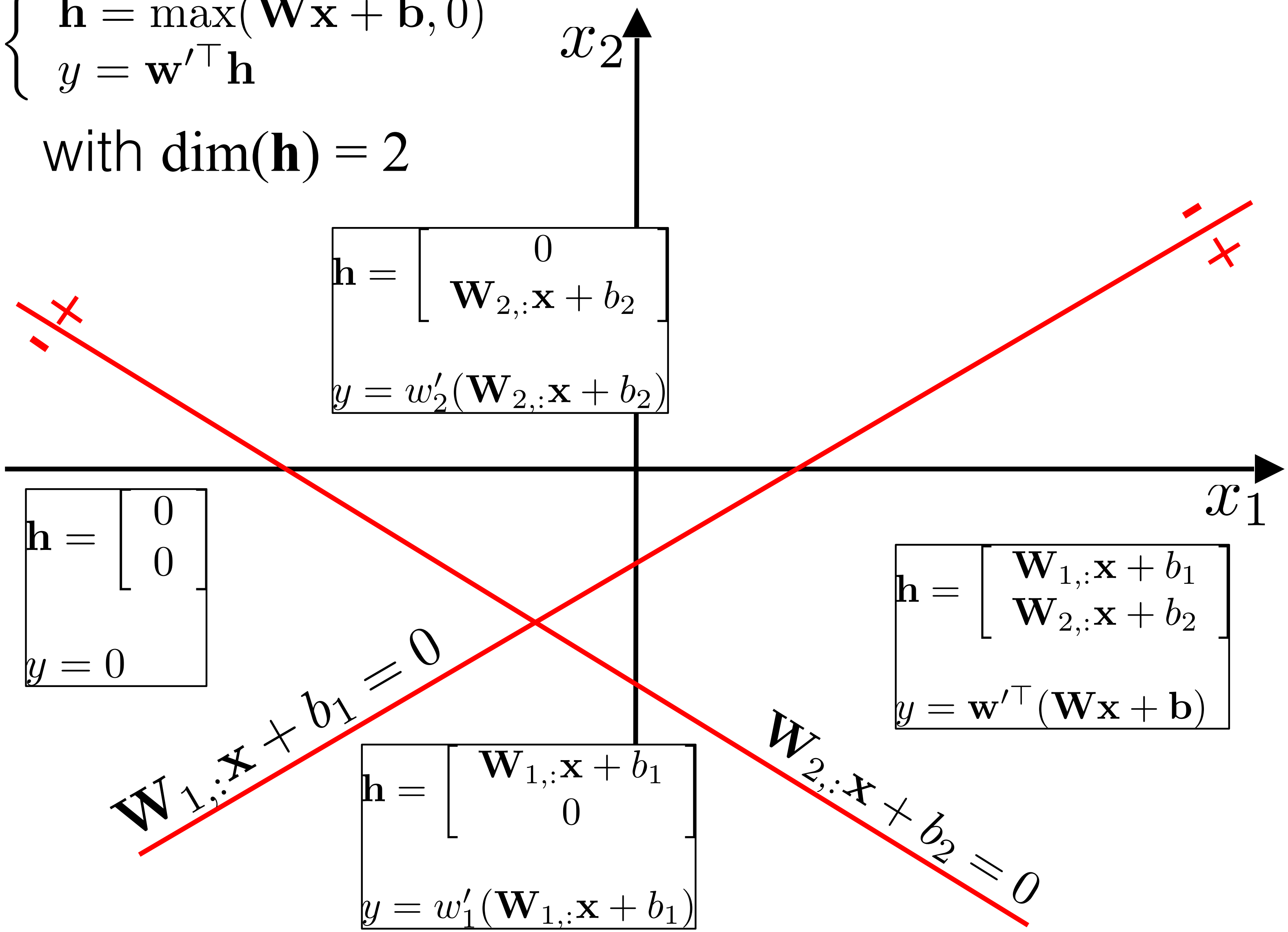
$$y = \max(\mathbf{w}^\top \mathbf{x} + b, 0)$$



TWO HYPERPLANES

$$\begin{cases} \mathbf{h} = \max(\mathbf{W}\mathbf{x} + \mathbf{b}, 0) \\ y = \mathbf{w}'^\top \mathbf{h} \end{cases}$$

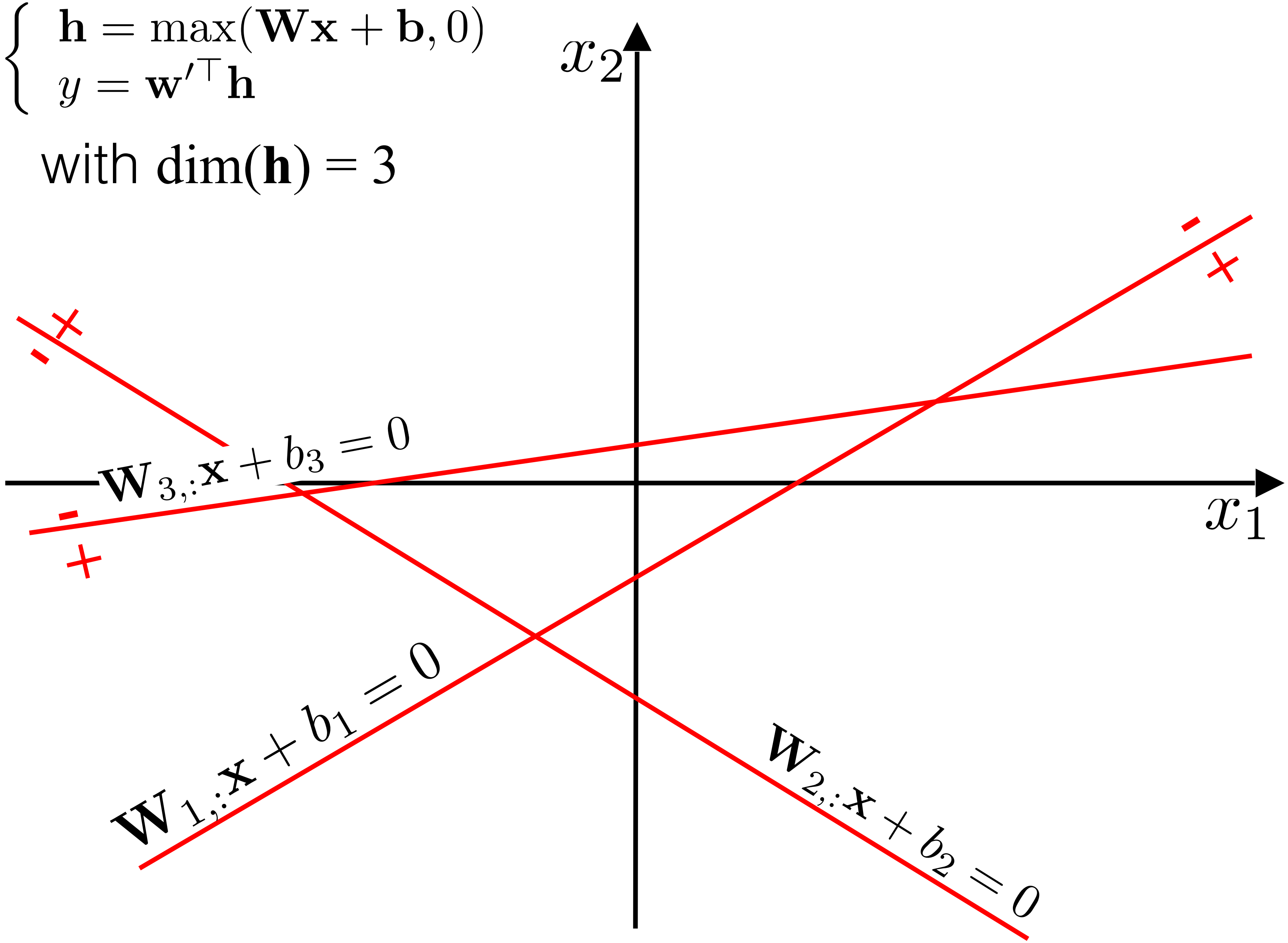
with $\dim(\mathbf{h}) = 2$



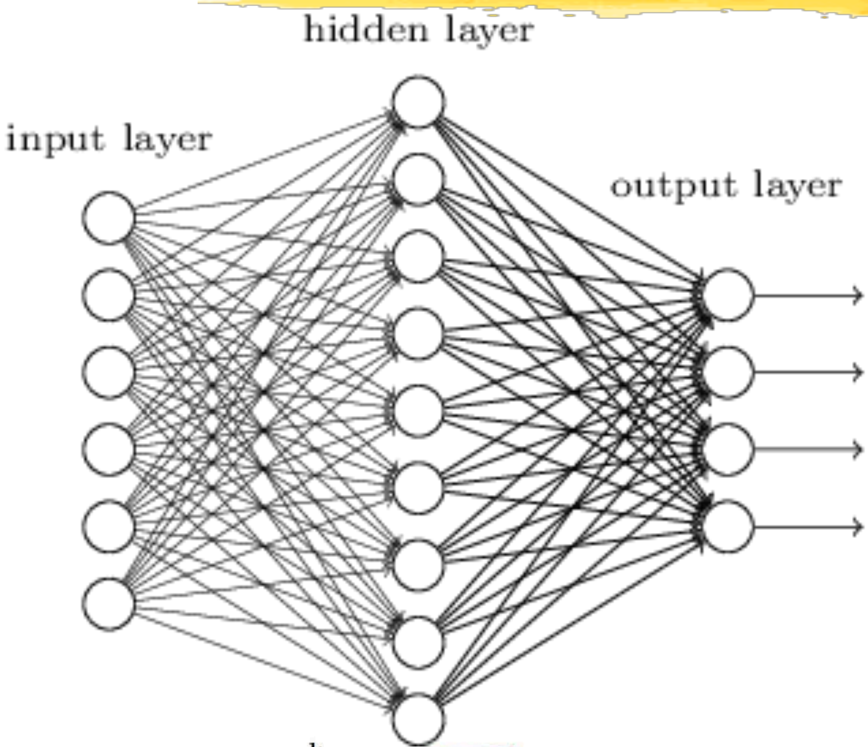
THREE HYPERPLANES

$$\begin{cases} \mathbf{h} = \max(\mathbf{W}\mathbf{x} + \mathbf{b}, 0) \\ y = \mathbf{w}'^{\top}\mathbf{h} \end{cases}$$

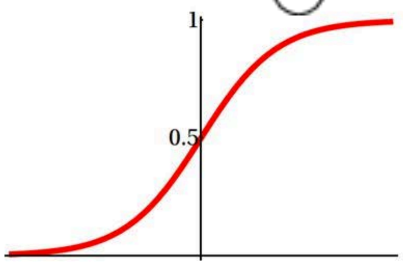
with $\dim(\mathbf{h}) = 3$

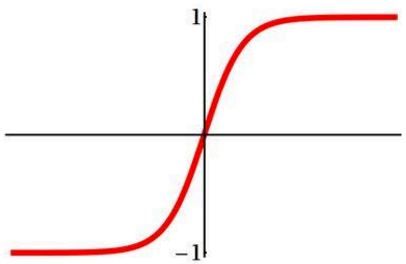


SIGMOID AND TANH



$$
\begin{aligned}
\mathbf{h} &= \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \\
\mathbf{y} &= \sigma(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2)
\end{aligned}
$$

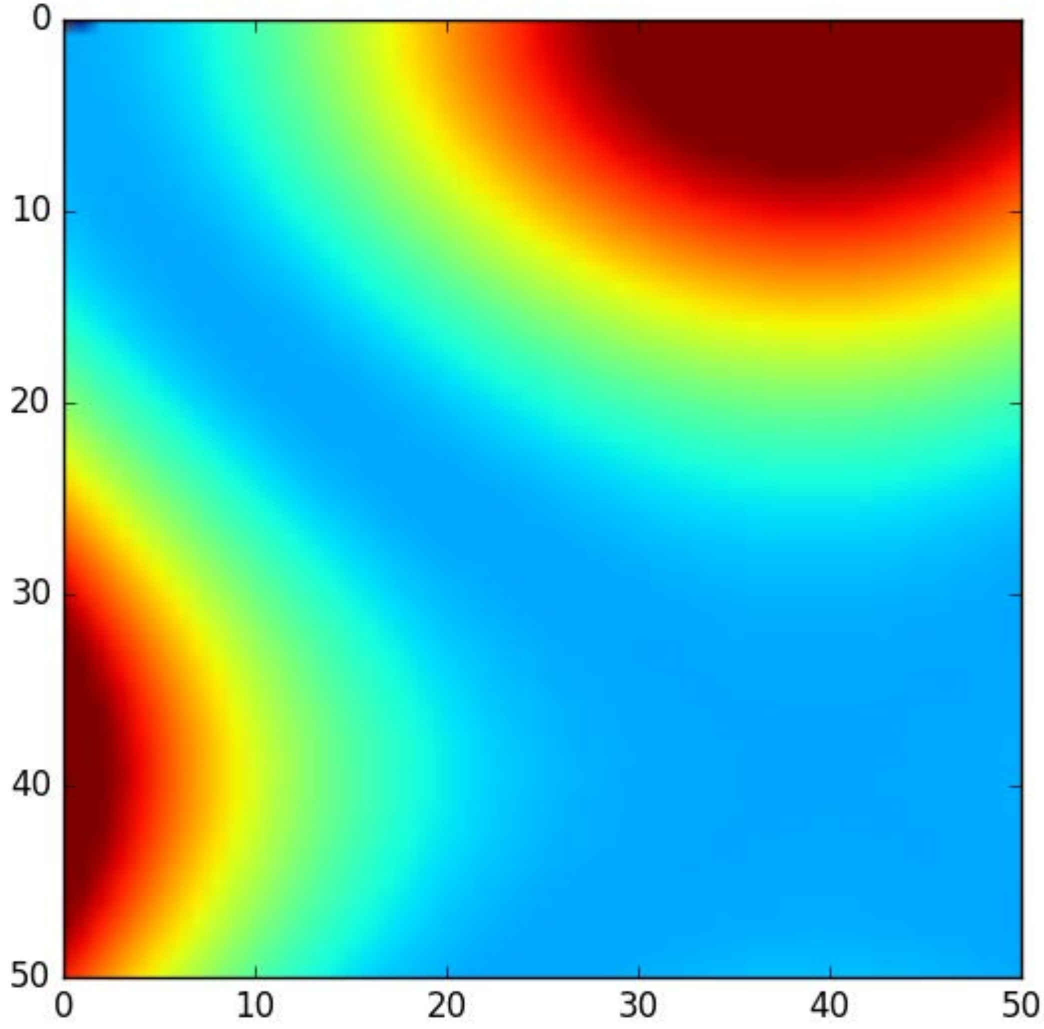




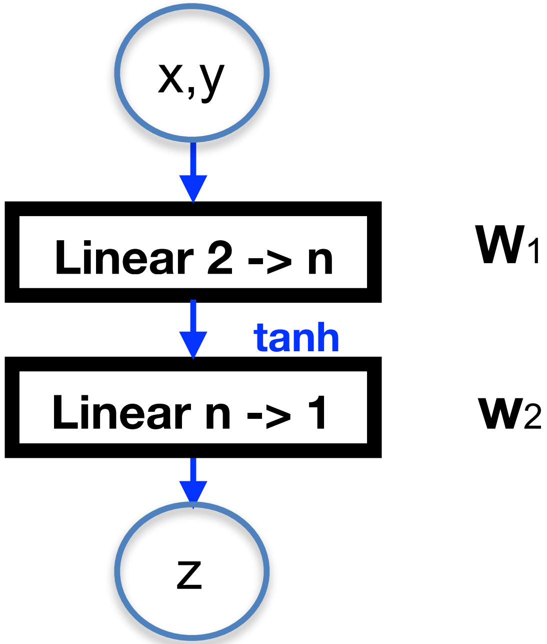
$$
\begin{aligned}
\text{sigm:} \quad \sigma(x) &= \frac{1}{1+\exp(-x)} \\
\text{tanh:} \quad \sigma(x) &= \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}
\end{aligned}
$$

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

INTERPOLATING A SURFACE



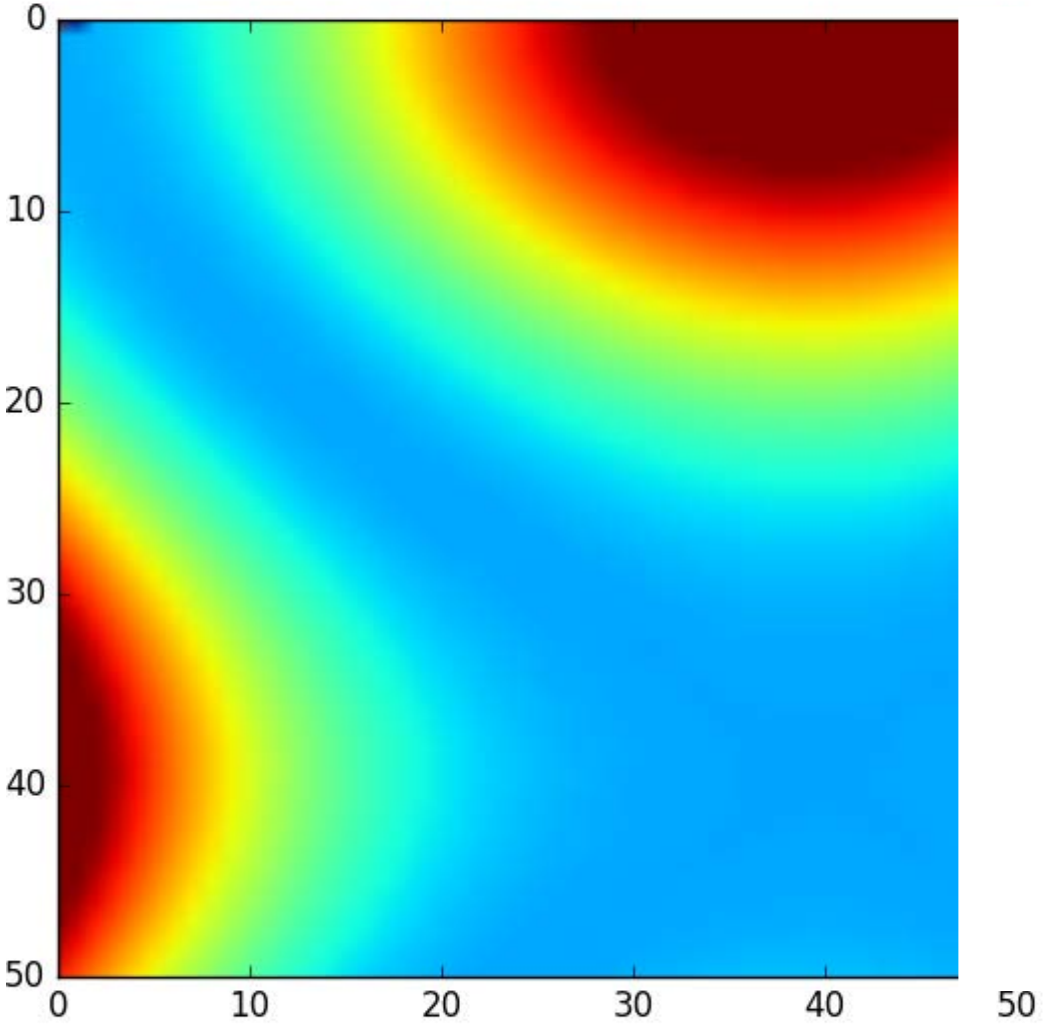
$z = 100 * (y - x^2)^2 + (1 - x)^2$



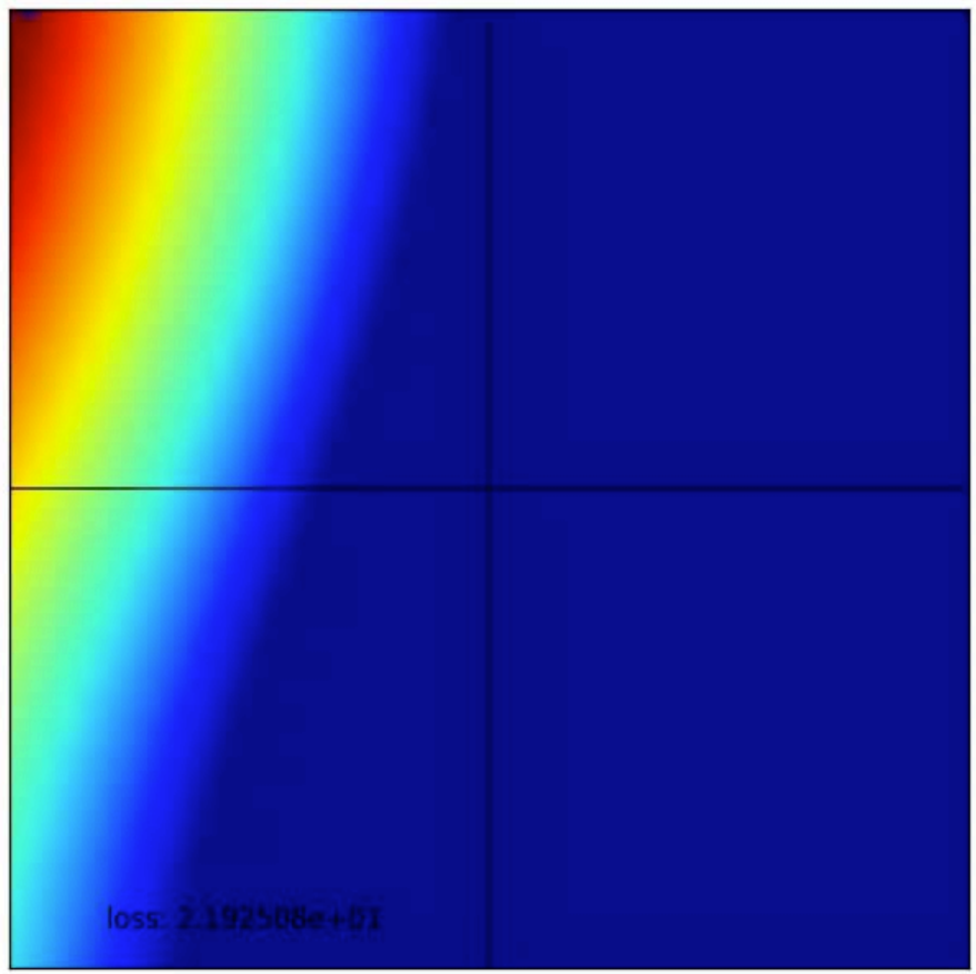
$$
\begin{aligned}
z &= f(x, y) \\
&= \mathbf{w}_2 \sigma(\mathbf{W}_1 \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix}) + b_z
\end{aligned}
$$

Minimize $\sum_i (z_i - f(x_i, y_i))^2$,
with respect to $\mathbf{W}_1, \mathbf{w}_2, b_x, b_y, b_z$.

