Deep Learning for Computer Vision

Spring 2019

http://vllab.ee.ntu.edu.tw/dlcv.html (primary)

https://ceiba.ntu.edu.tw/1072CommE5052 (grade, etc.)

FB: DLCV Spring 2019

Yu-Chiang Frank Wang 王鈺強, Associate Professor

Dept. Electrical Engineering, National Taiwan University

2019/03/13

Distance Metric of p-Norm

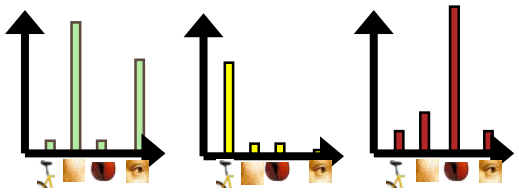
- Remarks
  - We have $\|x\|_p = \left(\sum_{i=1}^{d} {x_i}^p\right)^{1/p}$ and L_0 norm: $\|x\|_0$
    - Strictly speaking, L_0 norm is not a norm.
    - It is a cardinality function which has its definition in the form of L_p norm.
    - In practice, for L_0 norm, most engineers use the following definition instead:

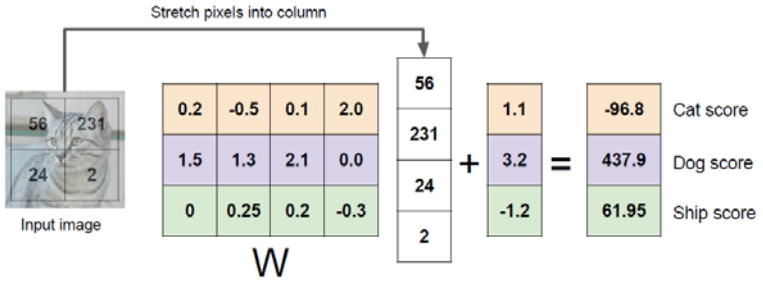
$$\|x\|_0 = \lim_{p \to 0} \sum_{i=1}^{d} |x_i|^p \ \ or \ \|x\|_0 = \#(i|x_i \neq 0)$$

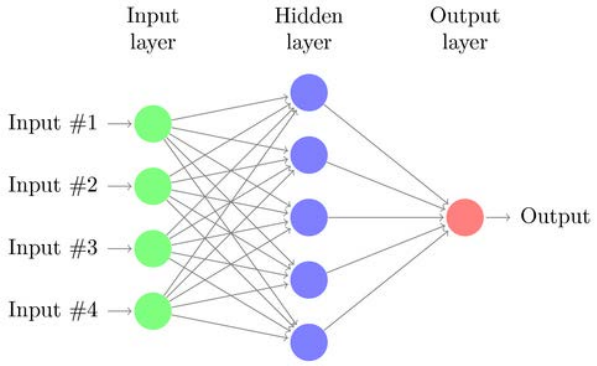
In other words, it returns the total number of non-zero entries in ***x***.

What's to Be Covered Today…

- General Framework for Visual Classification
  - Bag-of-Words Models as Image Features
- Intro to Neural Networks & CNN
  - Linear Classification
  - Neural Network for Machine Vision
  - Multi-Layer Perceptron
  - Convolutional Neural Networks