INTERPOLATING A SURFACE

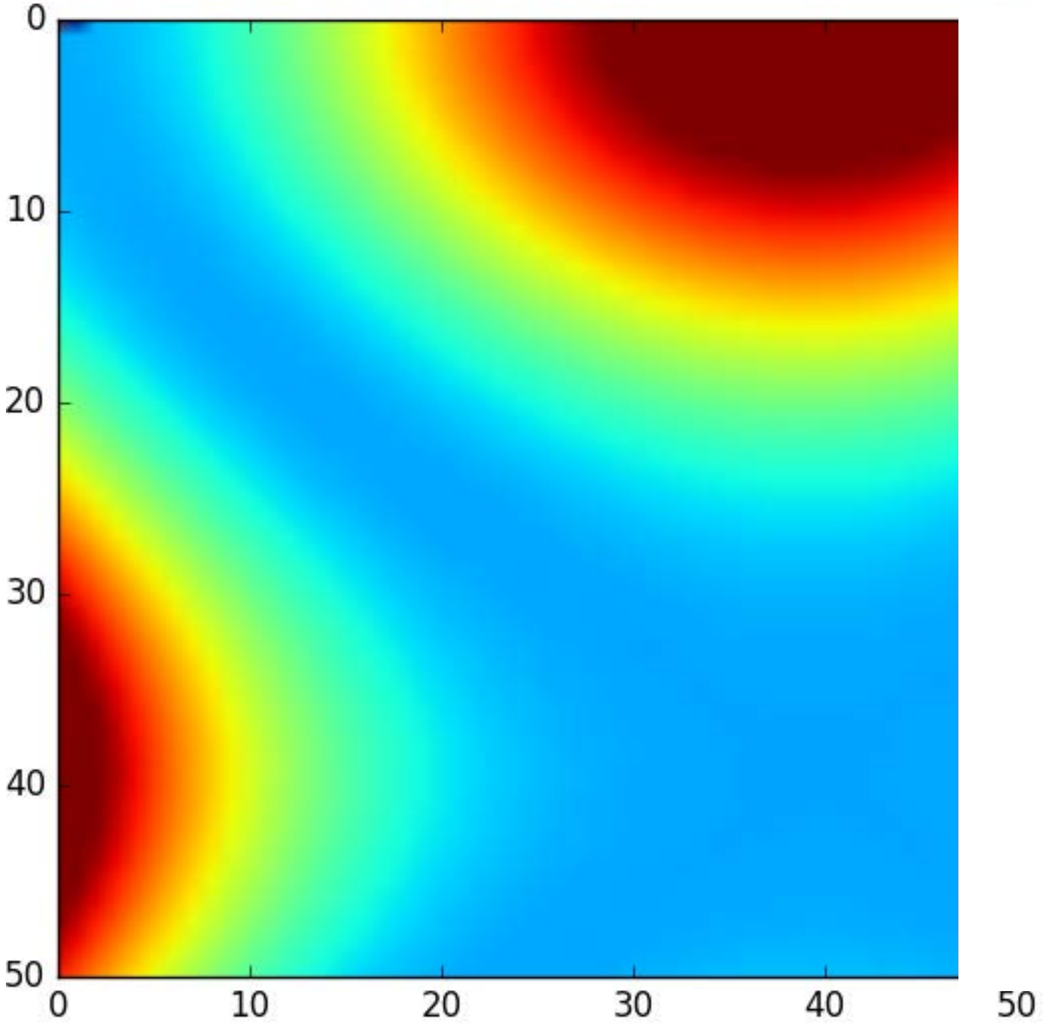


$$z = 100 * (y - x^2)^2 + (1 - x)^2$$

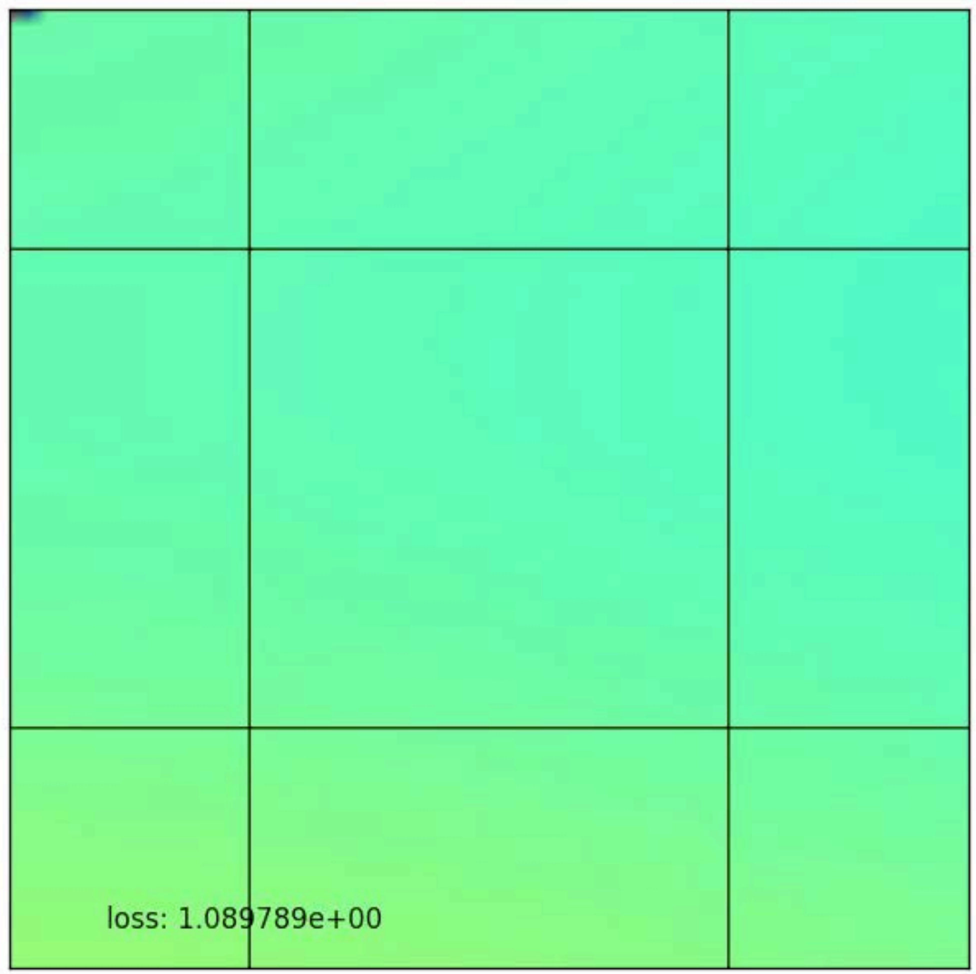


3-node hidden layer

INTERPOLATING A SURFACE

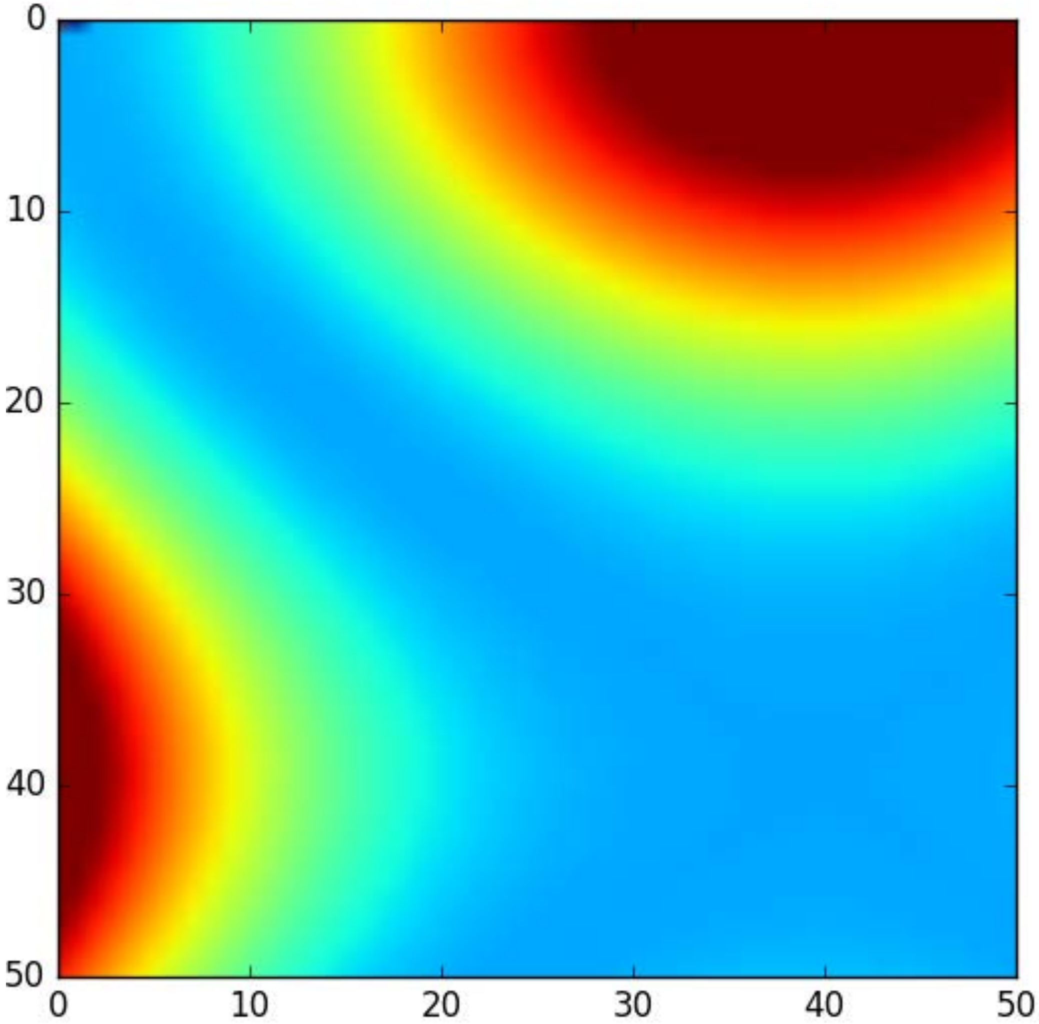


$$z = 100 * (y - x^2)^2 + (1 - x)^2$$

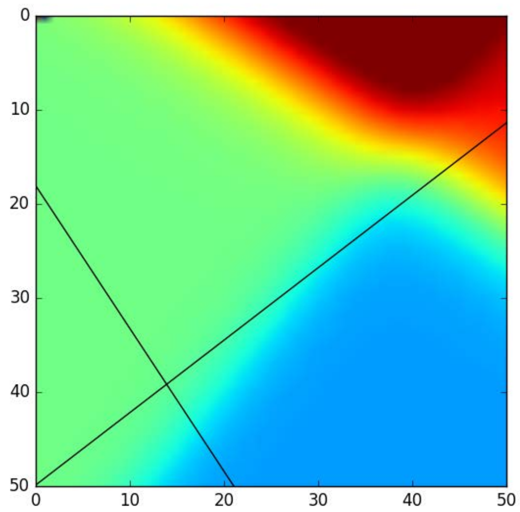


4-node hidden layer

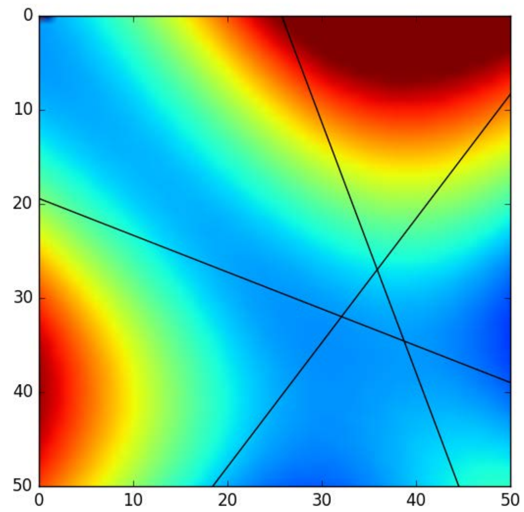
ADDING MORE NODES



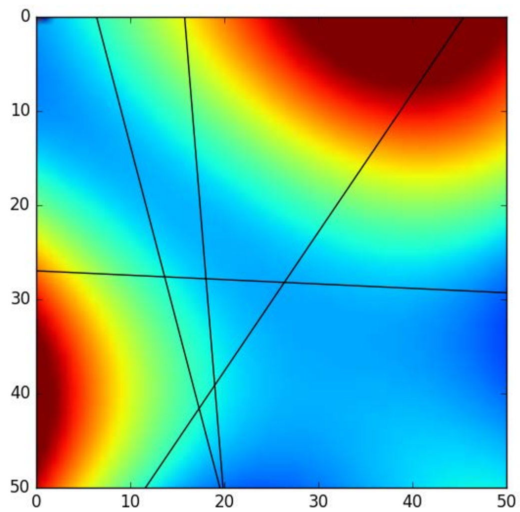
$$z = 100 * (y - x^2)^2 + (1 - x)^2$$



2 nodes -> loss 3.02e-01

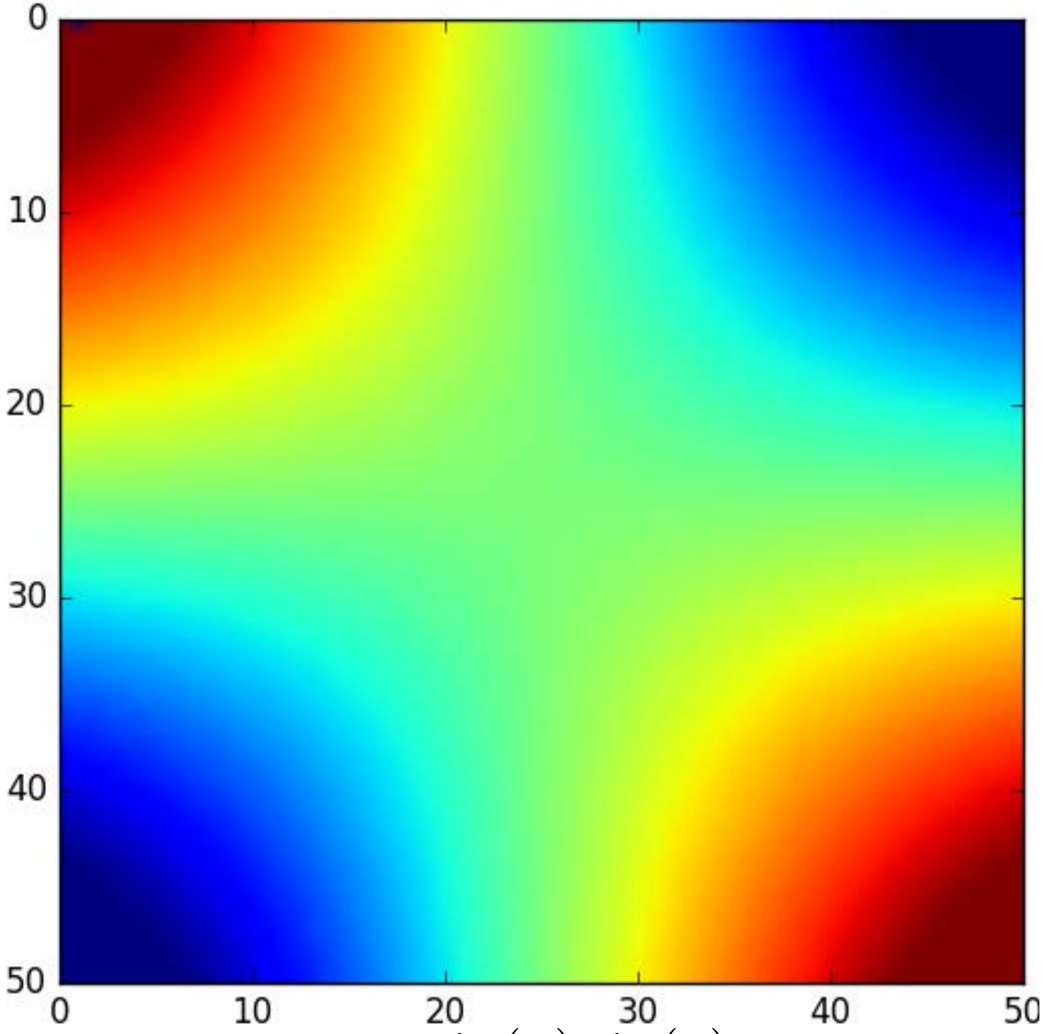


3 nodes -> loss 2.08e-02

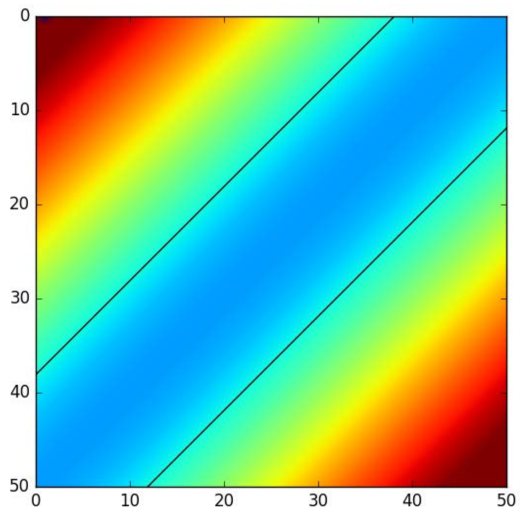


4 nodes -> loss 8.27e-03

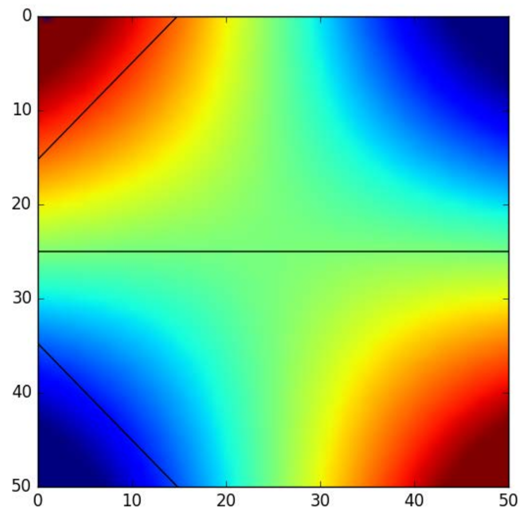
ADDING MORE NODES



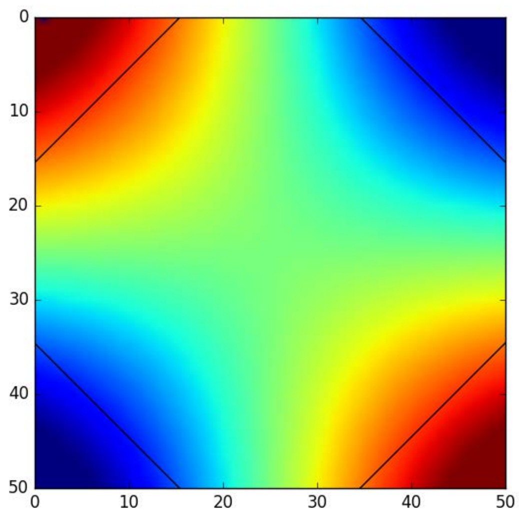
$$z = sin(x)sin(y)$$



2 nodes -> loss 2.61e-01

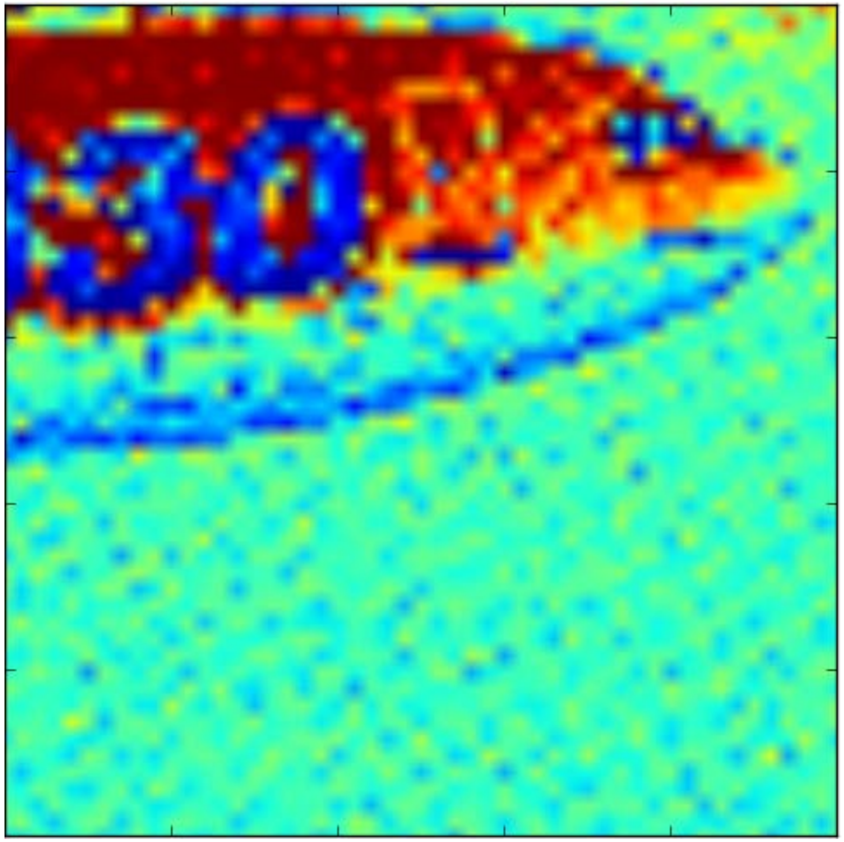


3 nodes -> loss 2.51e-04

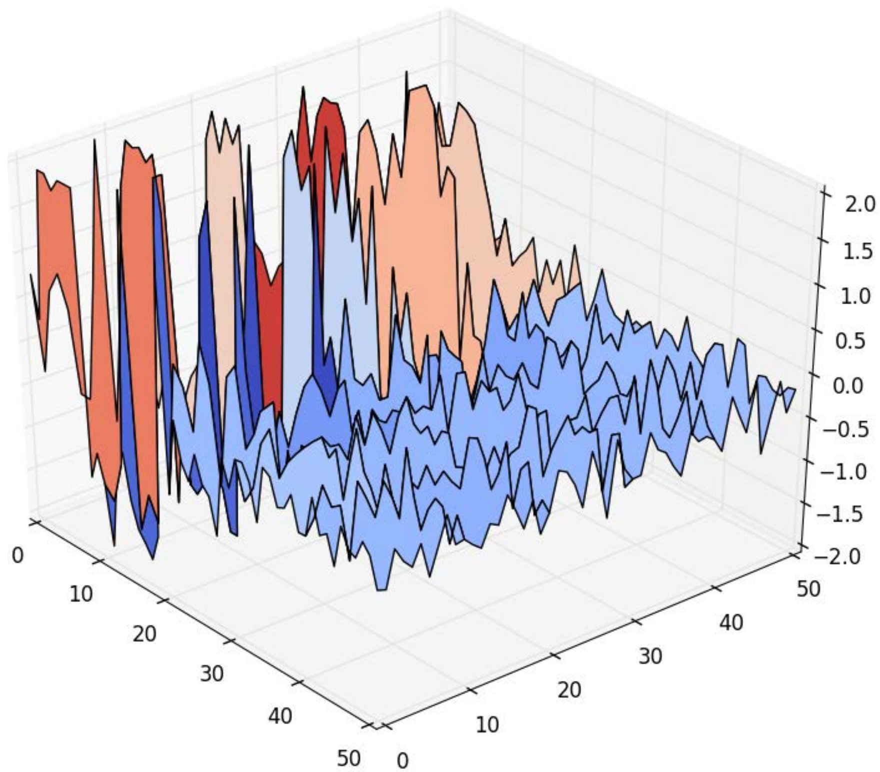


4 nodes -> loss 3.07e-07

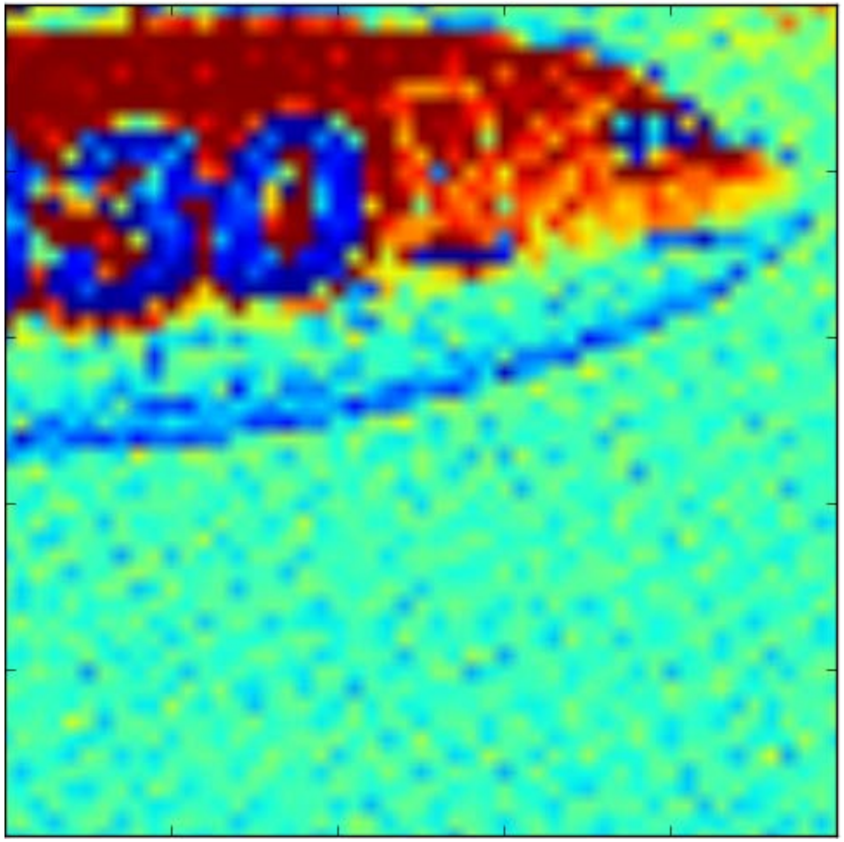
MORE COMPLEX SURFACE



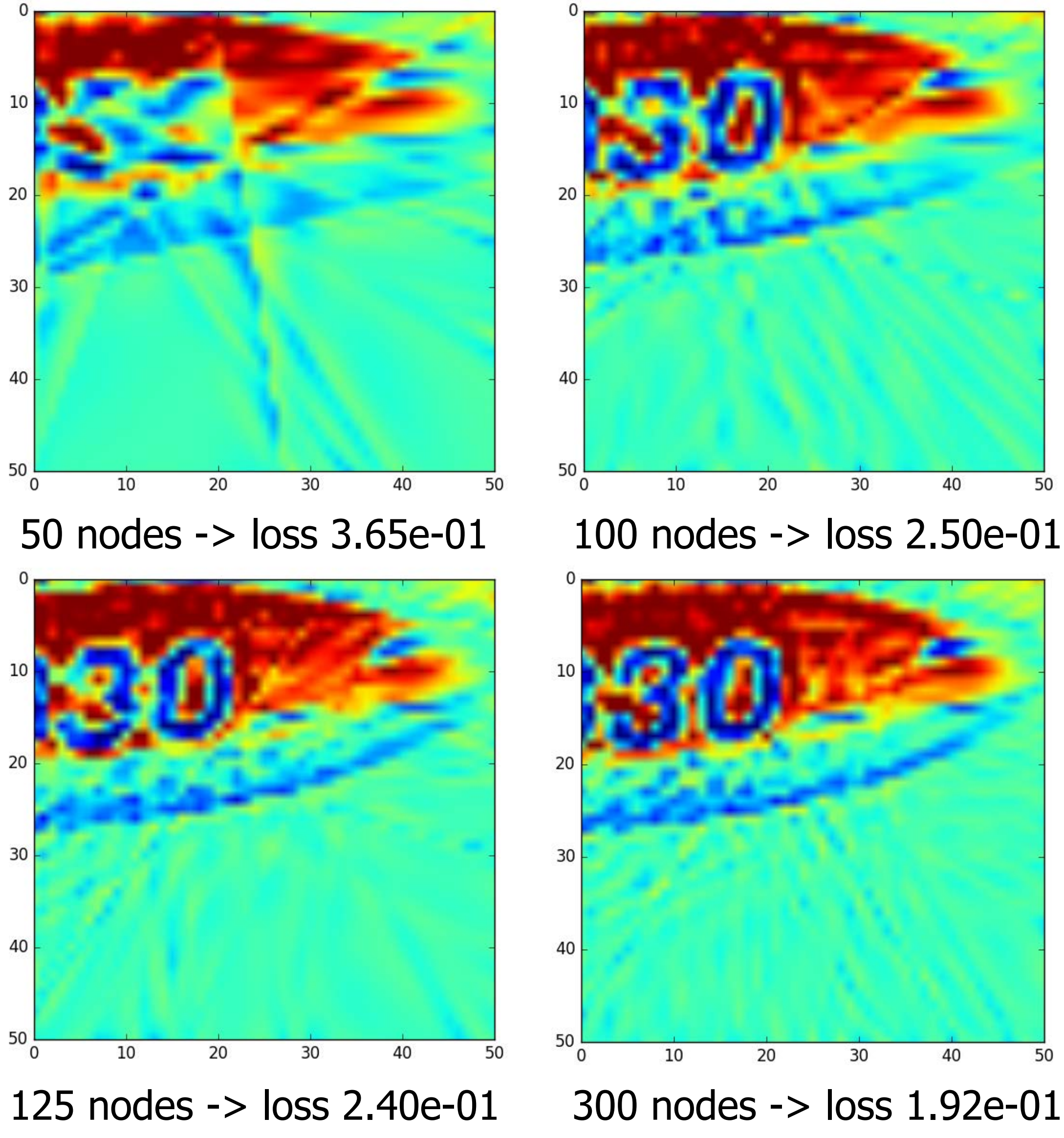
$$I = f(x, y)$$



ADDING MORE NODES



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



UNIVERSAL APPROXIMATION THEOREM

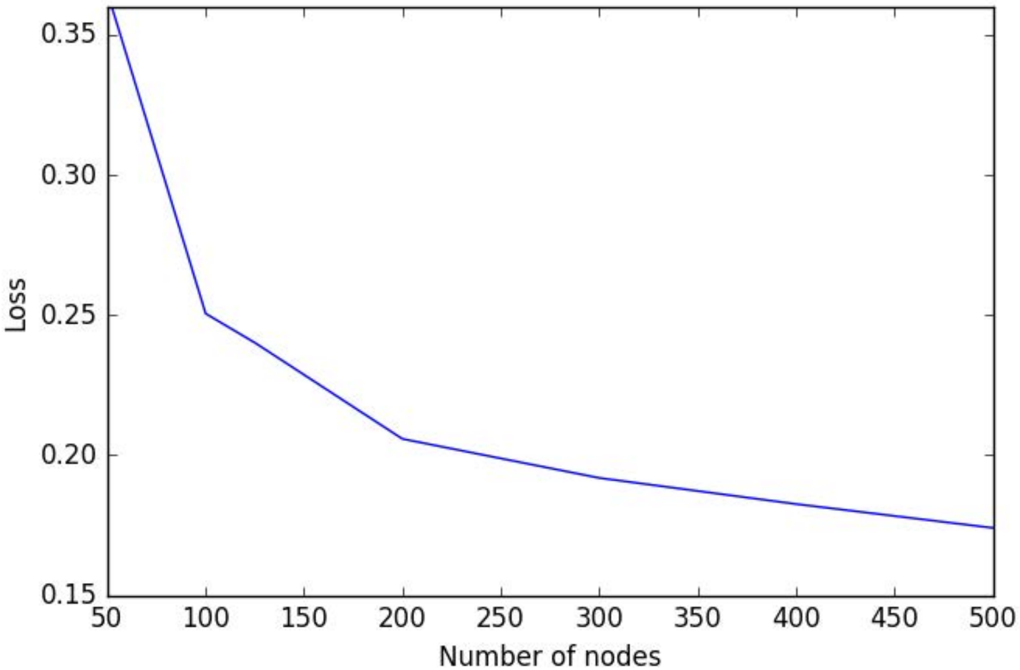
A feedforward network with a linear output layer and at least one hidden layer with any 'squashing' activation function (e.g. logistic sigmoid) can approximate any Borel measurable function (from one finite-dimensional space to another) with any desired nonzero error.

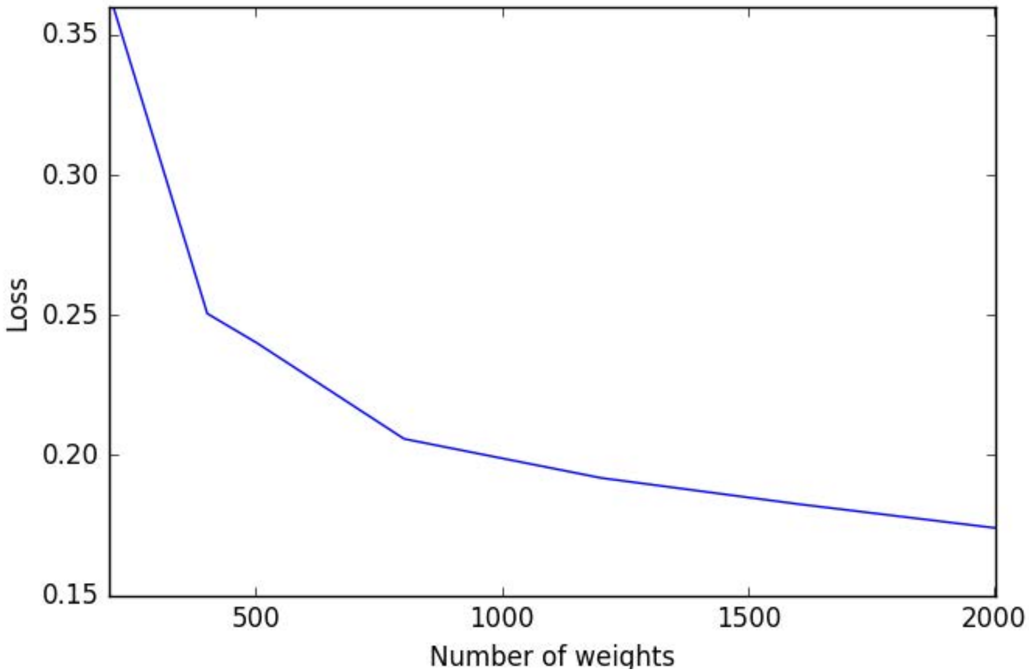
Any continuous function on a closed and bounded set of R^n is Borel-measurable.

—> 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

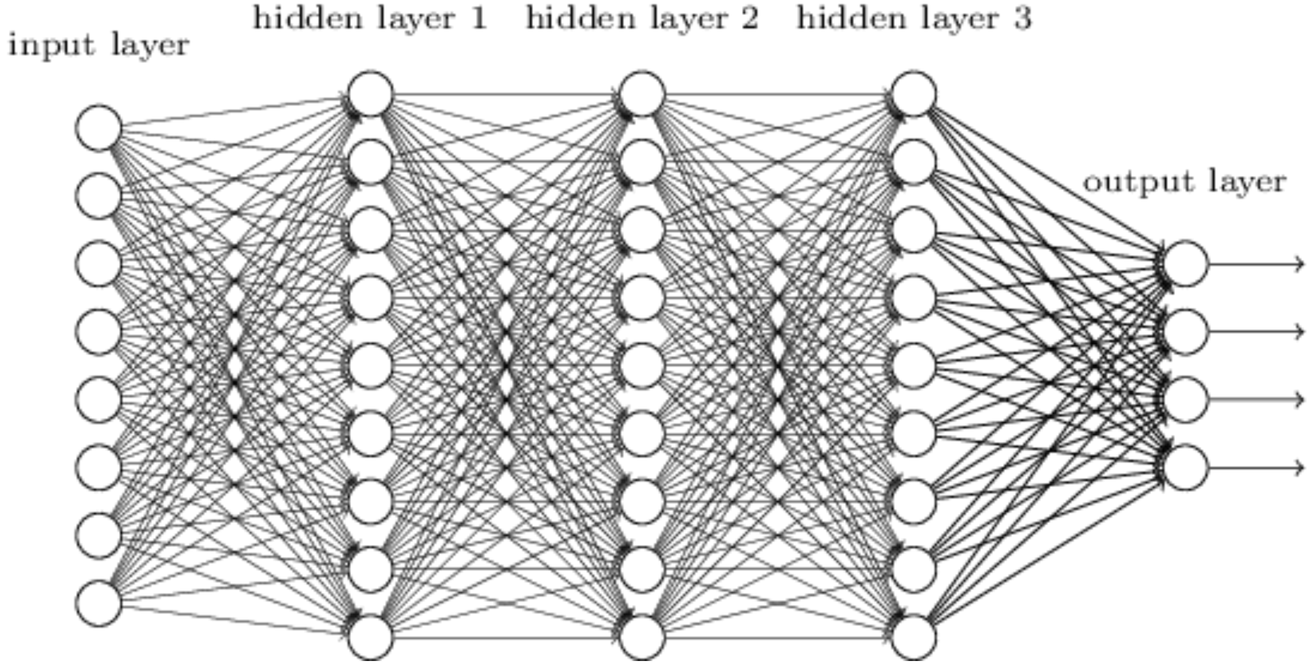




- It may take an exponentially large number of parameters for a good approximation.
- The optimization problem becomes increasingly difficult.

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

DEEP LEARNING



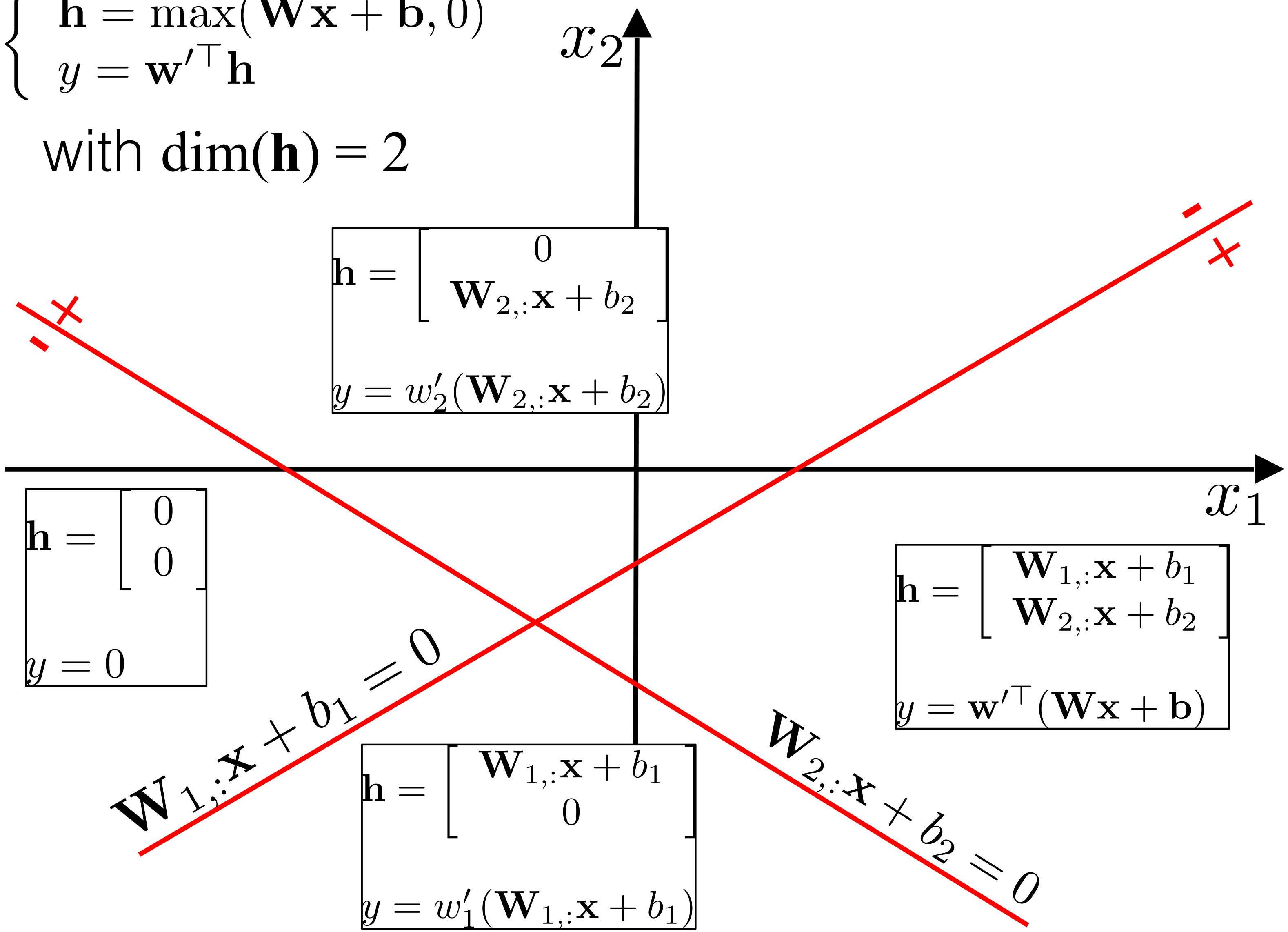
- The descriptive power of the net increases with the number of layers.
- In the case of a 1D signal, it is roughly proportional to $\prod_n W_n$ where W_n represents the width of a layer.

Telgarsky, JMLR'16

TWO HYPERPLANES
ONE SINGLE LAYER

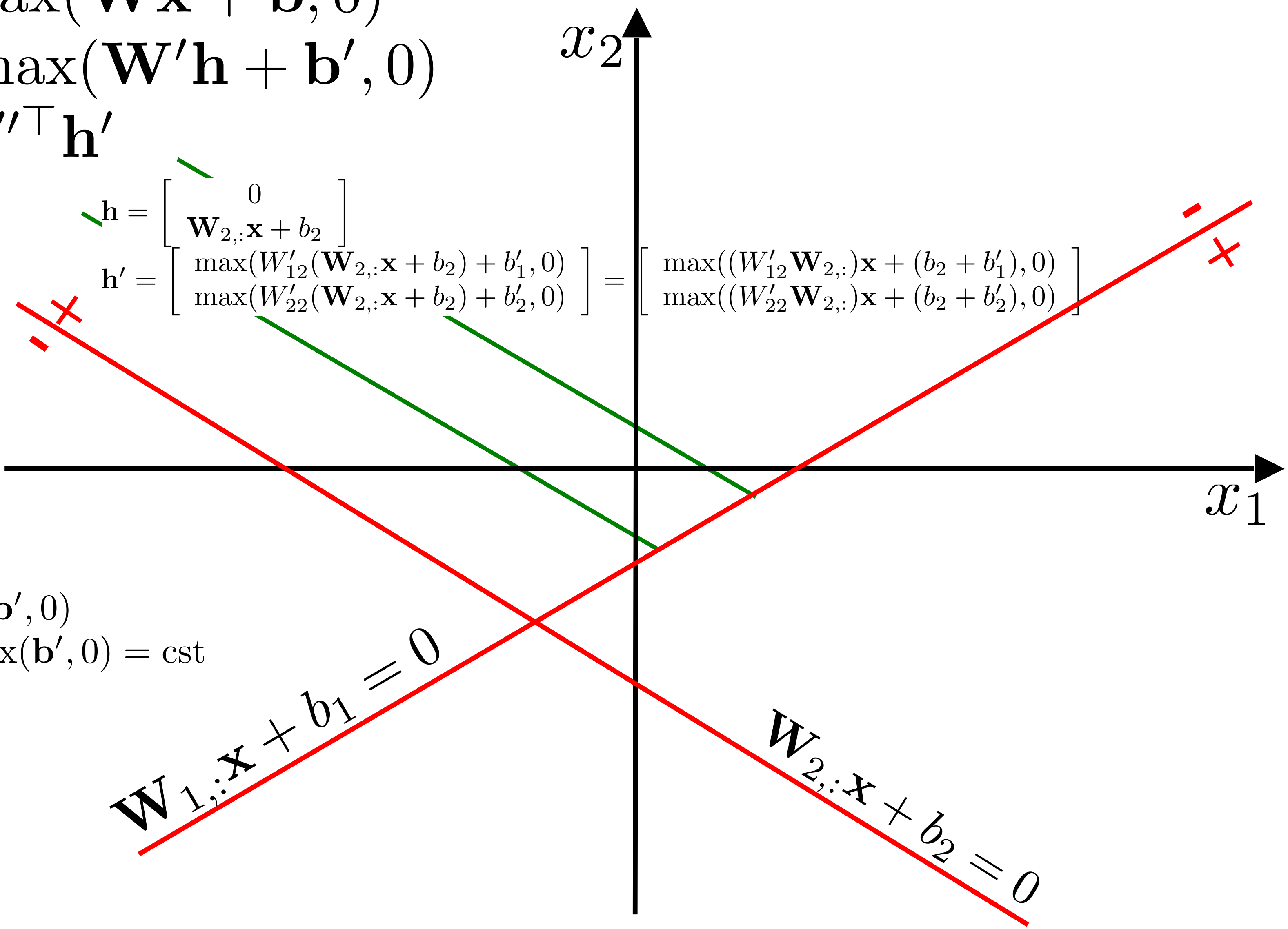
$$\begin{cases} \mathbf{h} = \max(\mathbf{W}\mathbf{x} + \mathbf{b}, 0) \\ y = \mathbf{w}'^{\top}\mathbf{h} \end{cases}$$

with $\dim(\mathbf{h}) = 2$



TWO HYPERPLANES
TWO LAYERS

$$
\begin{cases}
\mathbf{h} = \max(\mathbf{W}\mathbf{x} + \mathbf{b}, 0) \\
\mathbf{h}' = \max(\mathbf{W}'\mathbf{h} + \mathbf{b}', 0) \\
y = \mathbf{w}''^{\top}\mathbf{h}'
\end{cases}
$$



$$\mathbf{h} = \begin{bmatrix} 0 \\ \mathbf{W}_{2,:}\mathbf{x} + b_2 \end{bmatrix}$$

$$\mathbf{h}' = \begin{bmatrix} \max(W'_{12}(\mathbf{W}_{2,:}\mathbf{x} + b_2) + b'_1, 0) \\ \max(W'_{22}(\mathbf{W}_{2,:}\mathbf{x} + b_2) + b'_2, 0) \end{bmatrix} = \begin{bmatrix} \max((W'_{12}\mathbf{W}_{2,:})\mathbf{x} + (b_2 + b'_1), 0) \\ \max((W'_{22}\mathbf{W}_{2,:})\mathbf{x} + (b_2 + b'_2), 0) \end{bmatrix}$$

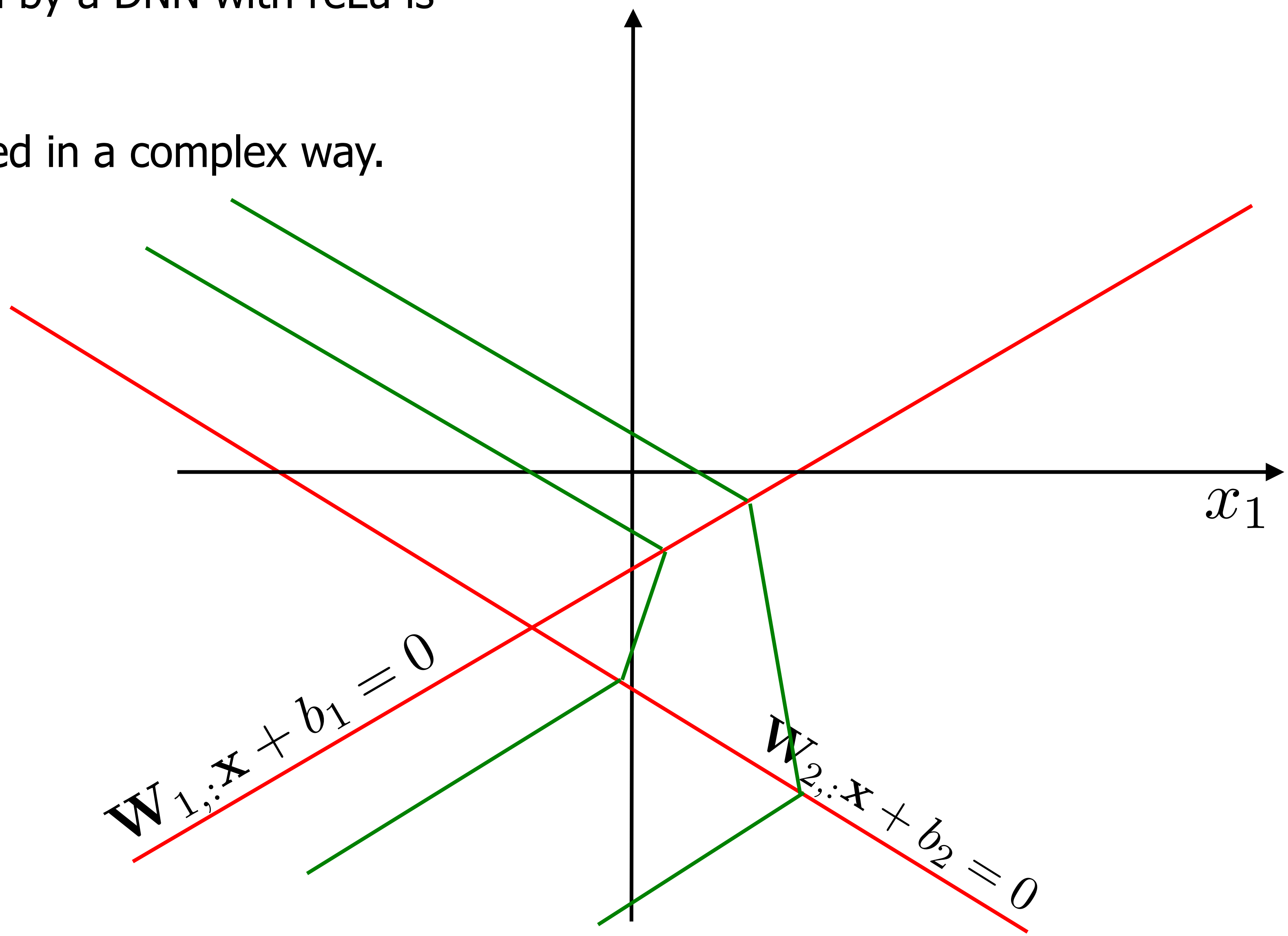
$$\mathbf{h} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{h}' = \max(\mathbf{b}', 0)$$

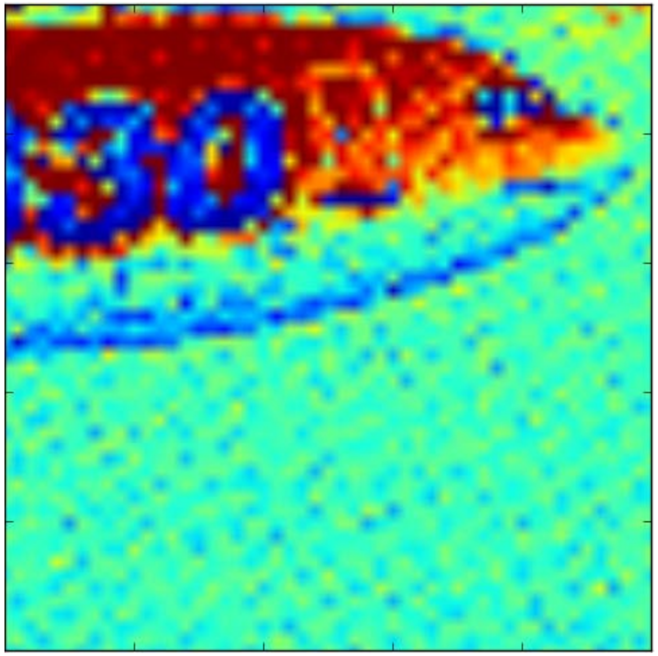
$$y = \mathbf{w}'' \max(\mathbf{b}', 0) = \text{cst}$$

BACK TO HYPERPLANES

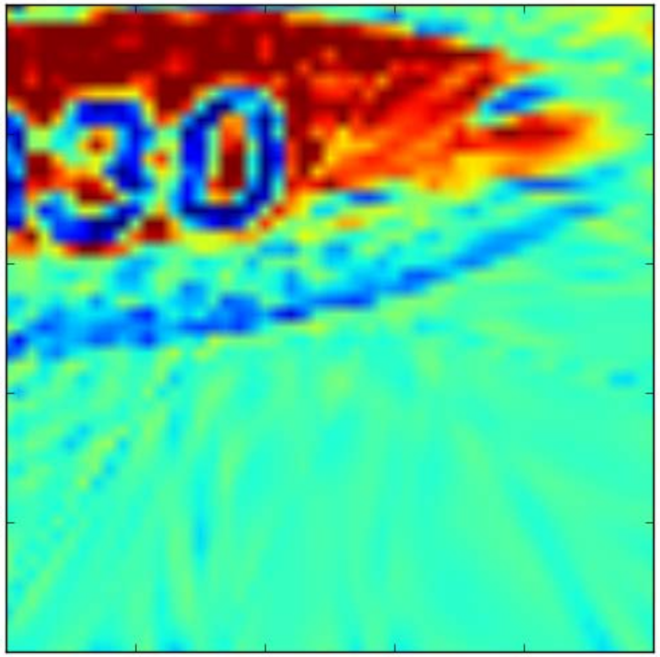
The function learned by a DNN with reLu is
- piecewise affine;
- continuous;
- with regions related in a complex way.