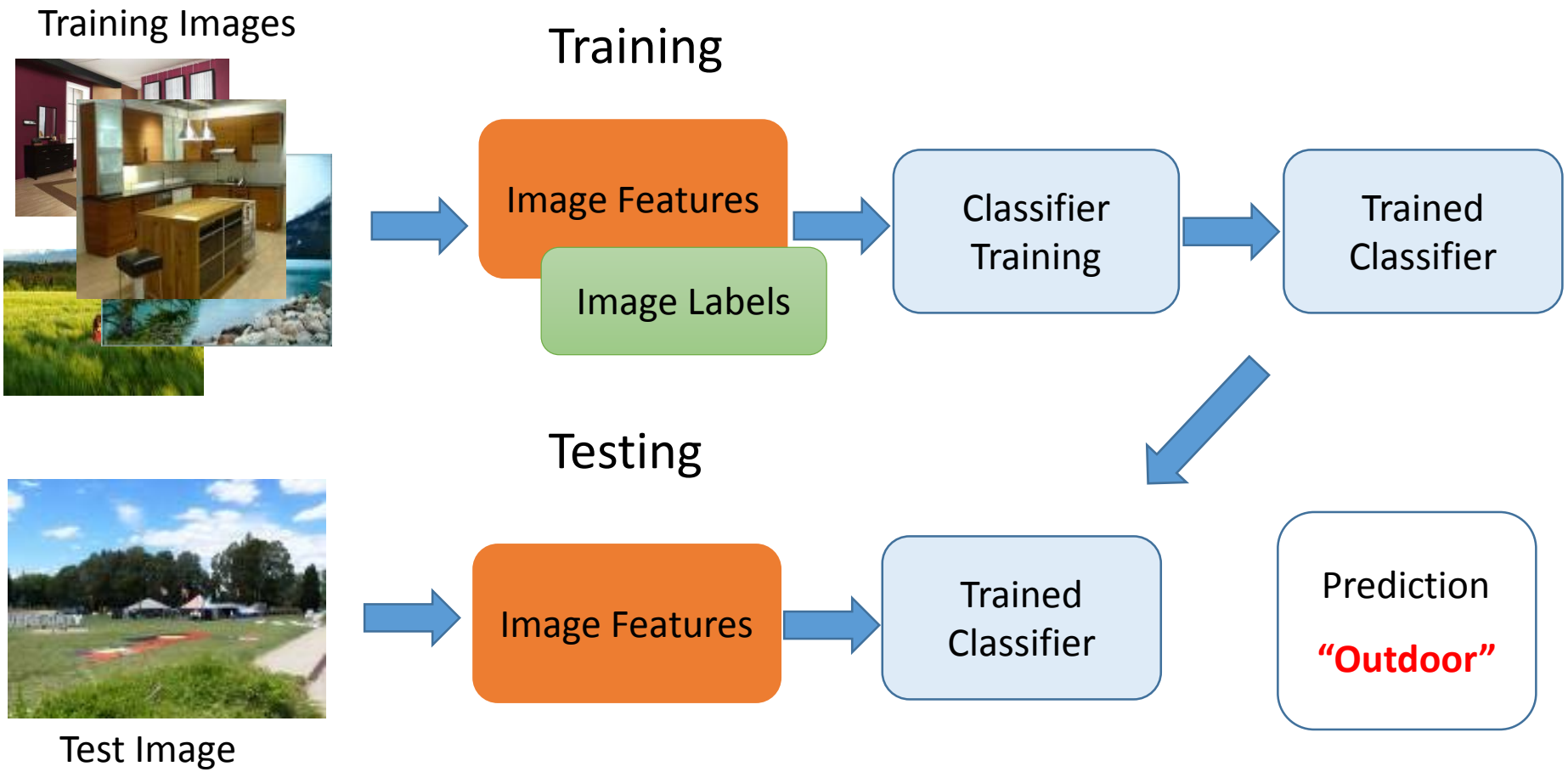




Many slides from Richard Turner, Fei-Fei Li, Yaser Sheikh, Simon Lucey, Kaiming He, and J.-B. Huang