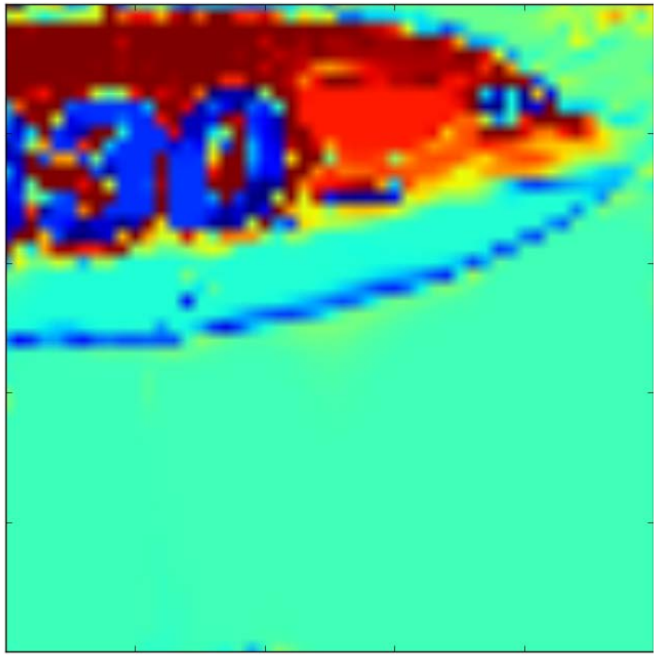
ADDING A SECOND LAYER



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



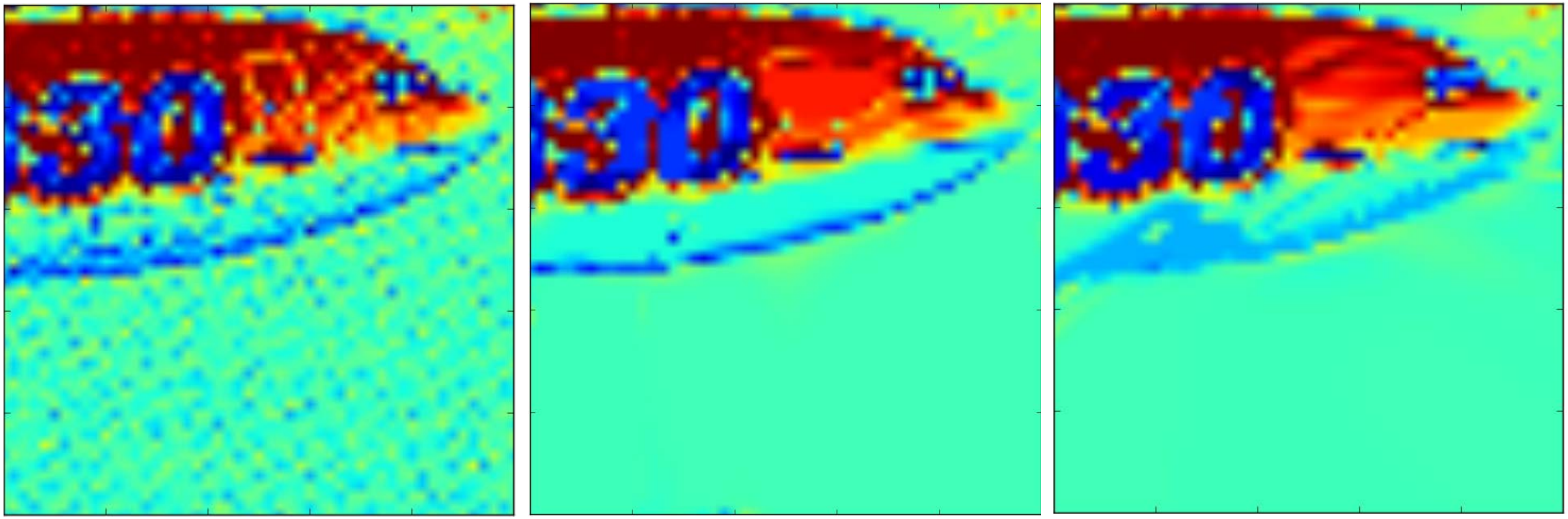
1 Layer: 125 nodes -> loss 2.40e-01



2 Layers: 20 nodes -> loss 8.31e-02

501 weights in both cases

ADDING A THIRD LAYER



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

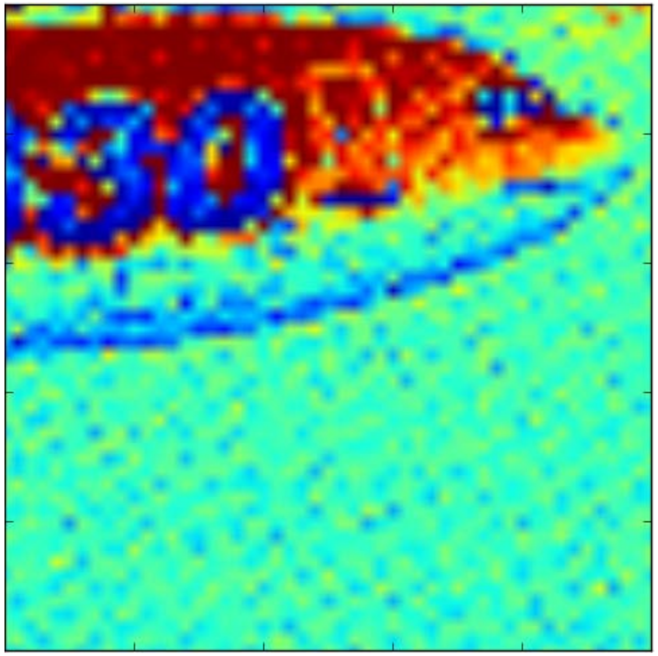
2 Layers: 20 nodes -> loss 8.31e-02

501 weights

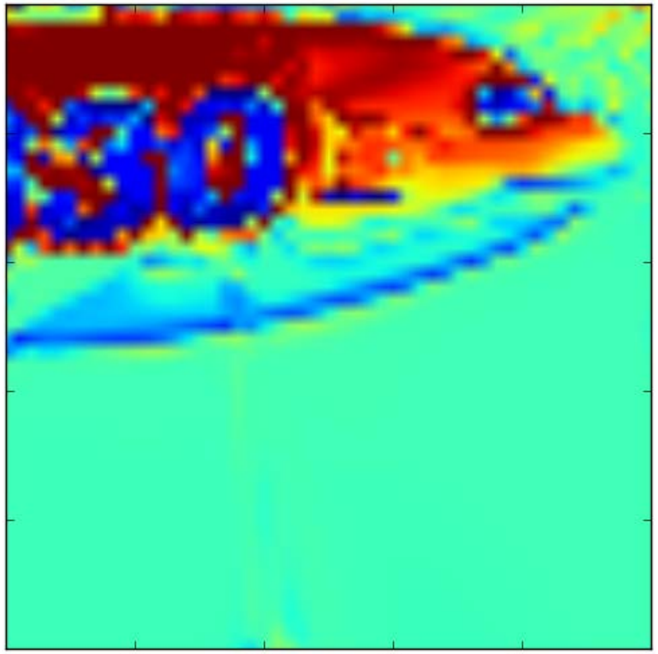
3 Layers: 14 nodes -> loss 7.55e-02

477 weights

ADDING A THIRD LAYER

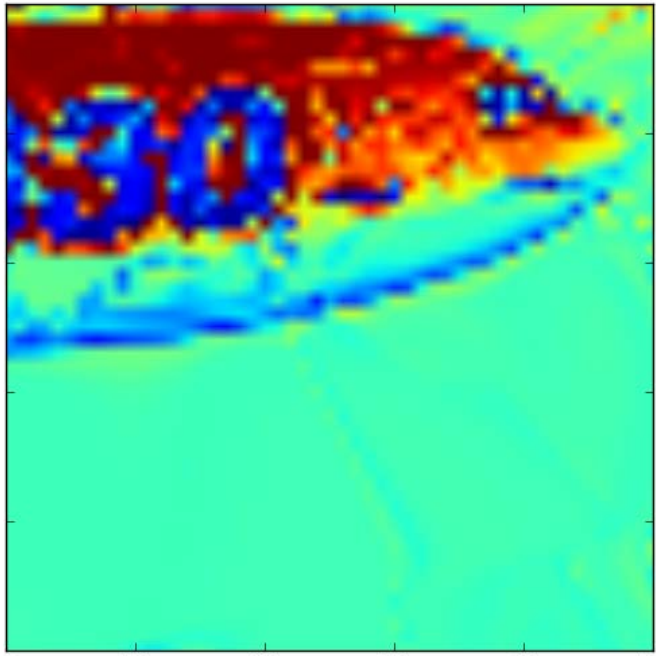


$$I = f(x, y)$$



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

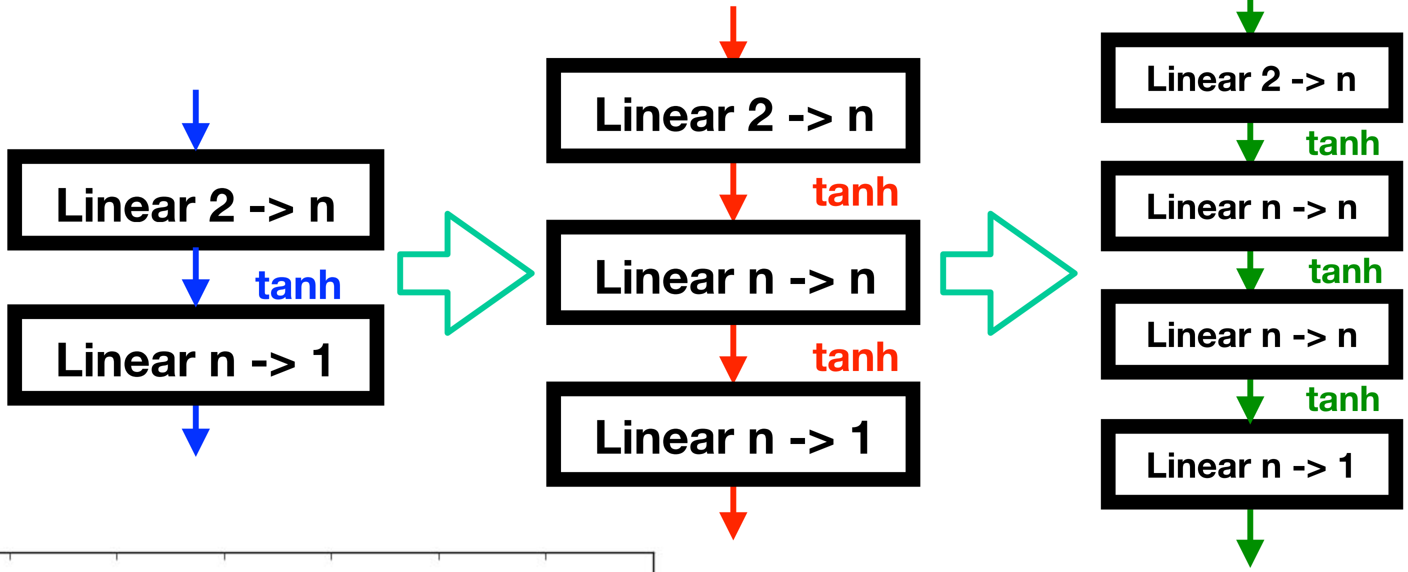
541 weights

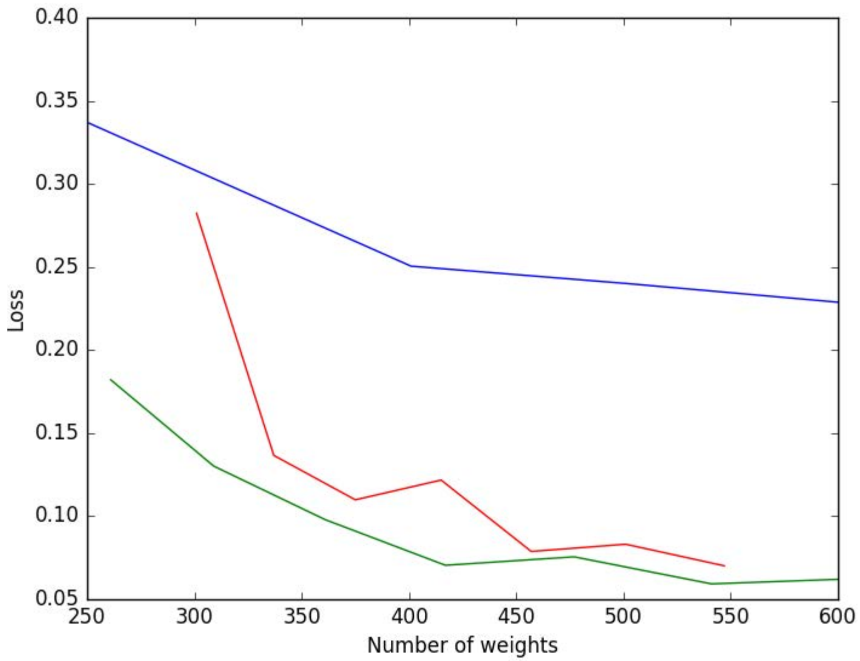


3 Layers: 19 nodes -> loss 4.38e-02

837 weights

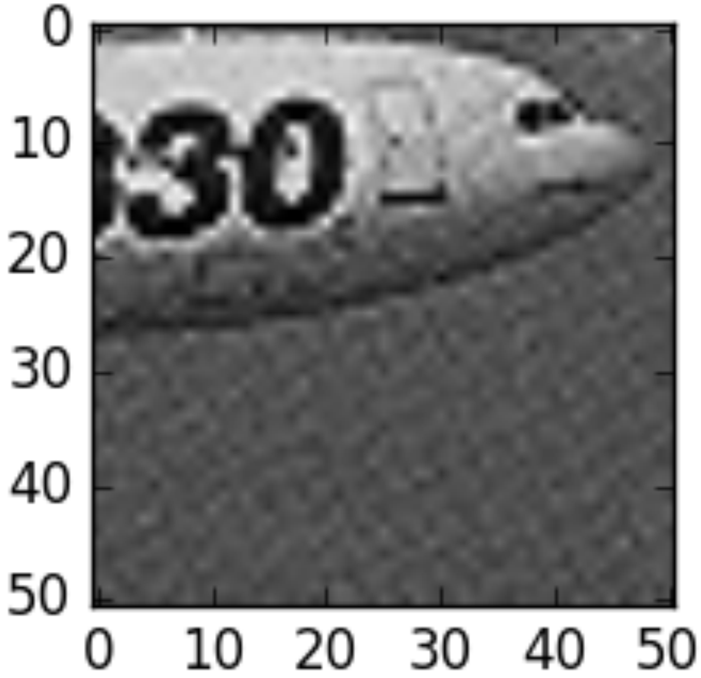
MULTILAYER PERCEPTRONS



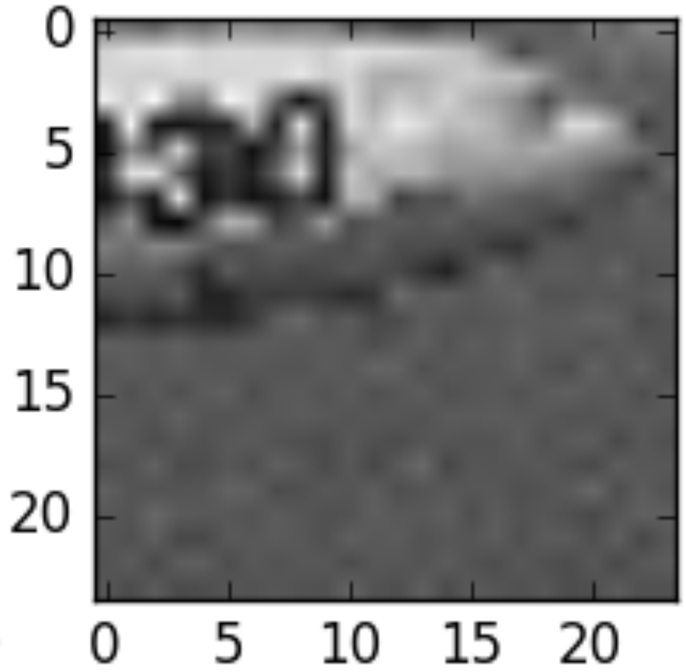


- Adding layers often yields better convergence properties.
- In current practice, deeper is usually better.

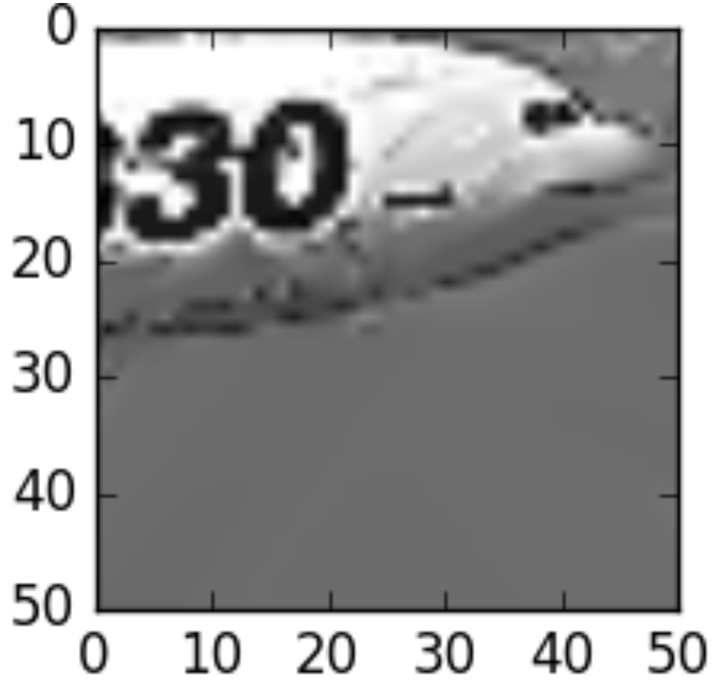
SIMPLER WAY TO INTERPOLATE



Original 51x51 image:
2601 gray level values.



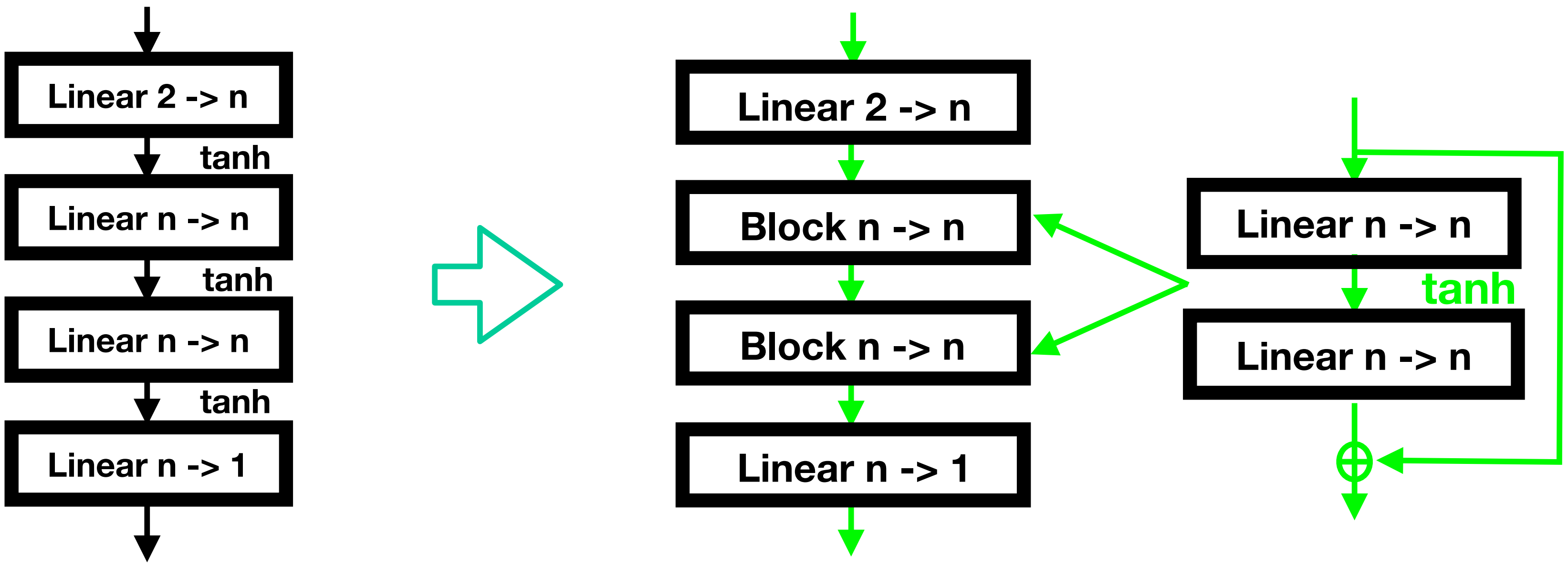
Scaled 24x24 image:
576 gray level values.



MLP 10/20/10 Interpolation:
471 weights.

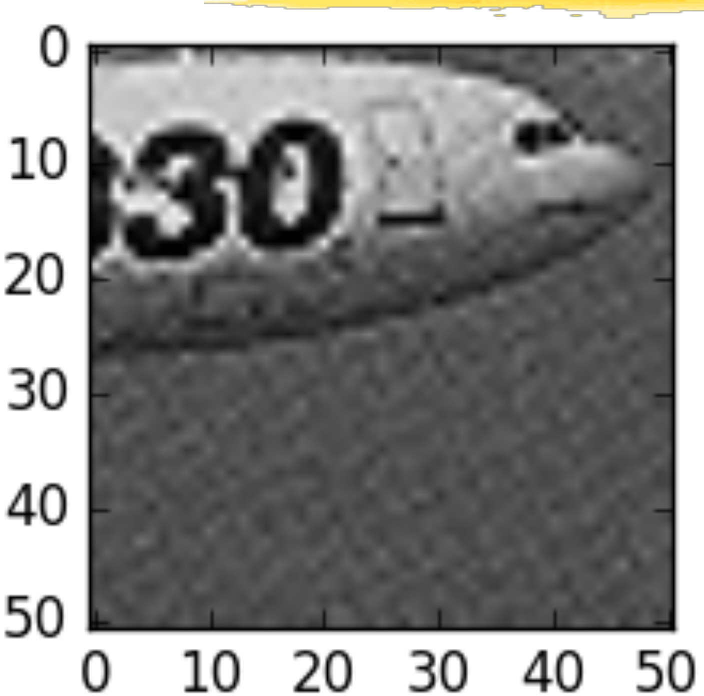
Simpler but not necessarily better!

MLP TO RESNET

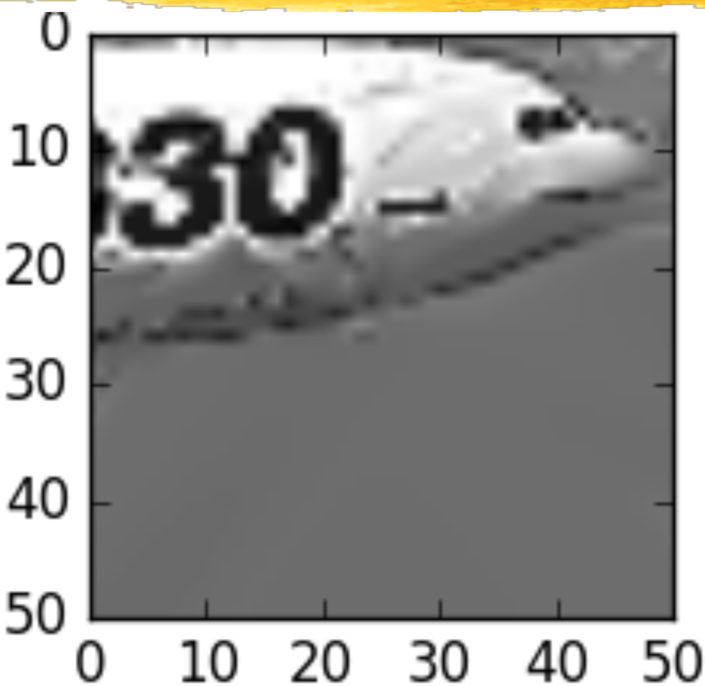


Further improvements in the convergence properties have been obtained by adding a bypass, which allows the final layers to only compute residuals.

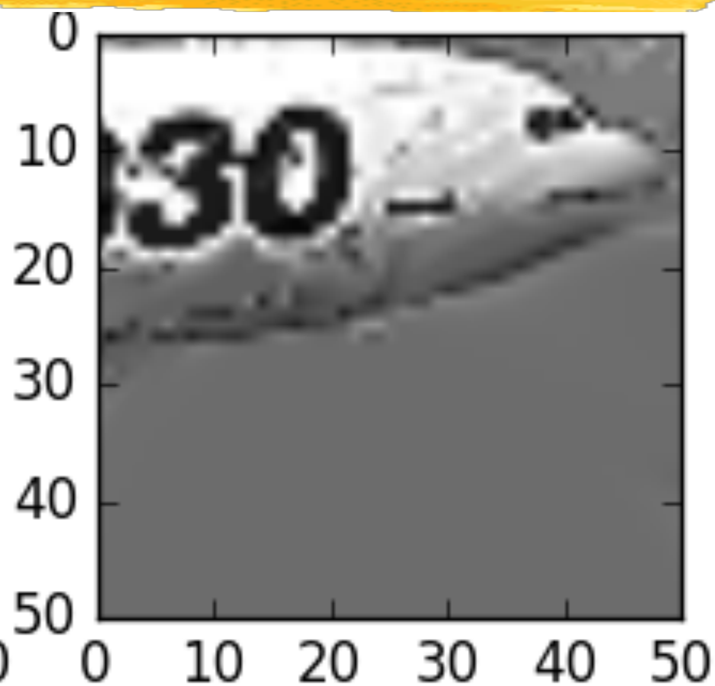
IMPROVING THE NETWORK



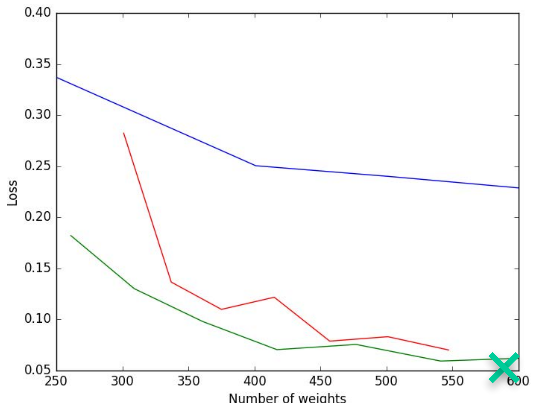
Original 51x51 image:
2601 gray level values.

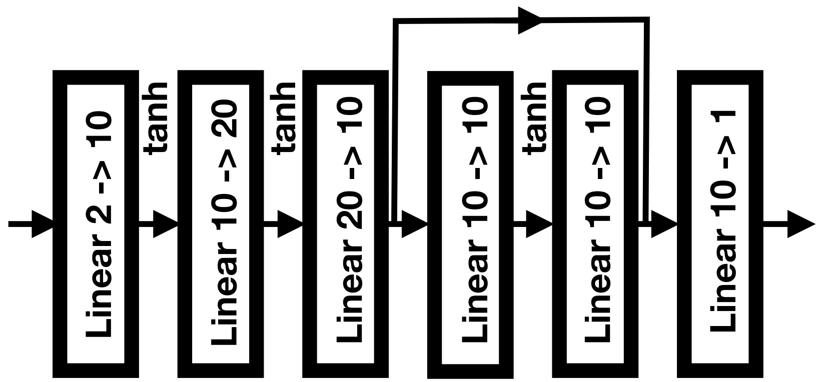


MLP 10/20/10 Interpolation:
471 weights, loss 6.43e-02.

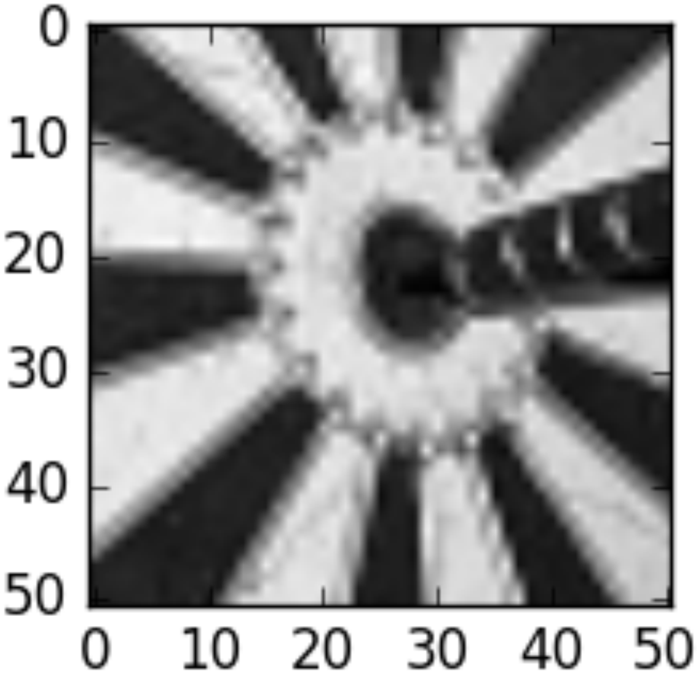


MLP 10/20/10/10 Interpolation:
581 weights, loss 5.30e-2.

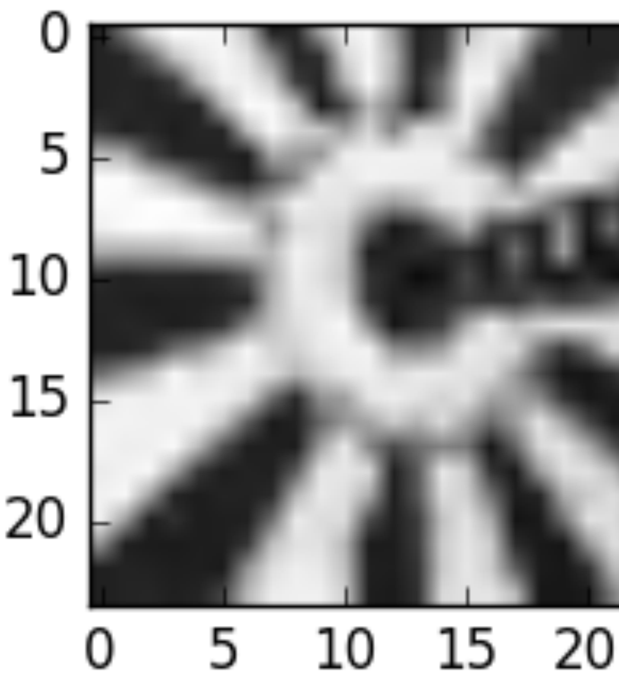




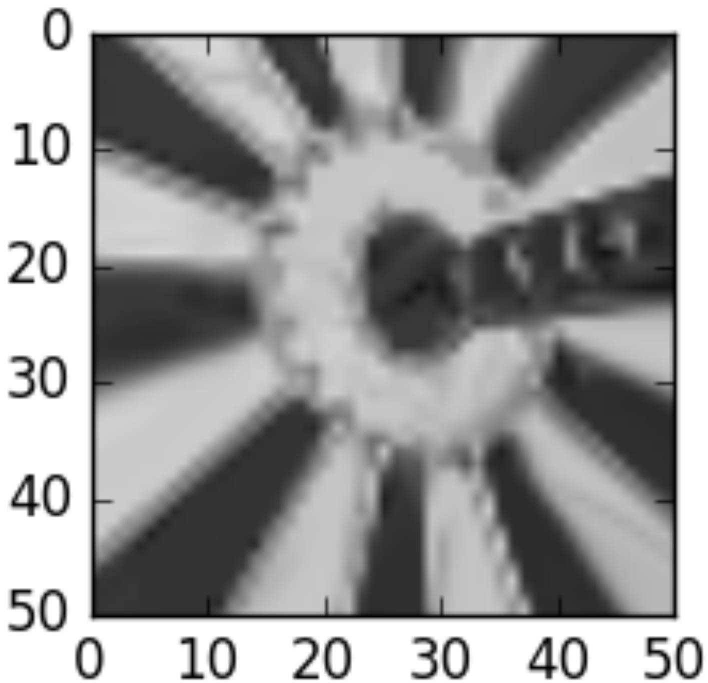
IMPROVING THE NETWORK



Original 51x51 image:
2601 gray level values.



MLP 10/20/10 Interpolation:
471 weights, loss 1.95e-02.

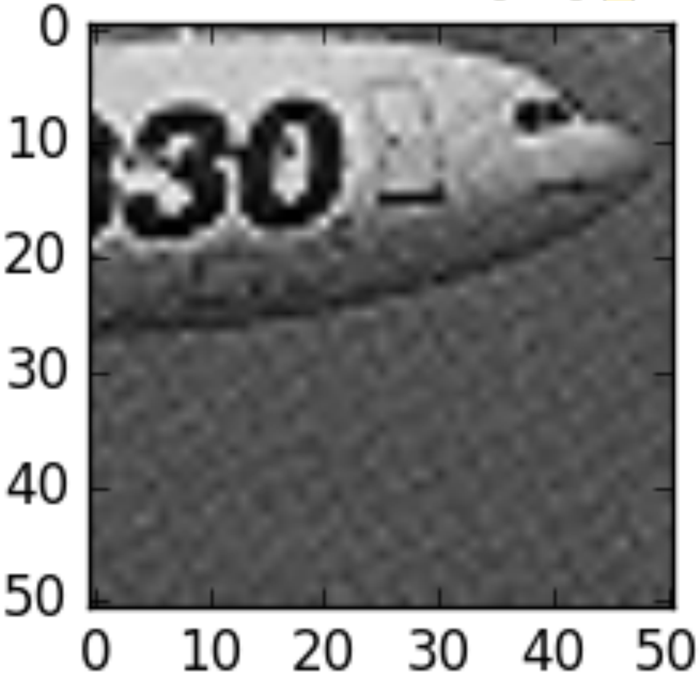


MLP 10/20/10/10 Interpolation:
581 weights, loss 1.36e-2.

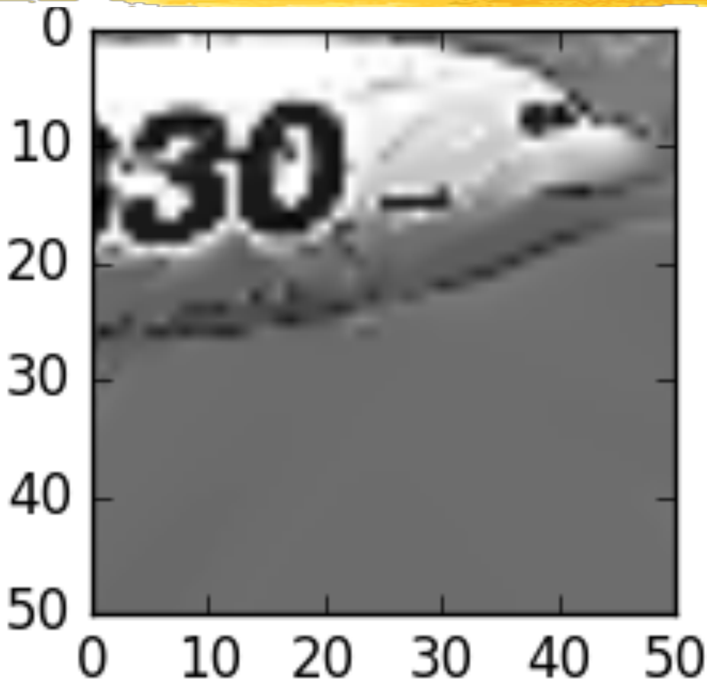
- Relatively small improvement in this case.
- The problem is probably too small.

—> Networks can behave very differently for small and large problems!

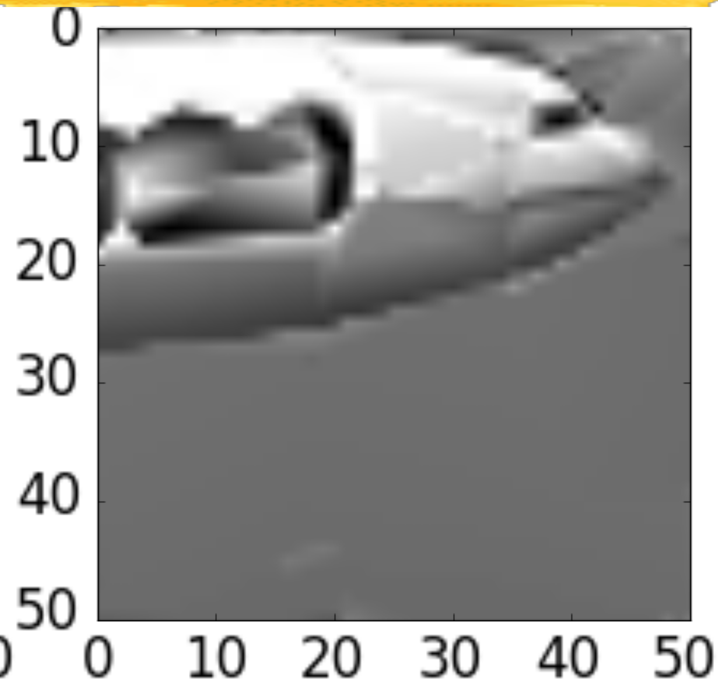
TANH vs ReLU



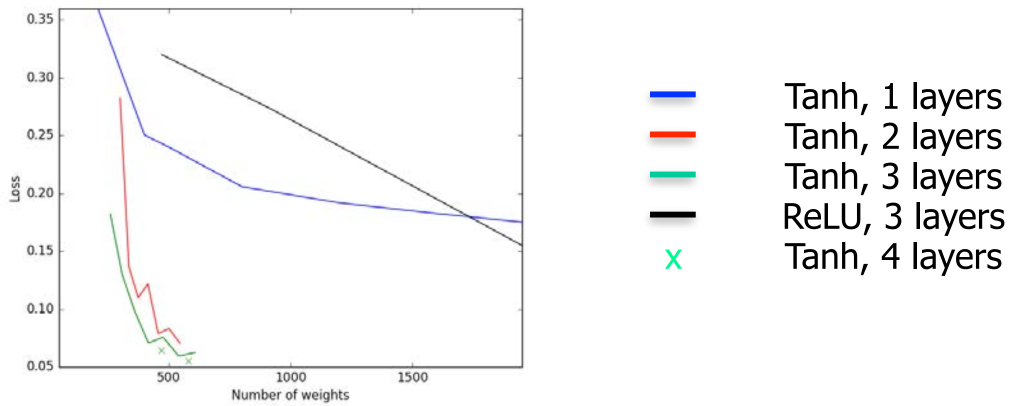
Original 51x51 image:
2601 gray level values.



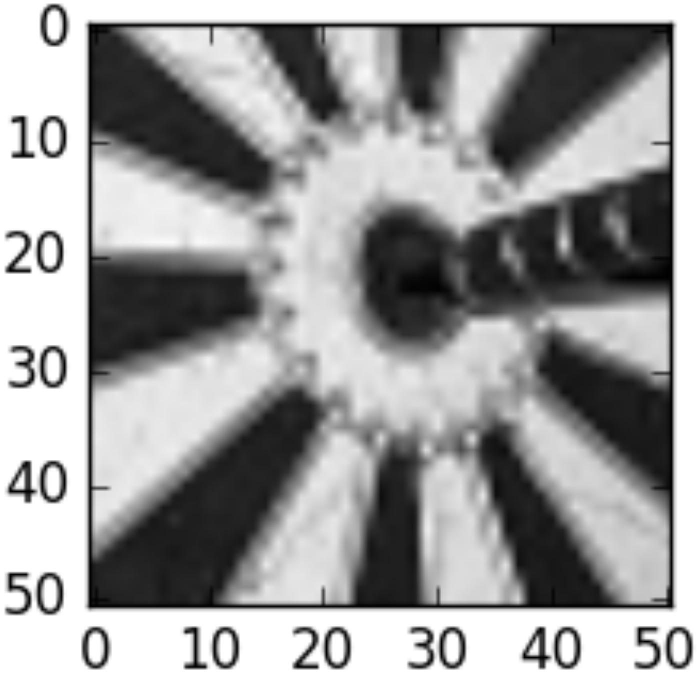
MLP 10/20/10 Interpolation:
Tanh, loss 6.43e-02.



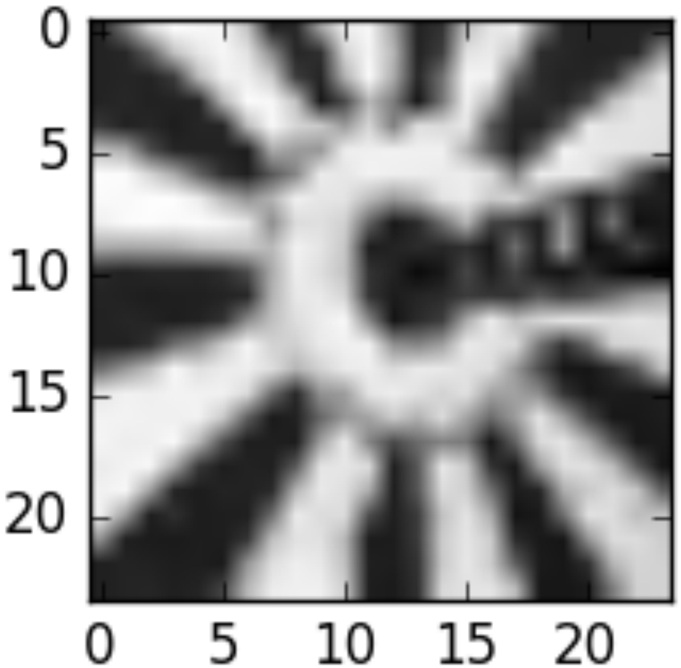
MLP 10/20/10 Interpolation:
ReLU, loss 3.07e-1.



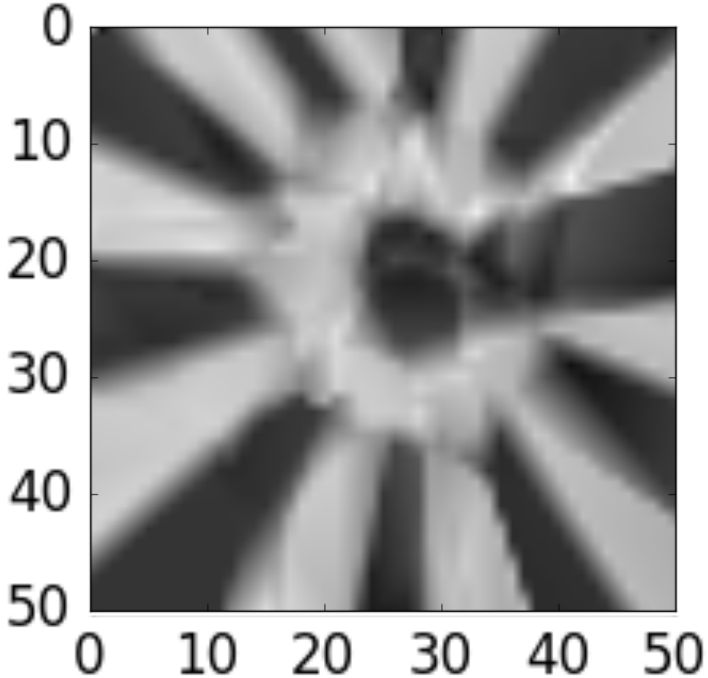
TANH vs ReLU



Original 51x51 image:
2601 gray level values.



MLP 10/20/10 Interpolation:
tanh, loss 1.95e-02.



MLP 10/20/10 Interpolation:
relu, loss 7.21e-2.

- 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!

MULTILAYER PERCEPTRONS

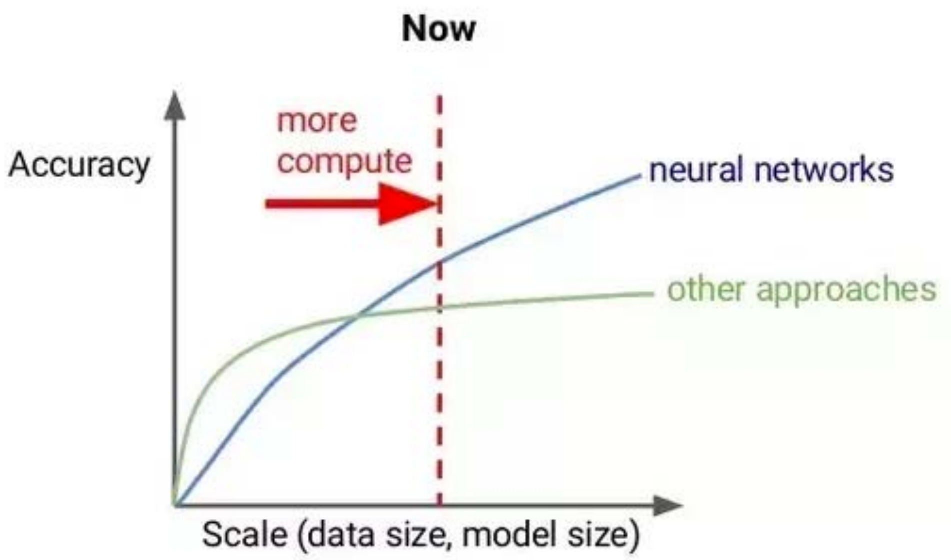
The function learned by a DNN using either the ReLU or Tanh operators is:

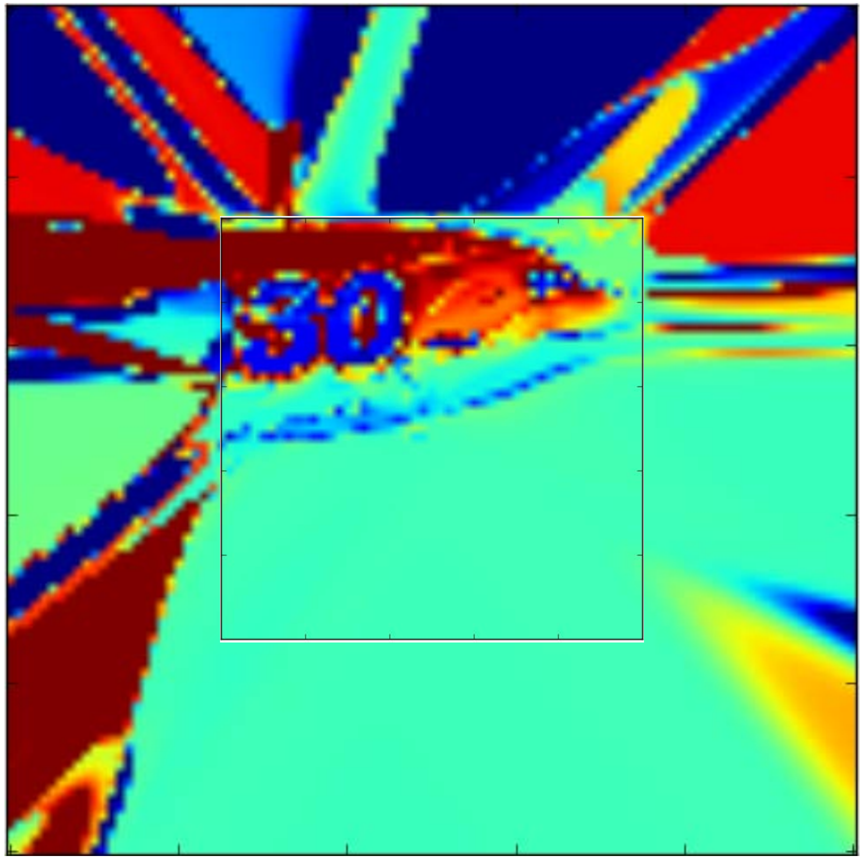
- piecewise affine or smooth;
- continuous because it is a composition of continuous functions.

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





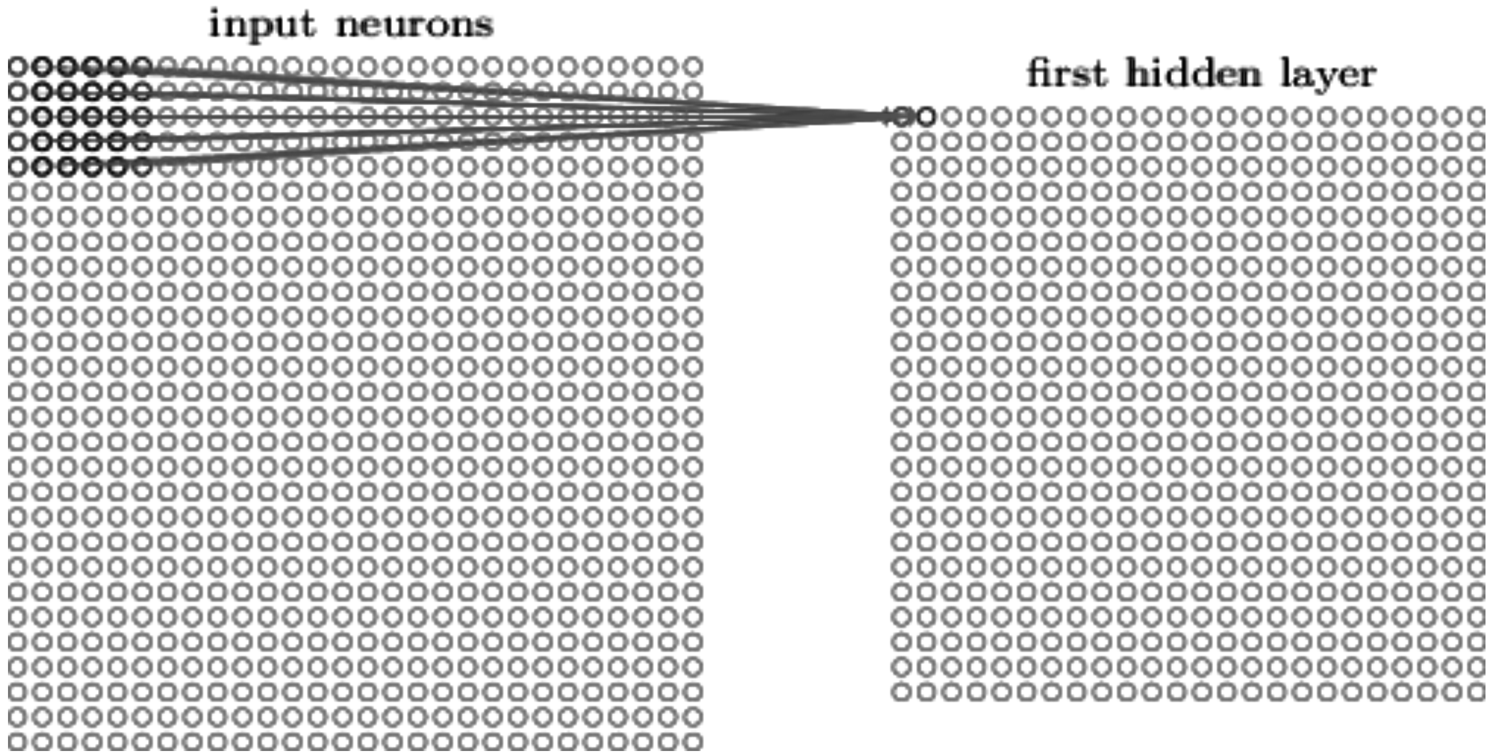
- Powerful regressors but require many parameters, and therefore large training databases.
- Excellent at interpolation but less good at extrapolation. The training data must cover all cases of interest.