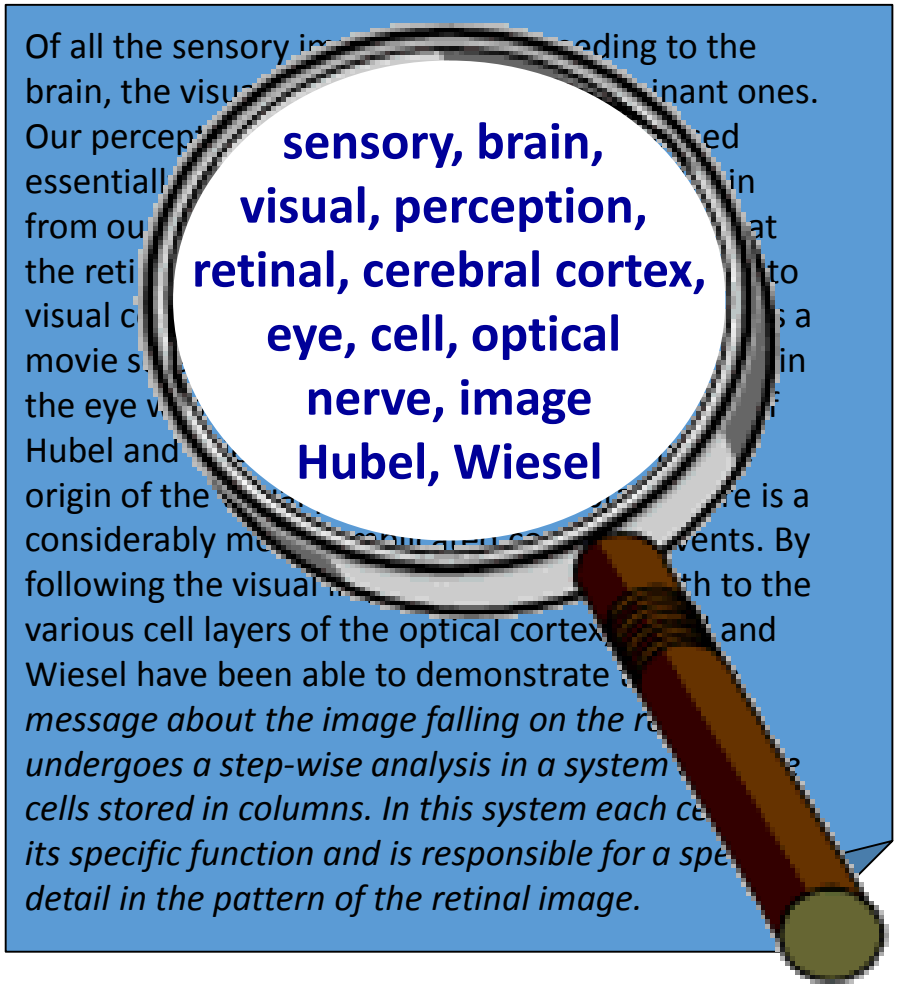
Supervised Learning for Visual Classification

- Training vs. Testing Phases



Bag-of-Words Models for Image Classification

- Analogy to document categorization



Bag of Words (or Visual Words)

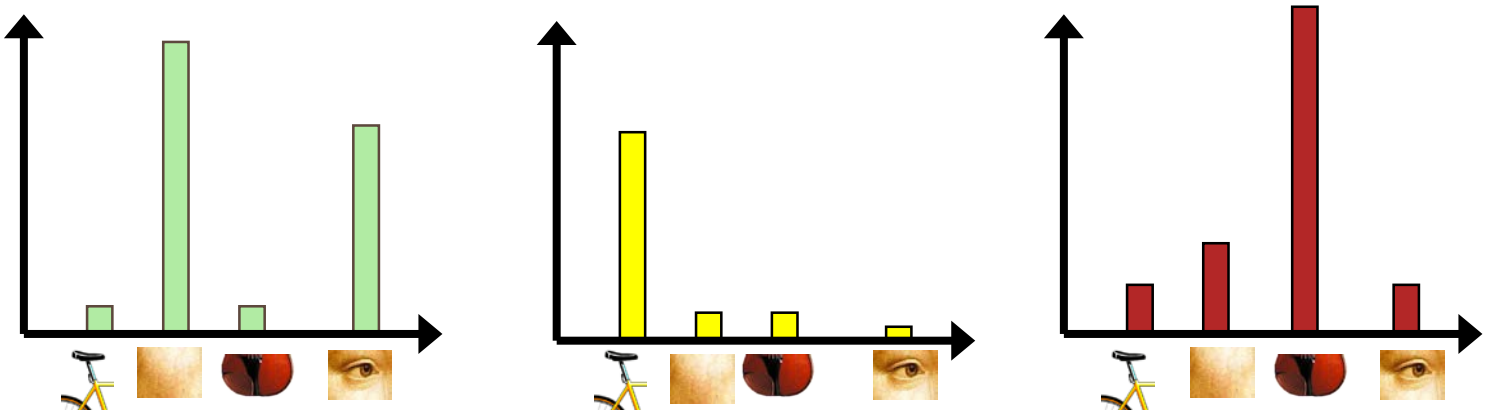
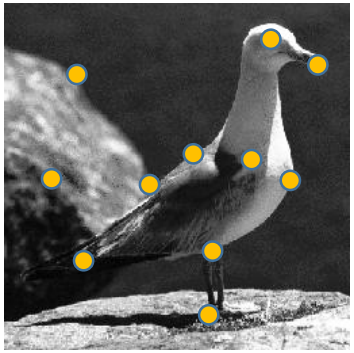