IMAGE SPECIFICITIES

- 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.

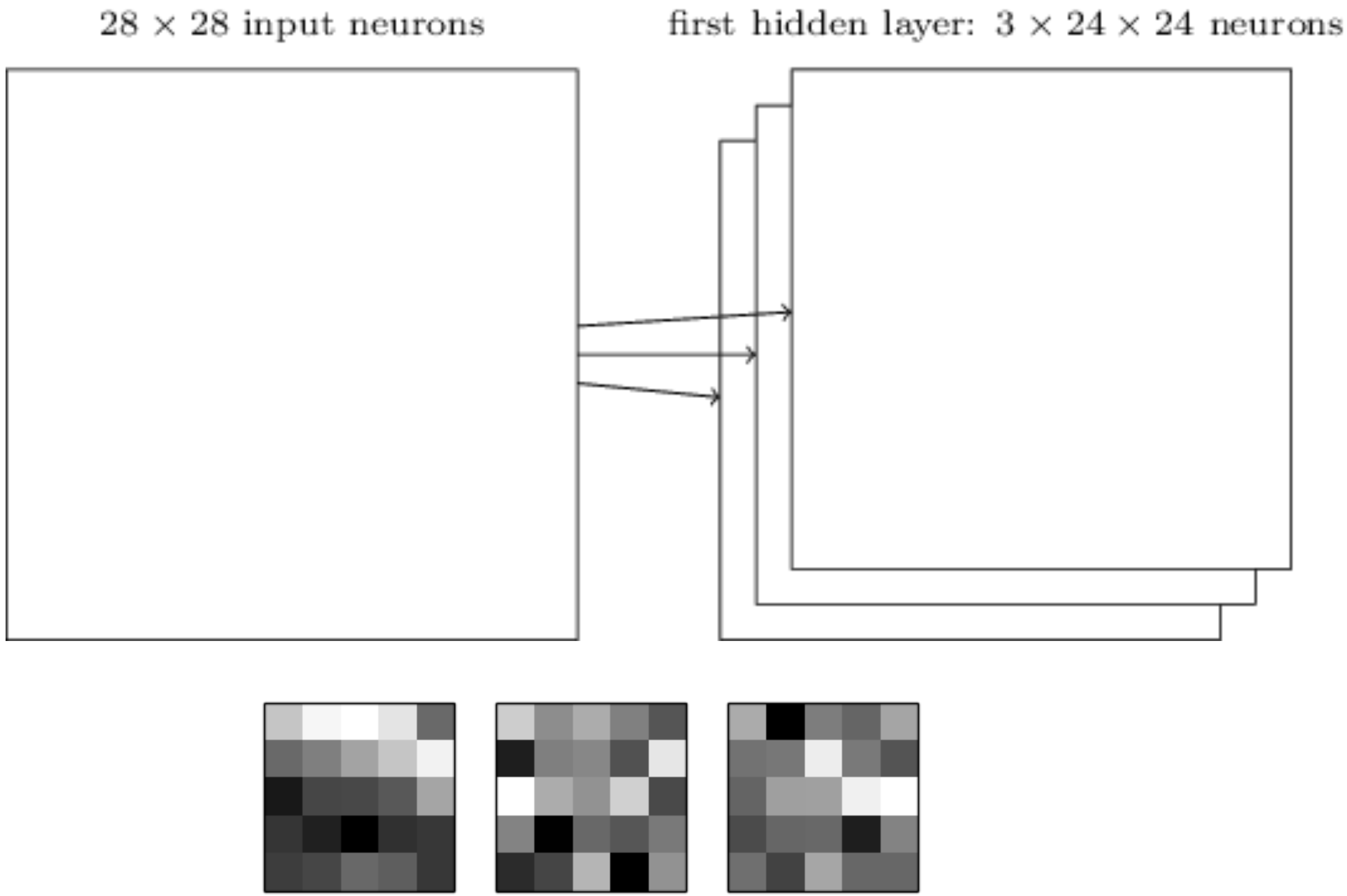
—> These two properties can be exploited to drastically reduce the number of weights required by CNNs using so-called convolutional layers.

CONVOLUTIONAL LAYER

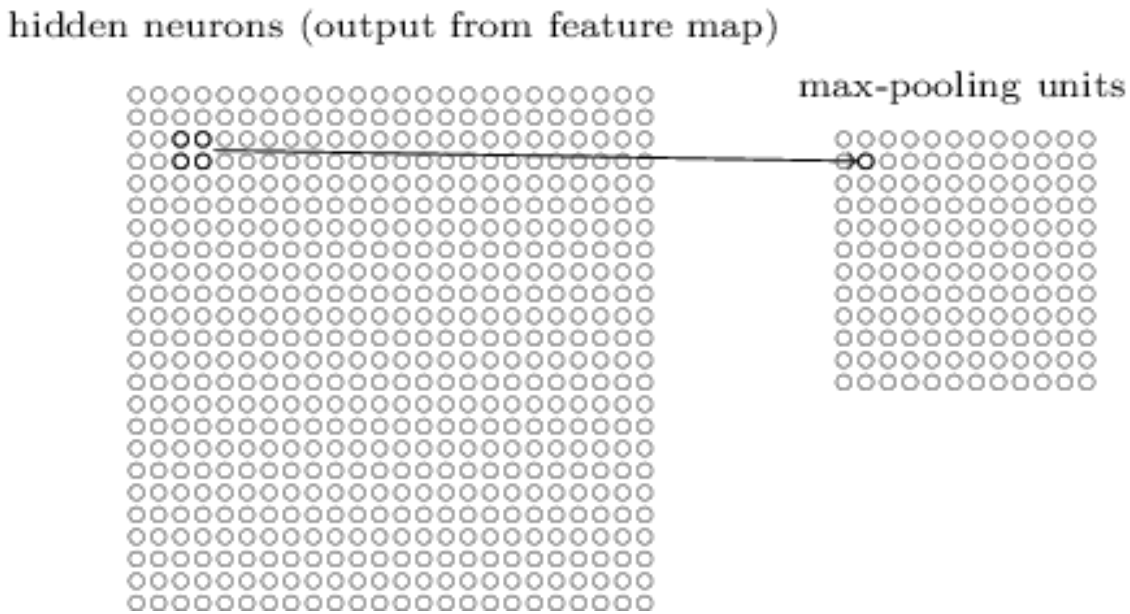


$$\sigma\left(b+\sum_{x=0}^{n_x}\sum_{y=0}^{n_y} w_{i,j}a_{i+x,j+y}\right)$$

FEATURE MAPS

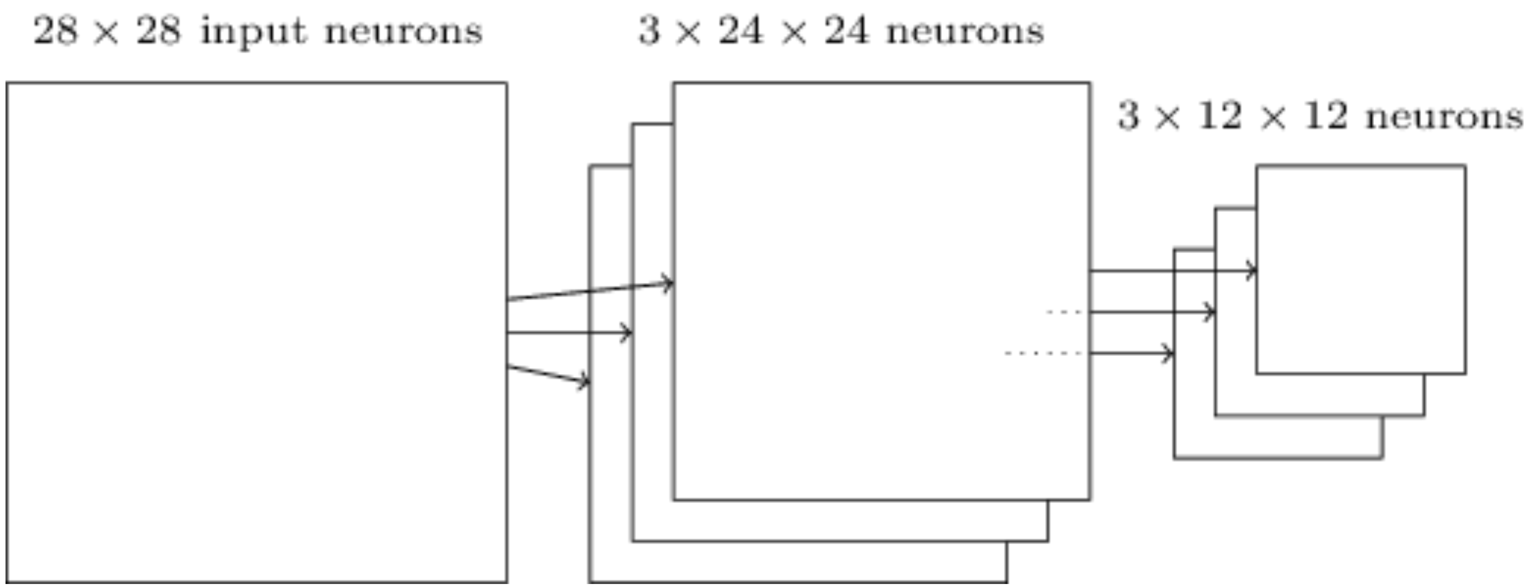


POOLING LAYER



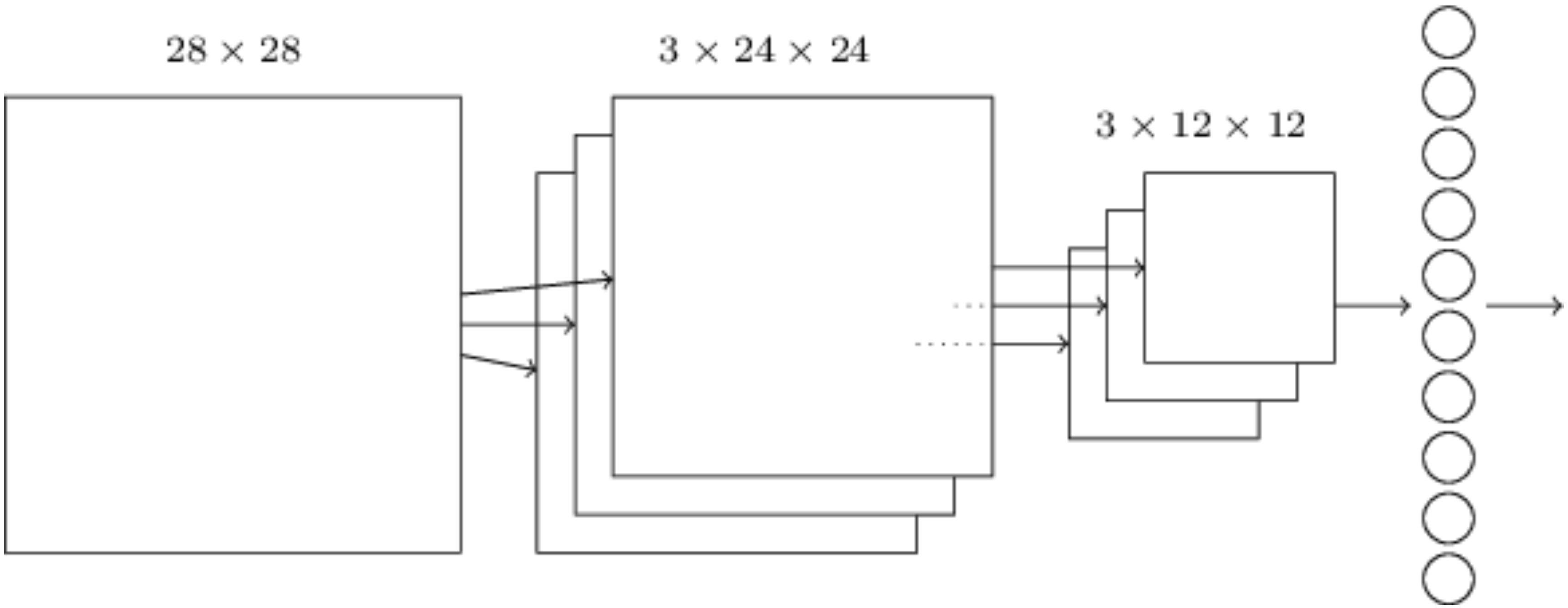
- Reduce the number of inputs by replacing all activations in a neighborhood by a single one.
- Can be thought as asking if a particular feature is present in that neighborhood while ignoring the exact location.

ADDING THE POOLING LAYERS



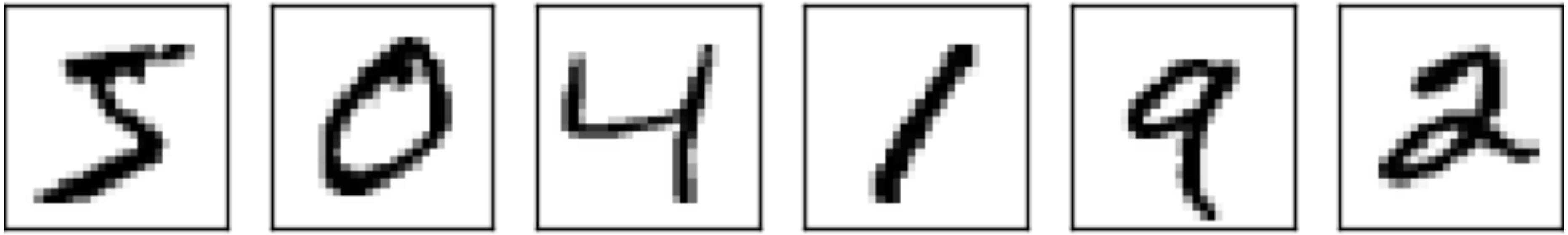
The output size is reduced by the pooling layers.

ADDING A FULLY CONNECTED LAYER



- Each neutron 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

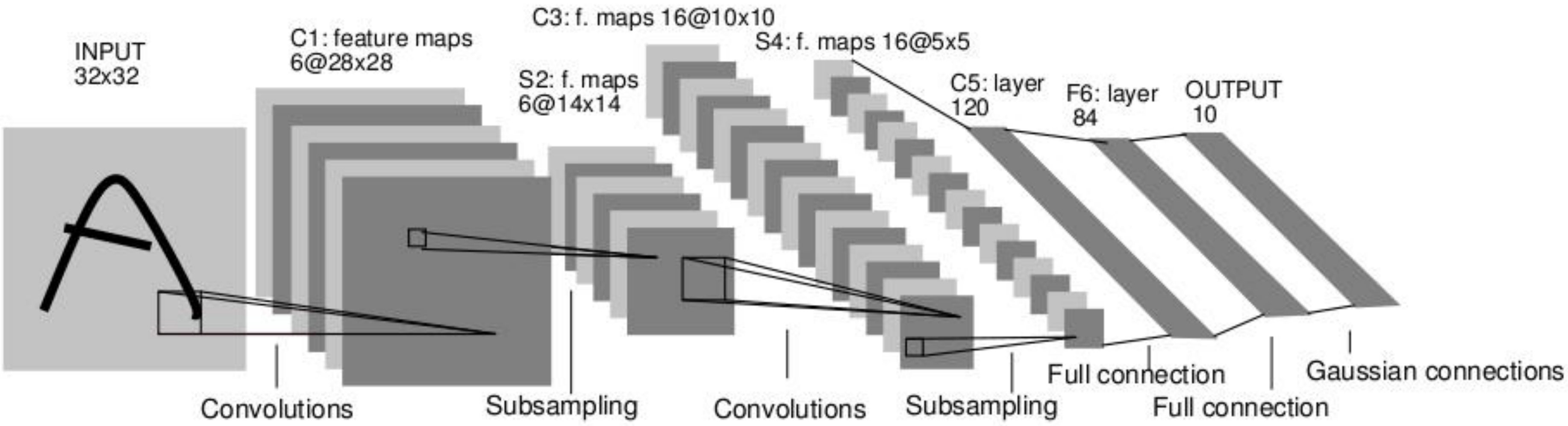


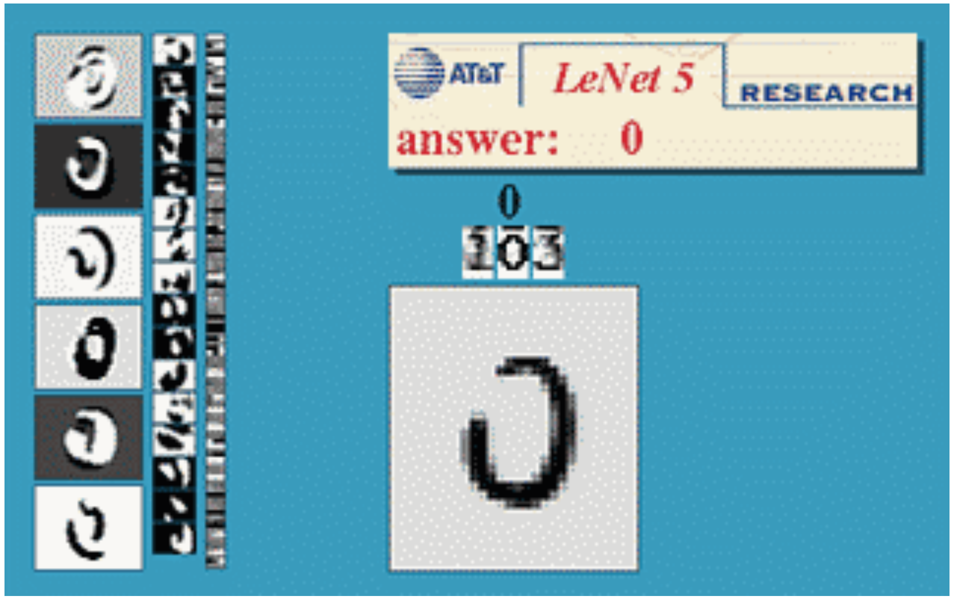
• The network takes as input 28x28 images represented as 784D vectors.

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

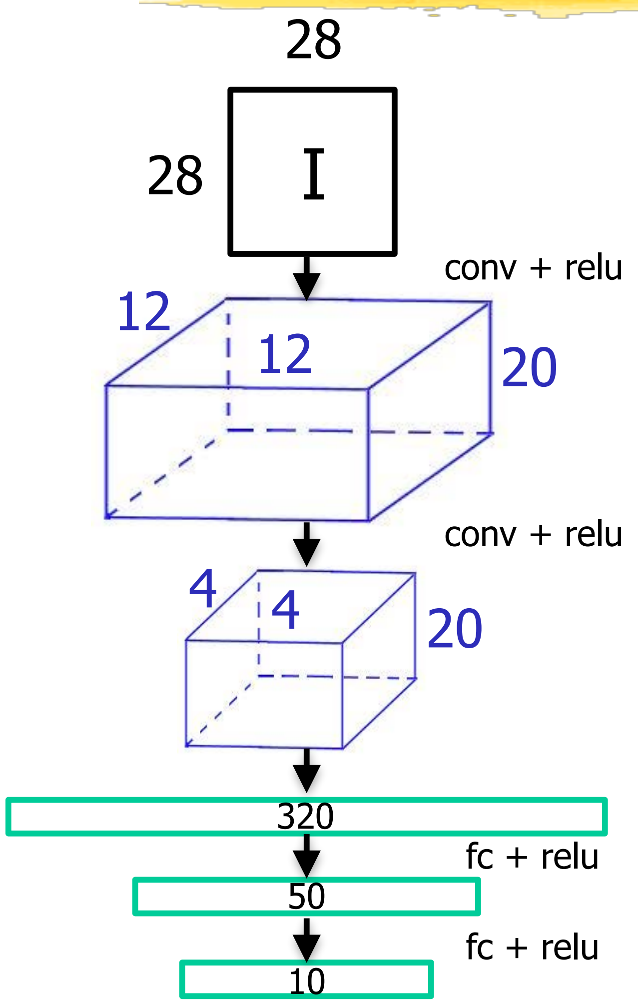
• There are 50’000 training pairs of images and the corresponding label, 10’000 validation pairs, and 5’000 testing pairs.

LeNet (1989-1999)





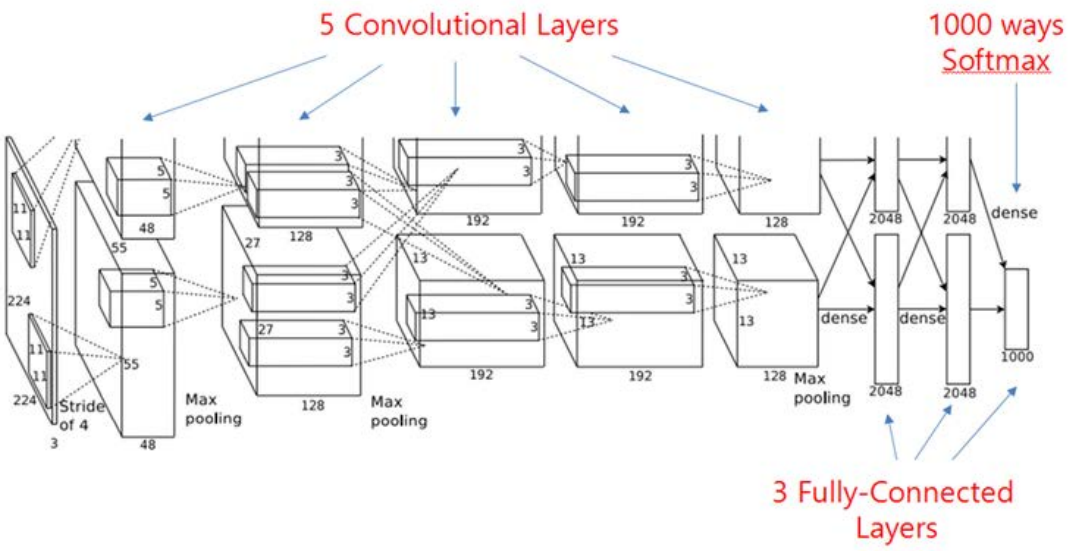
IS MAX POOL REQUIRED?



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)



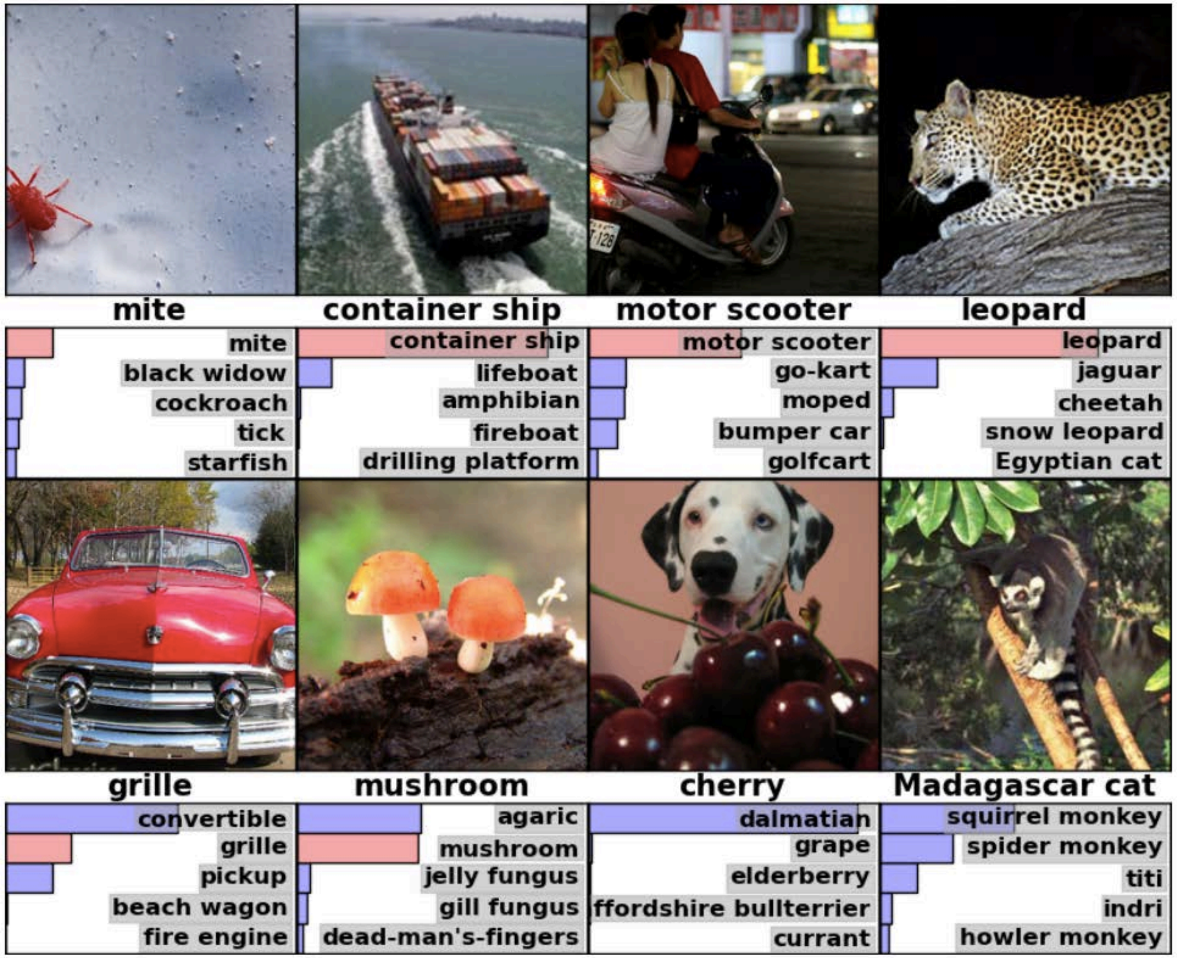
Task: Image classification

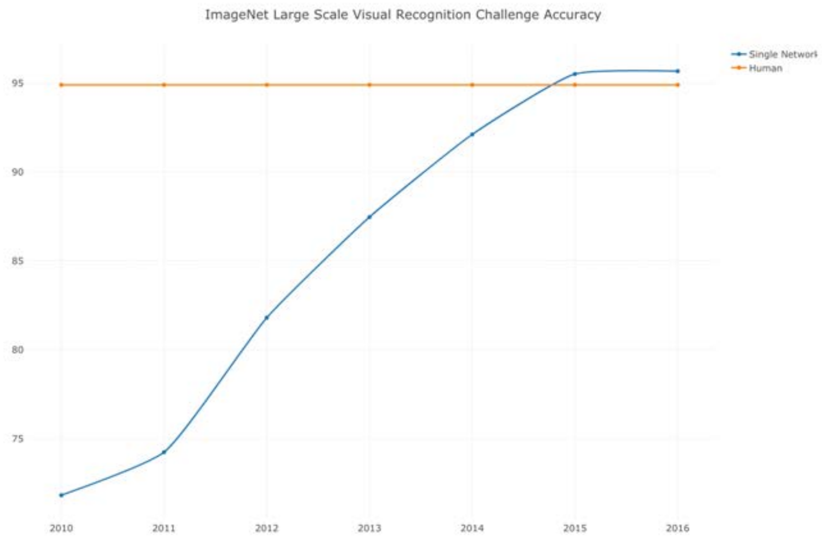
Training images: Large Scale Visual Recognition Challenge 2010

Training time: 2 weeks on 2 GPUs

Major Breakthrough: Training large networks has now been shown to be practical!!