Image Representation: Histograms



- Take images with 2D features/descriptors as an example

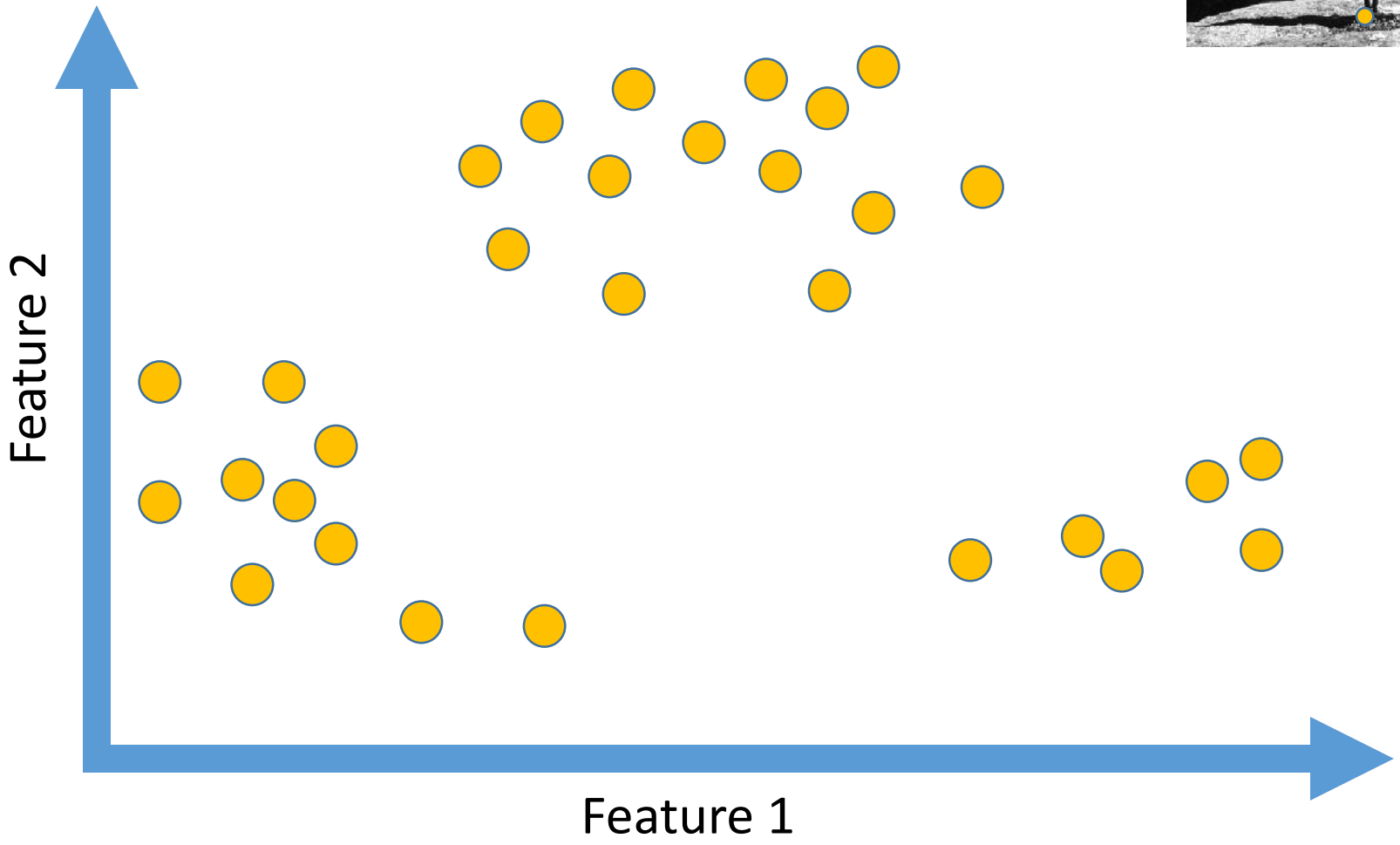


Image Representation: Histograms

- # of occurrence of data in each bin
- Marginal histogram of feature 1