AlexNet RESULTS

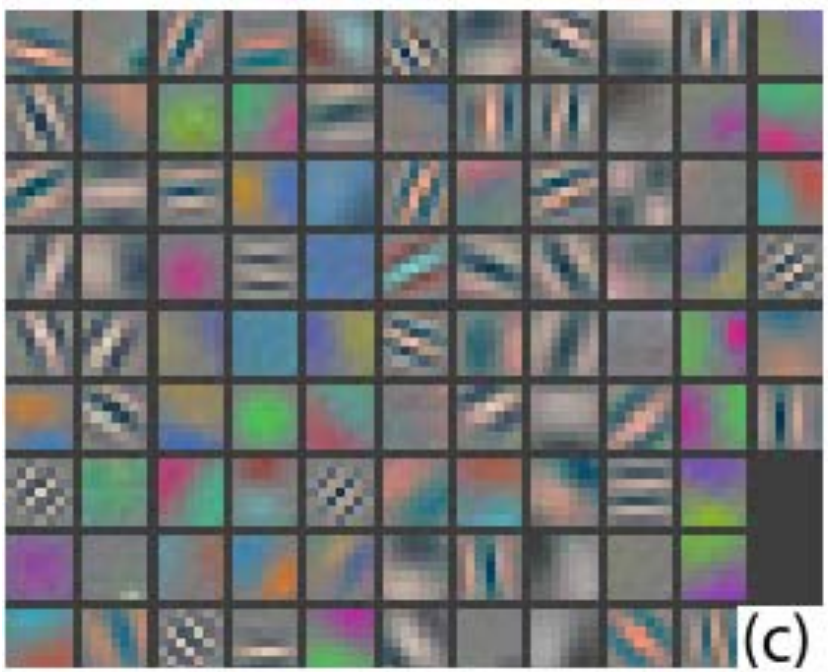




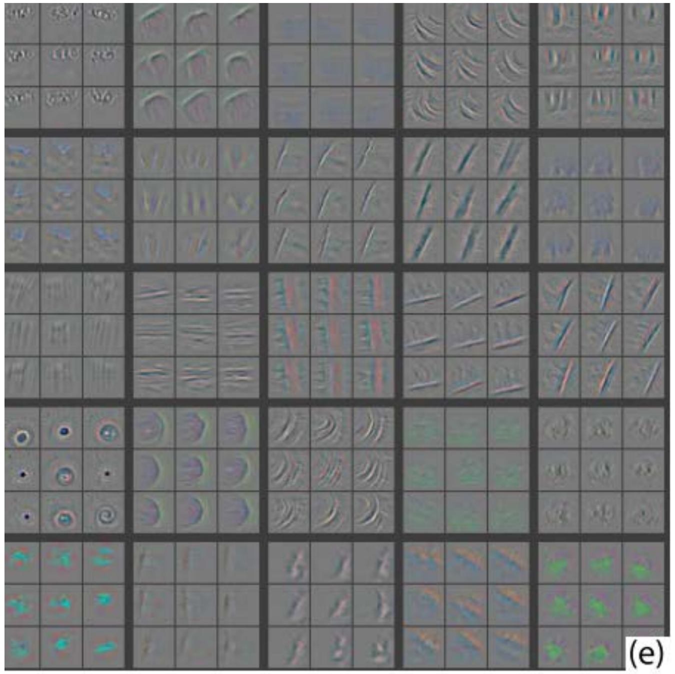
- At the 2012 ImageNet Large Scale Visual Recognition Challenge, AlexNet achieved a top-5 error of 15.3%, more than 10.8% lower than the runner up.
- Since 2015, networks outperform humans on this task.

Krizhevsky, NIPS'12

FEATURE MAPS



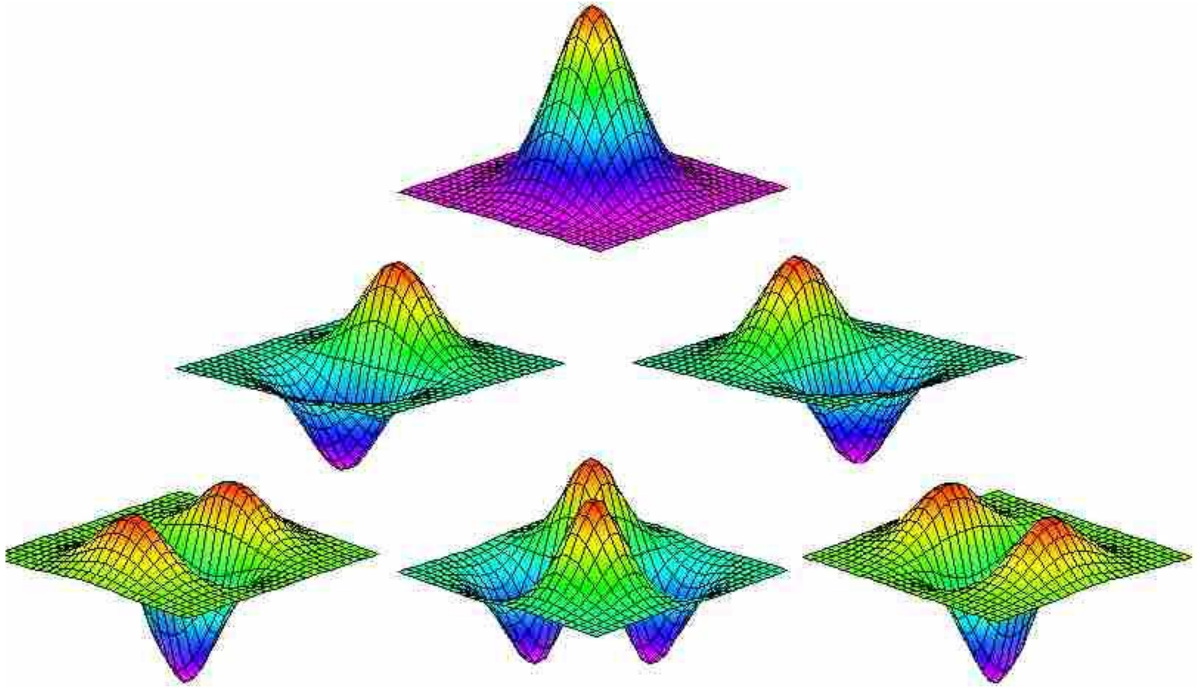
First convolutional layer

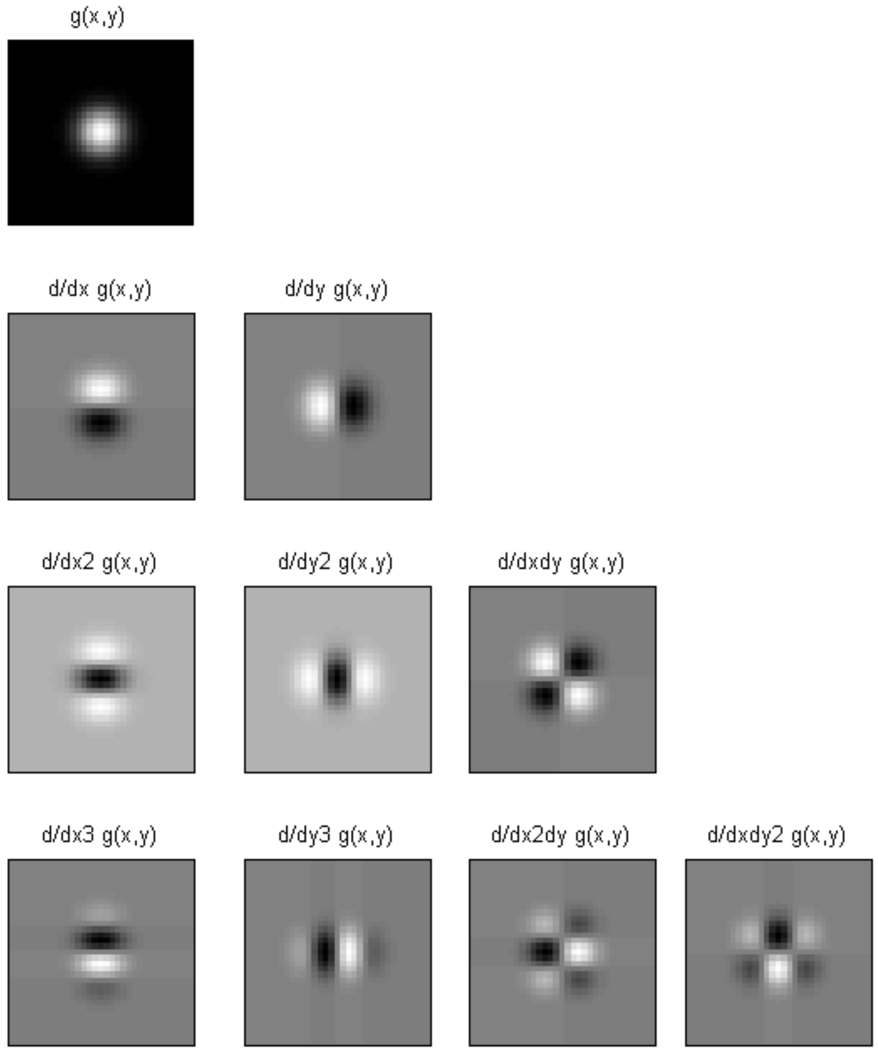


Second convolutional layer

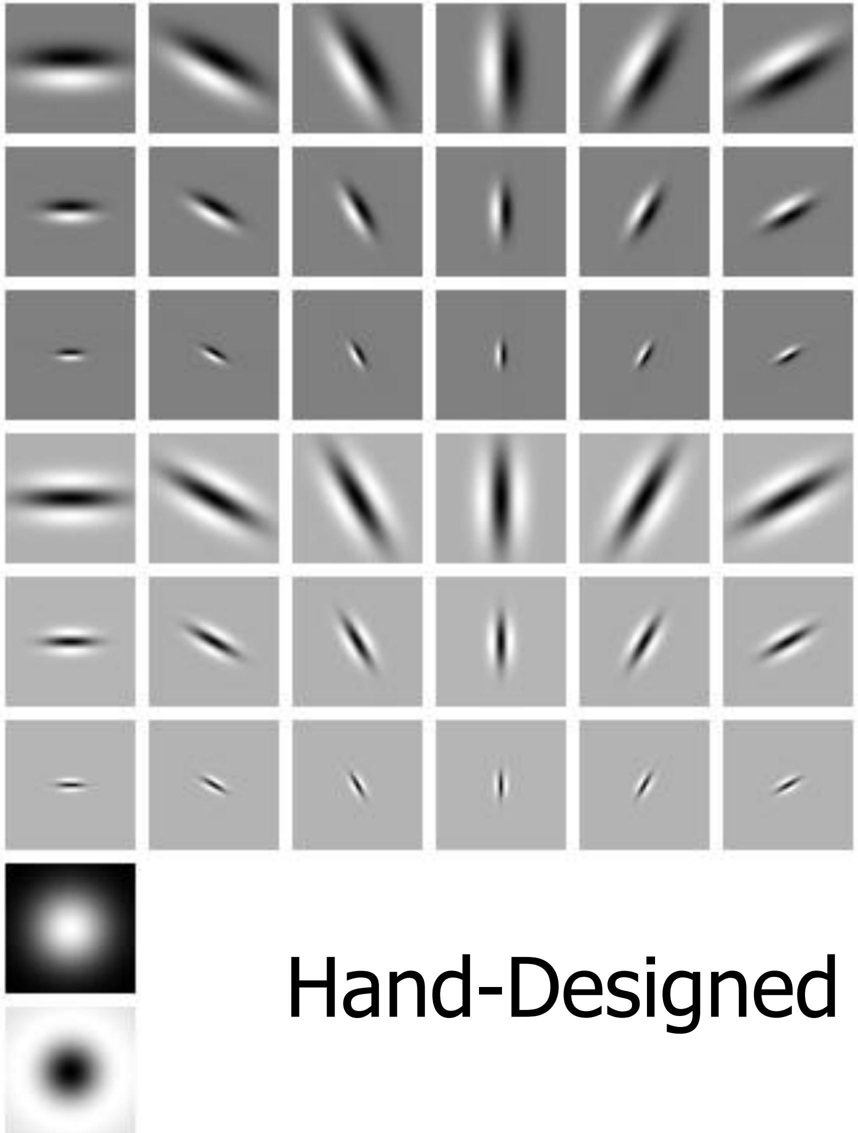
- Some of the convolutional masks seem very similar to oriented Gaussian or Gabor filters!
- Much ongoing work to better understand this.

HIGHER ORDER DERIVATIVES

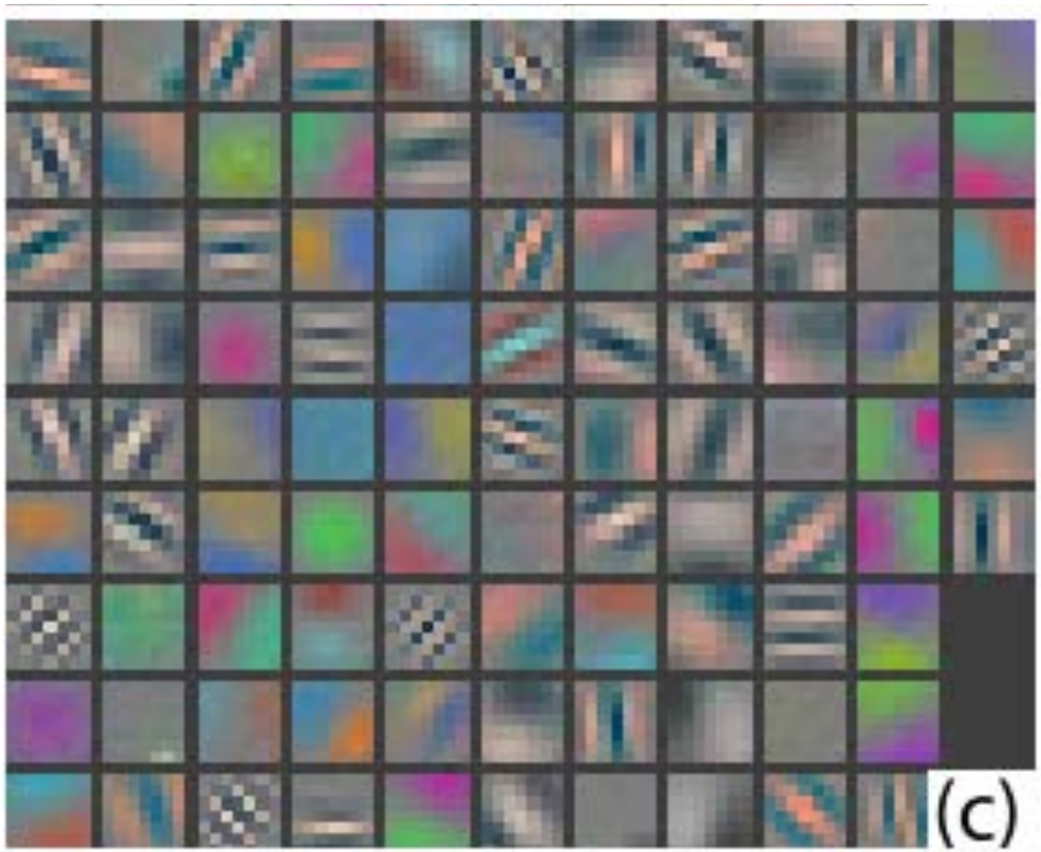




FILTER BANKS

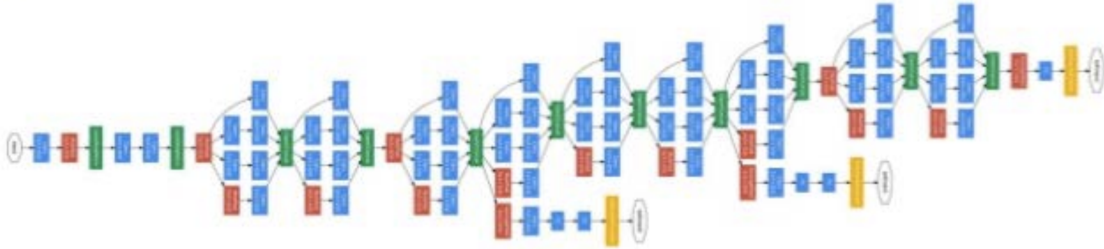


Hand-Designed



Learned

SIZE AND DEPTH MATTER



"hibiscus" "dahlia"

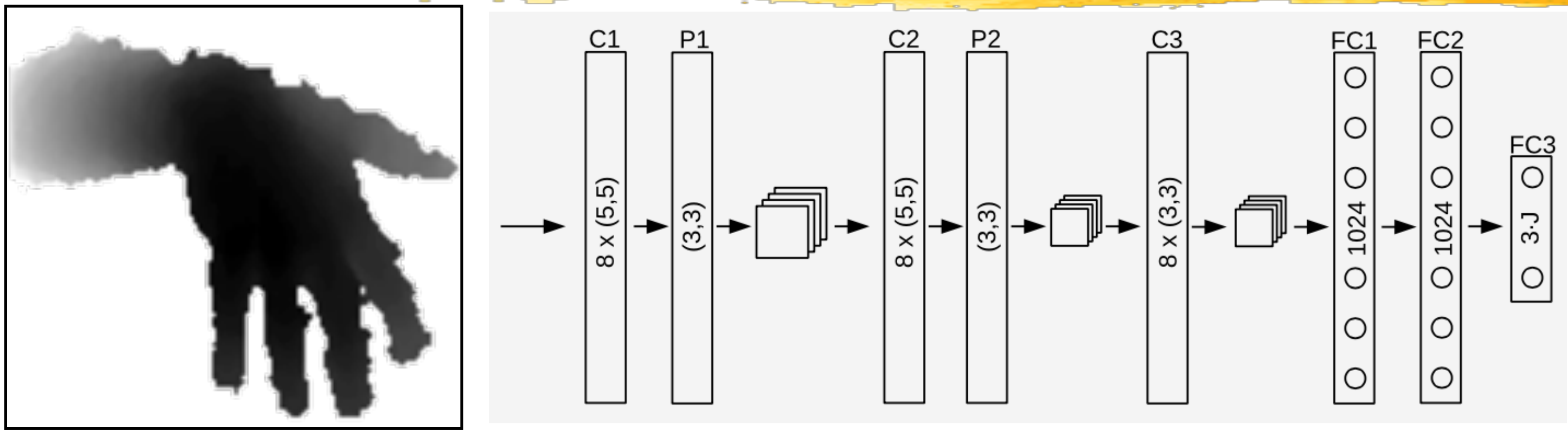
VGG19, 3 weeks of training.

GoogleLeNet.

"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)

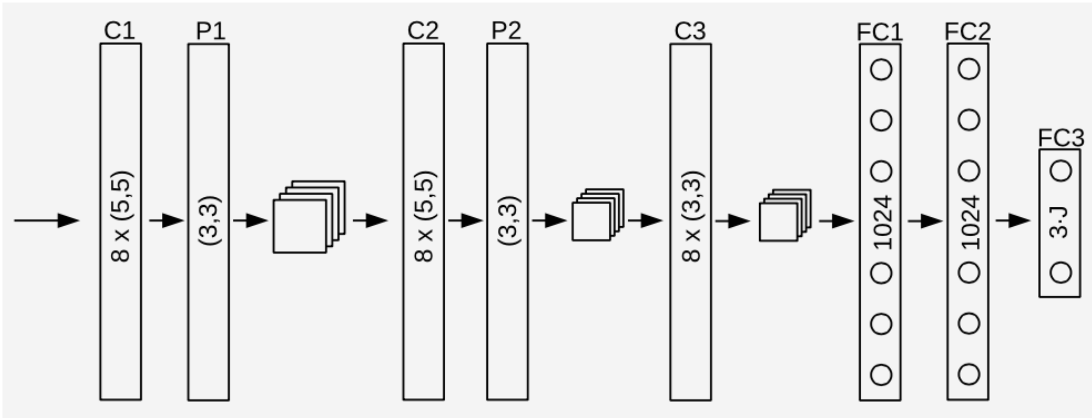


Input: Depth image.

Output: 3D pose vector.

Oberweger et al. , ICCV'15

REGRESSION

Input  Output

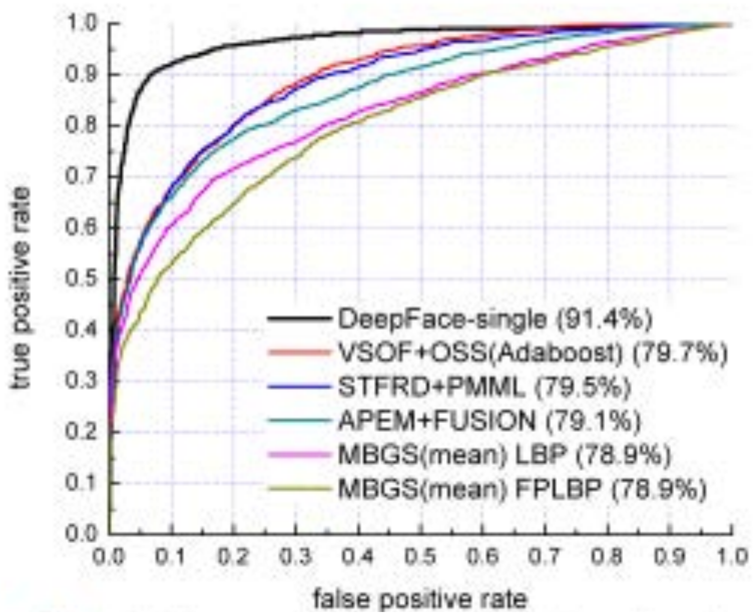
Network parameters are found by minimizing an objective function of the form

$$\min_{\mathbf{W}_l, \mathbf{B}_l} \sum_i ||\mathbf{F}(\mathbf{x}_i, \mathbf{W}_1, \ldots, \mathbf{W}_L, \mathbf{b}_1, \ldots, \mathbf{b}_L) - \mathbf{y}_i||^2$$

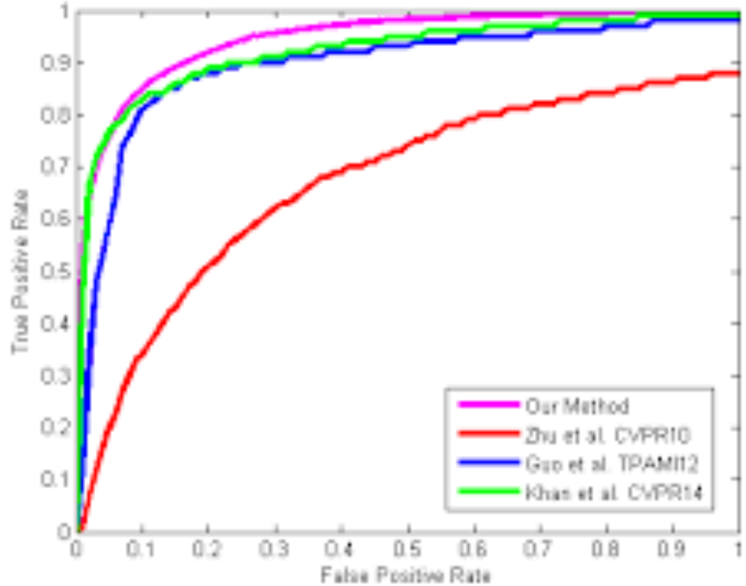
using

- stochastic gradient descent on mini-batches,
- dropout,
- hard example mining,
- ……….