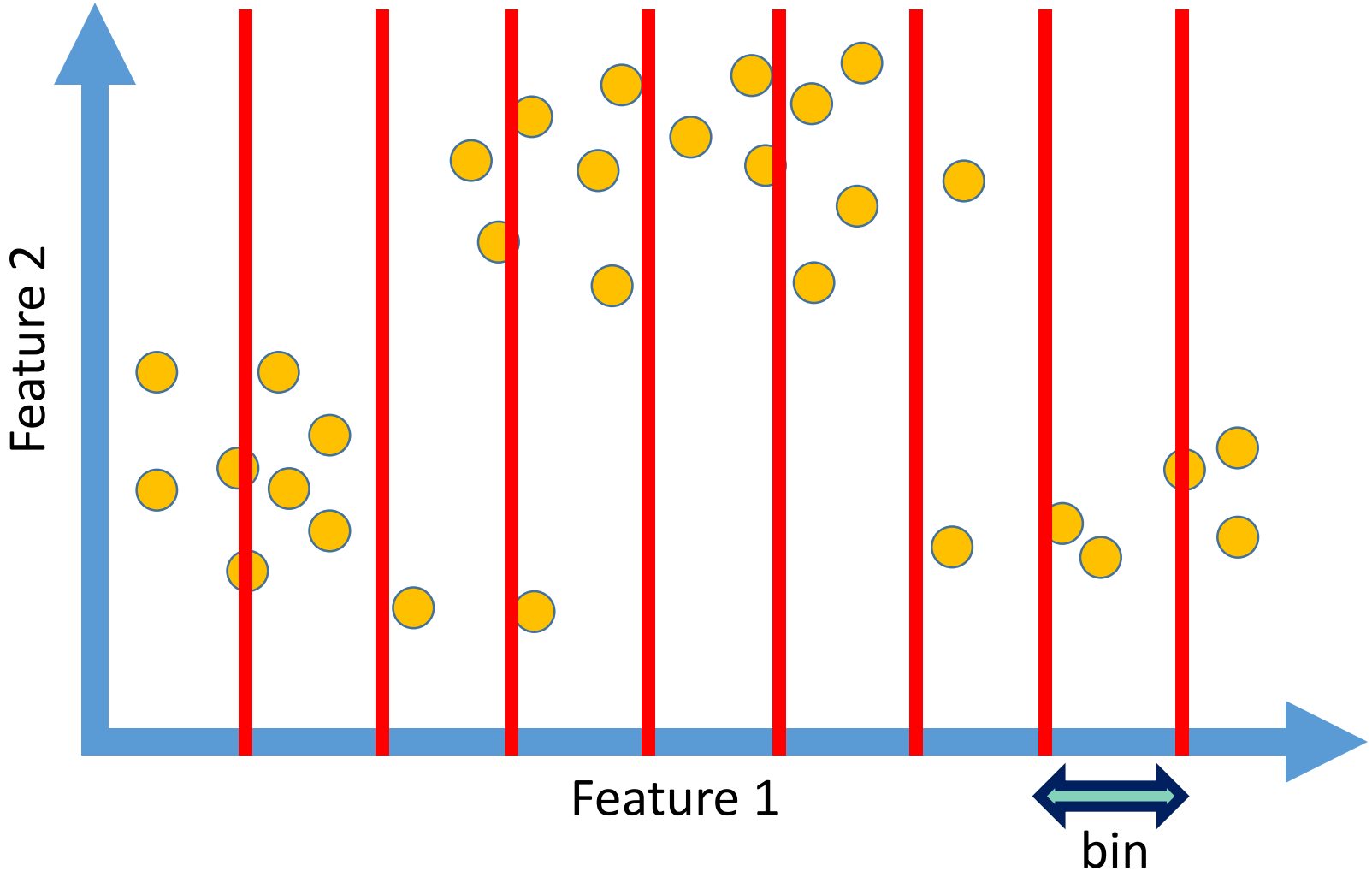


Image Representation: Histograms

- # of occurrence of data in each bin
- Marginal histogram of feature 2

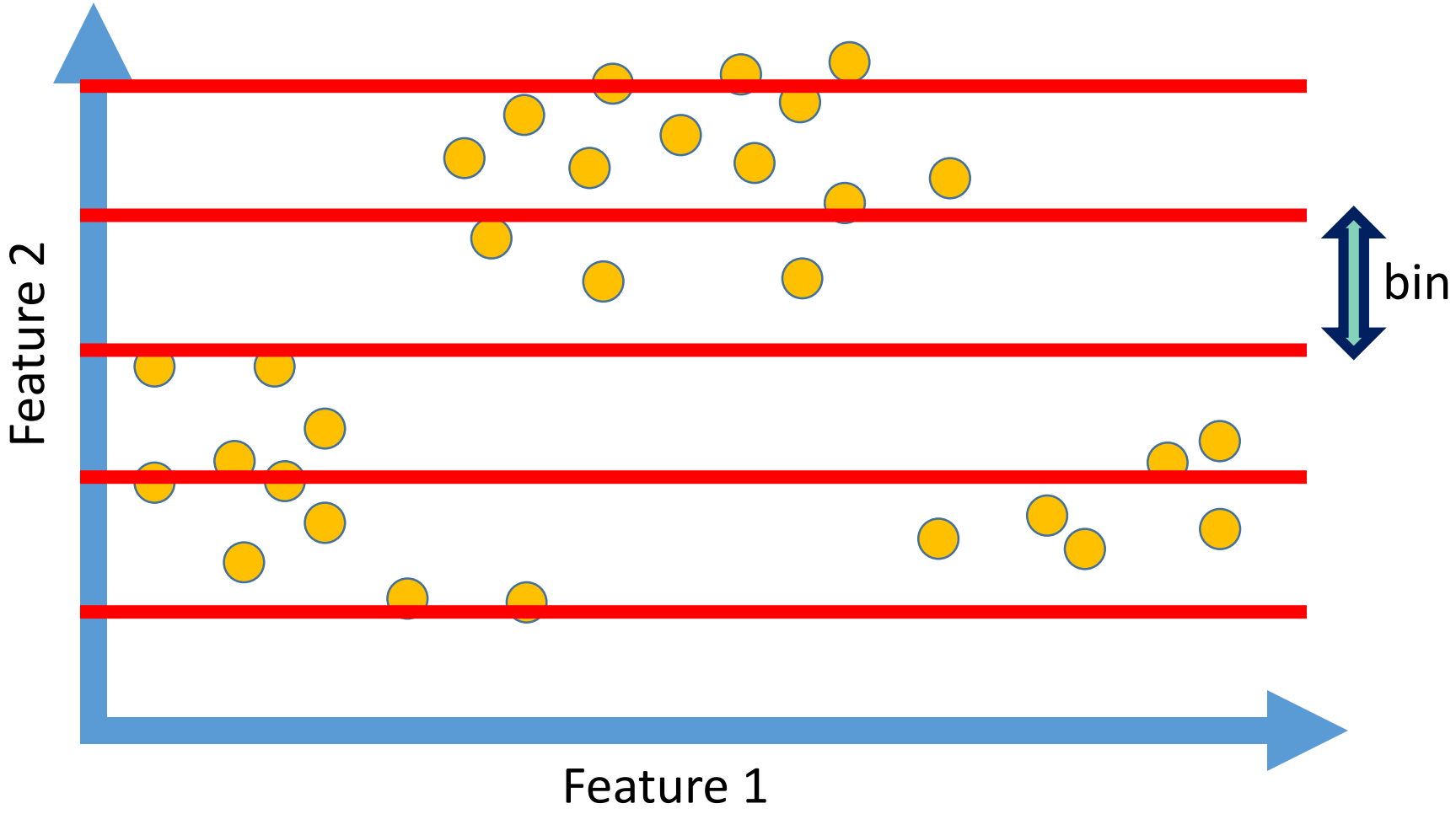


Image Representation: Histograms

- Better modeling (quantization) of multi-dimensional data