ROC HUNTING

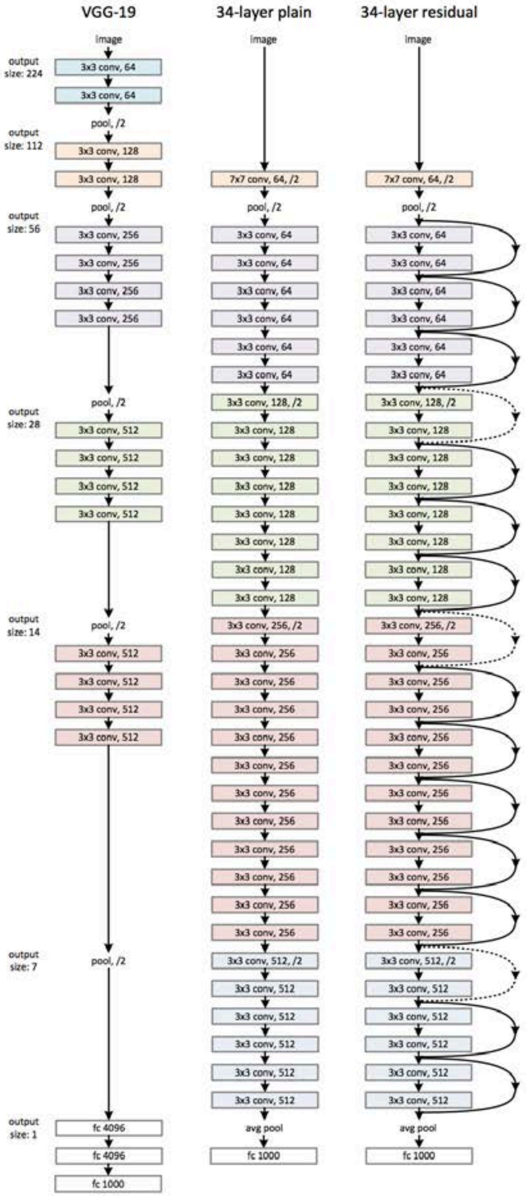


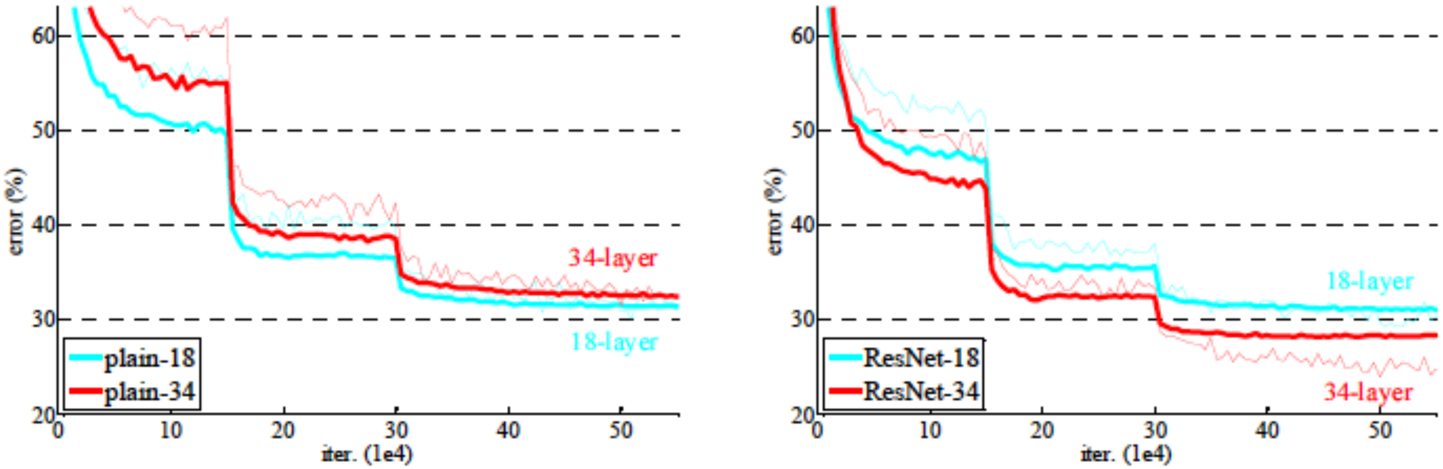
DeepFace
Taigman et al. 2014



Deep Edge Detection
Shen et al. 2015

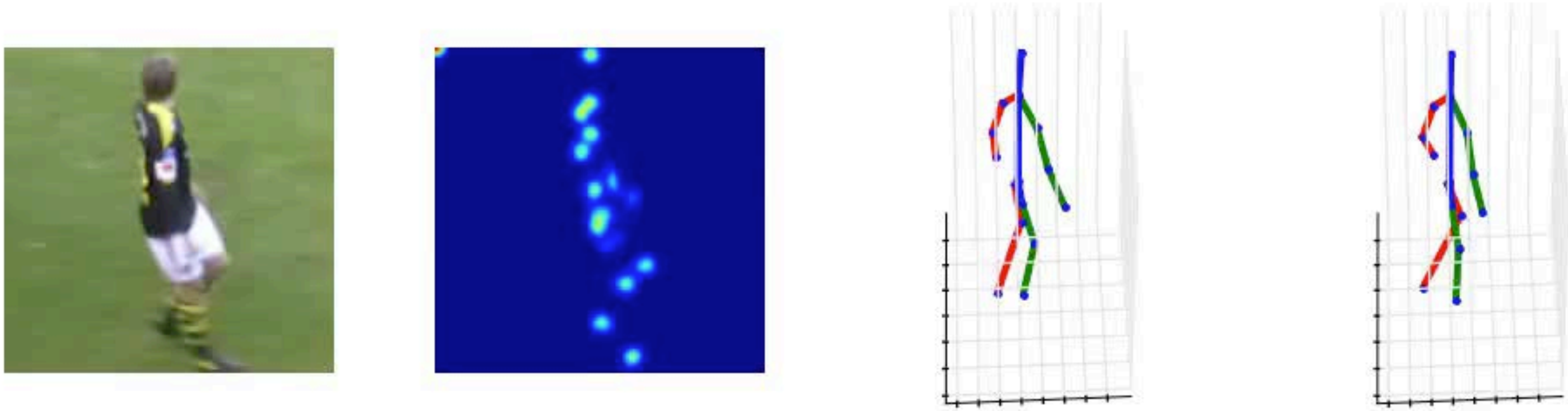
DEEPER AND DEEPER

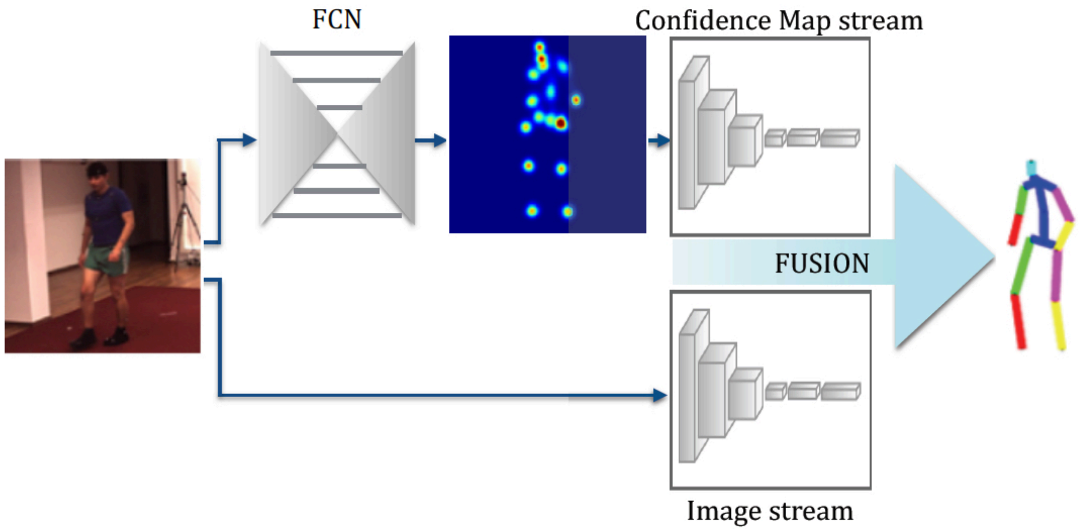




He et al. , CVPR’16

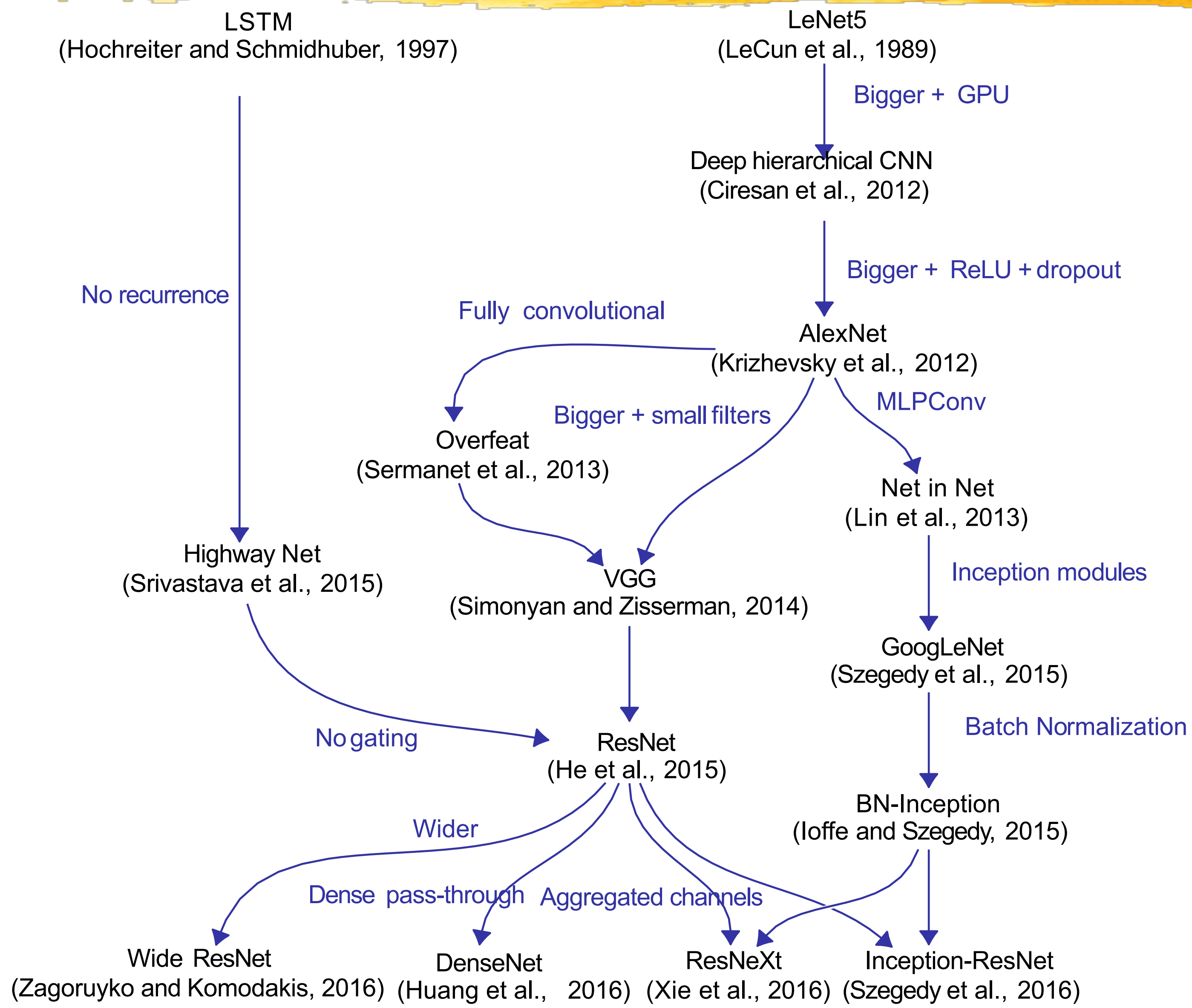
MONOCULAR POSE ESTIMATION





Tekin et al. , ICCV’17

IMAGE CLASSIFICATION TAXONOMY

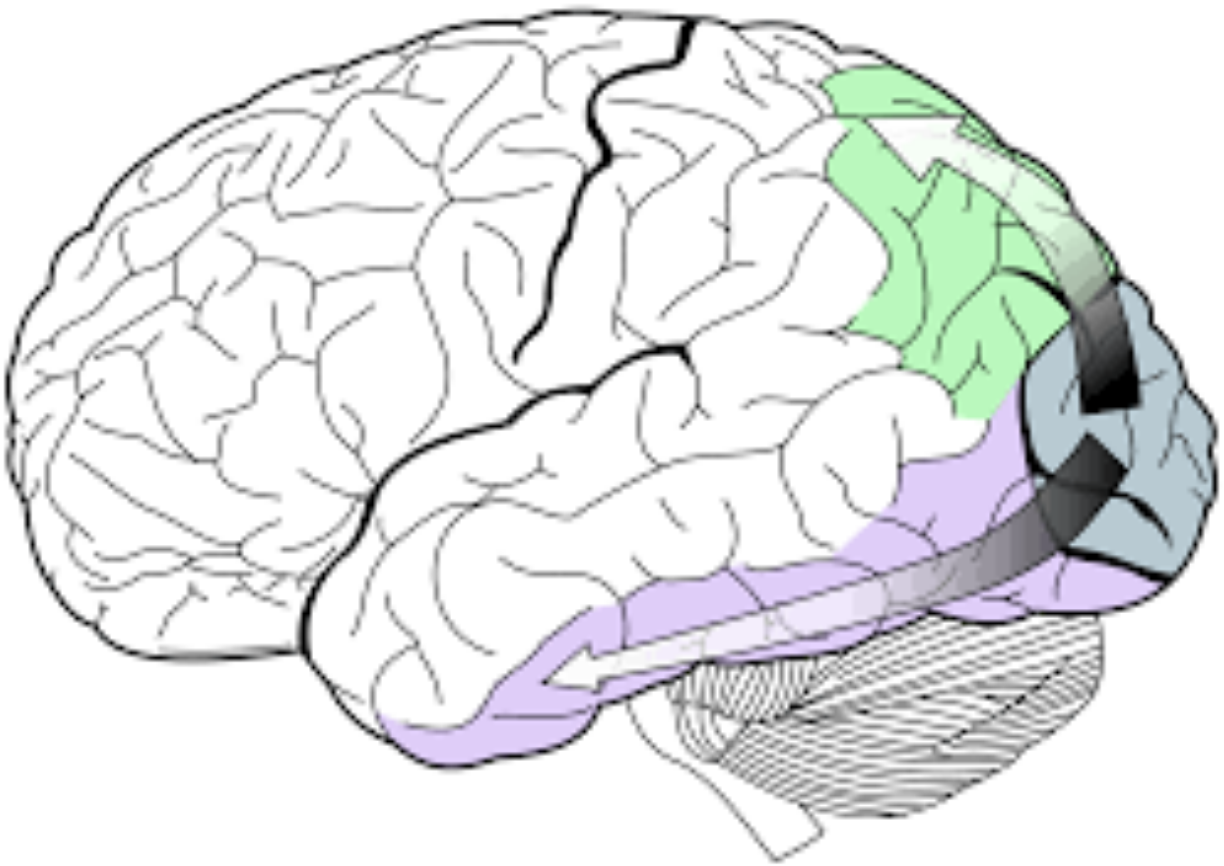


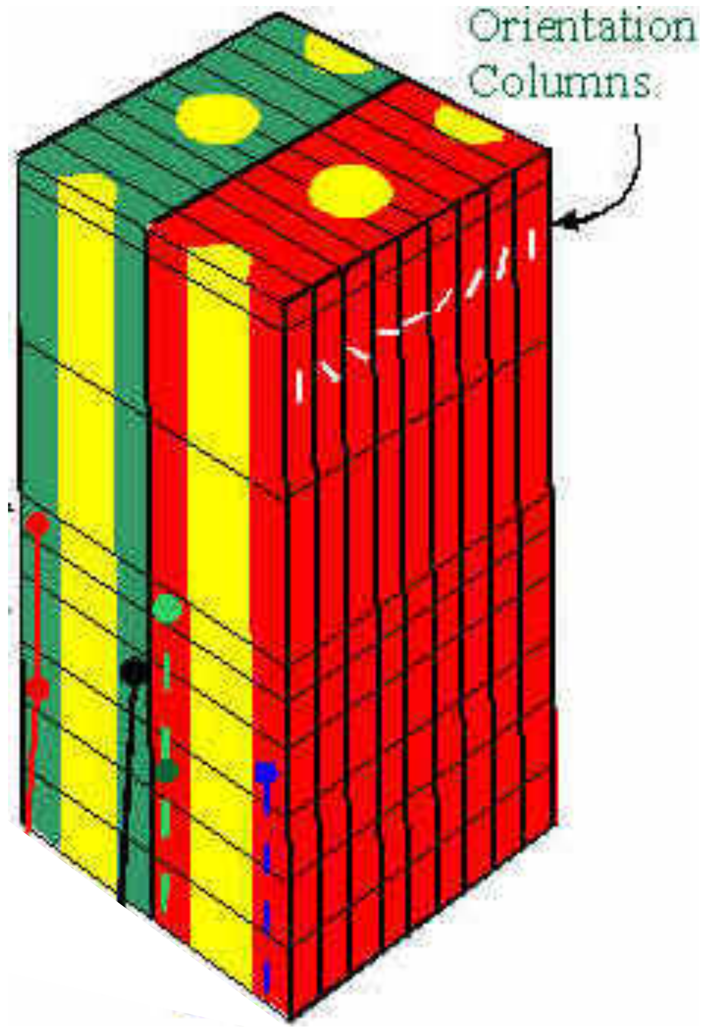
ANOTHER POINT OF VIEW

To summarize roughly the evolution of convnets for image classification:

- Standard ones are extensions of LeNet5.
- Everybody loves ReLU.
- Newer ones 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

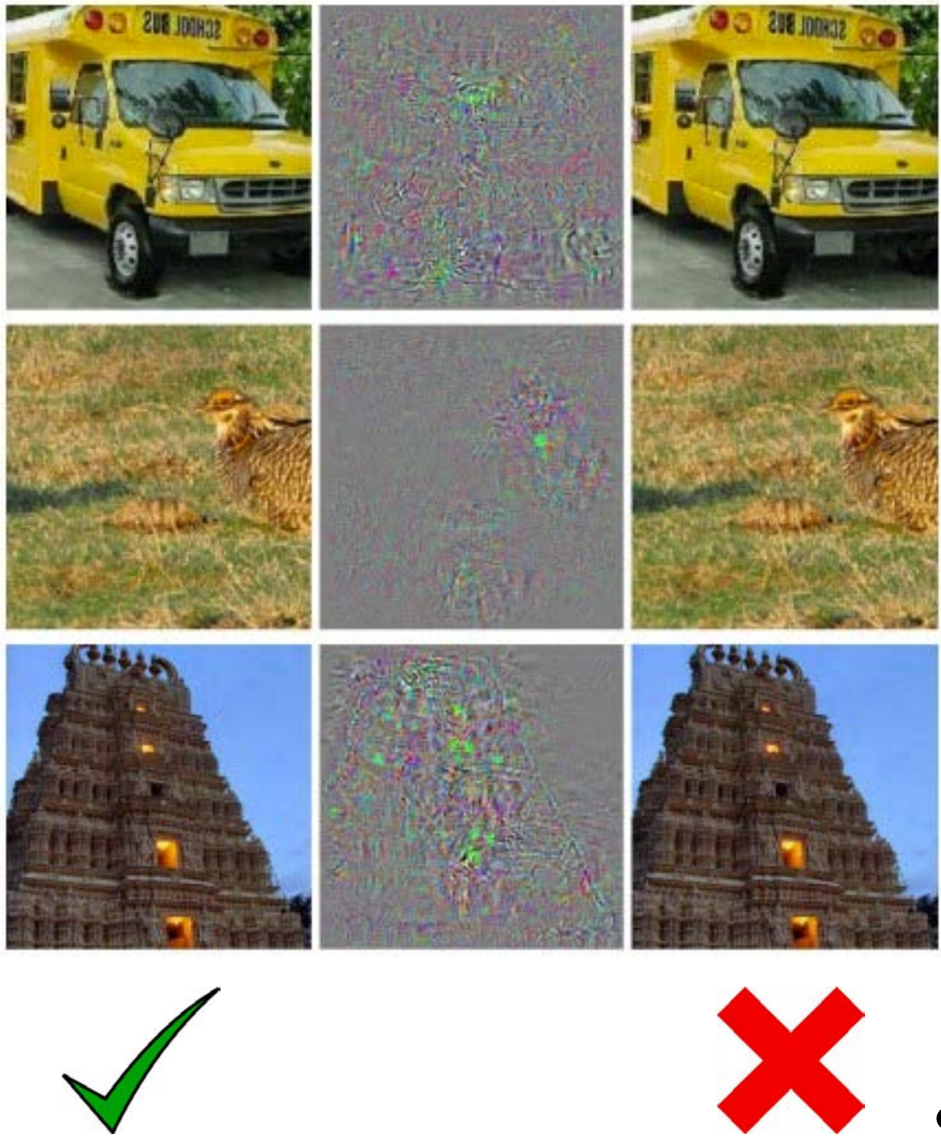




AlphaGo

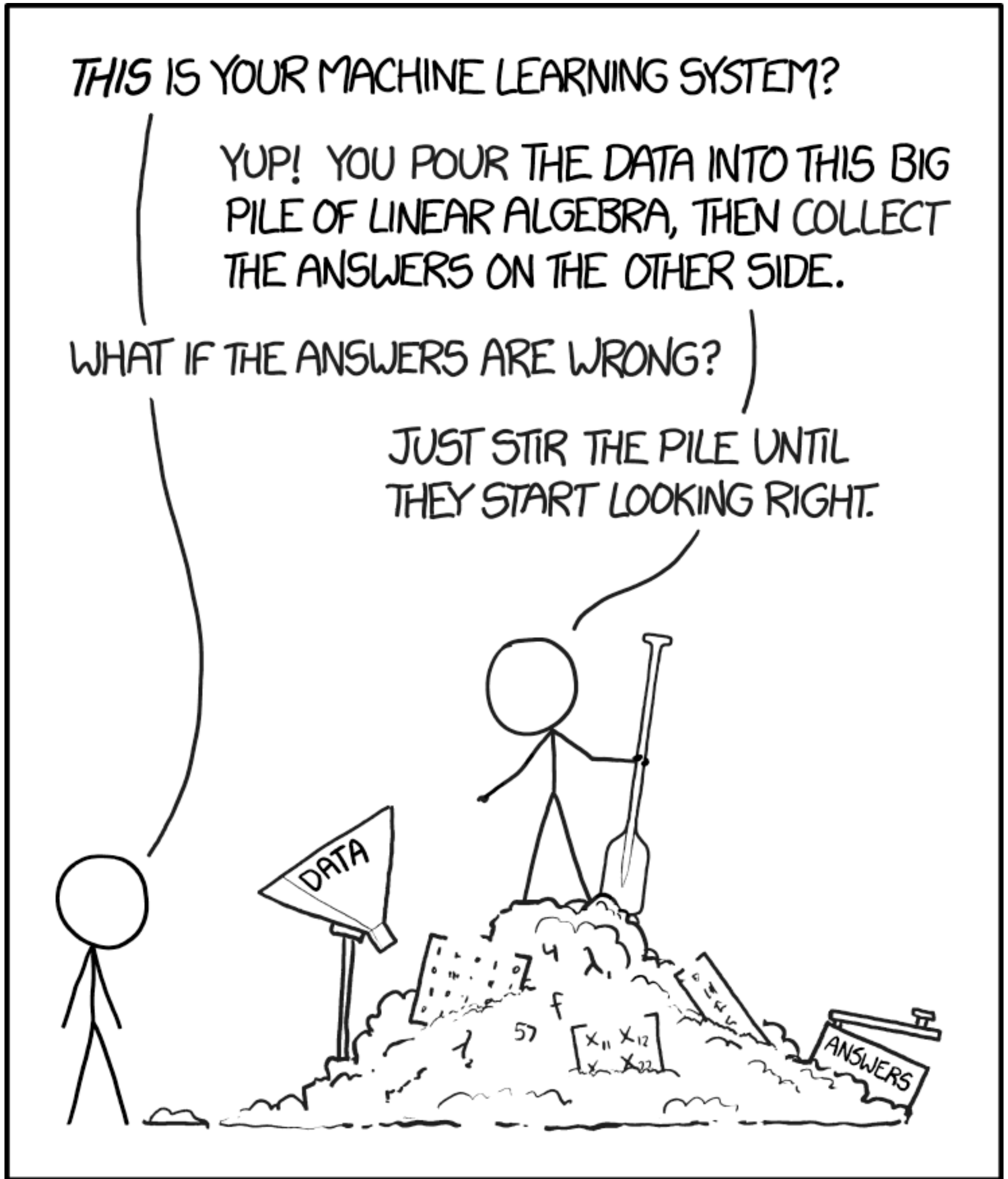
- Uses Deep Nets to find the most promising locations to focus on.
- Performs Tree based search when possible.
- Relies on reinforcement learning and other ML techniques to train.

ADVERSARIAL IMAGES



Szegedy et al. 2013

XKCD'S VIEW ON THE MATTER



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.