- Clustering

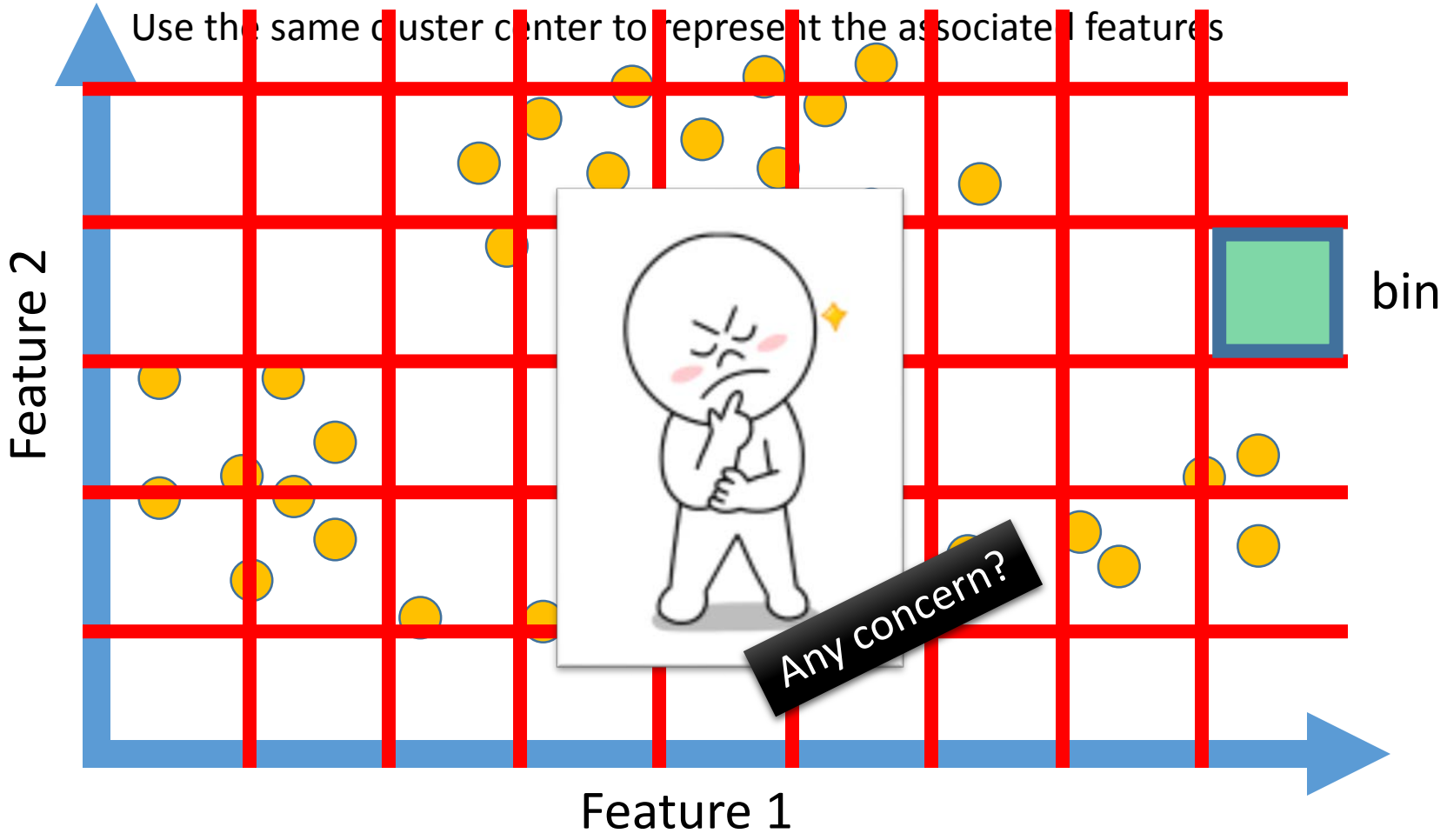
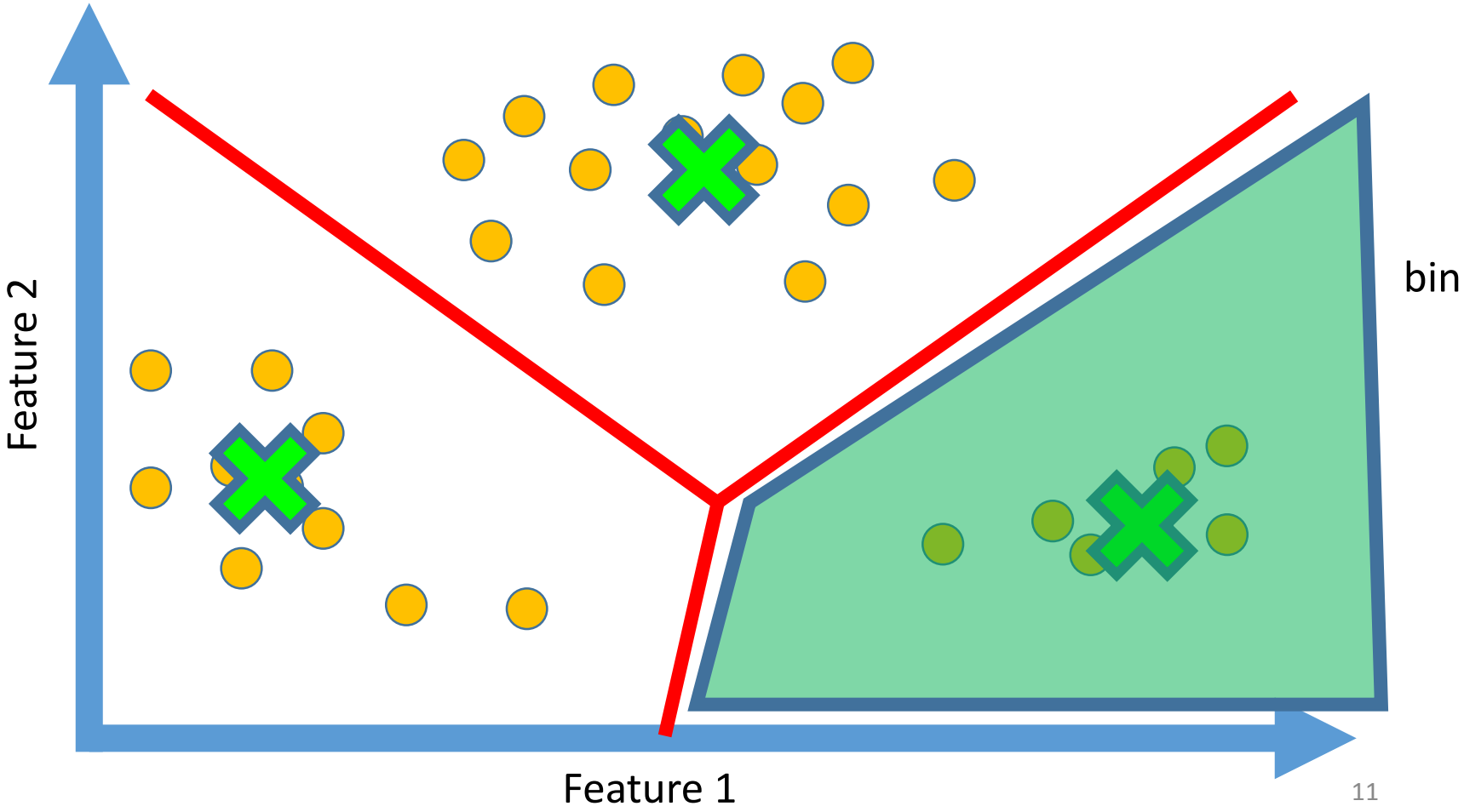


Image Representation: Histograms

- Better modeling (quantization) of multi-dimensional data
- Clustering
  - Use the same cluster center to represent the associated features



Remarks on Histogram-Based Image Representation

- Quantization
  - Grids vs. clusters

Fewer Bins
Need less data
Coarser representation

More Bins
Need more data
Finer representation

- Possible distance metrics
  - Euclidean distance
  - Histogram intersection kernel
  - Chi-squared distance
  - Earth mover's distance
    (min cost to transform one distribution to another)

$$\text{histint}(h_i, h_j) = 1 - \sum_{m=1}^{K} \min\left(h_i(m), h_j(m)\right)$$

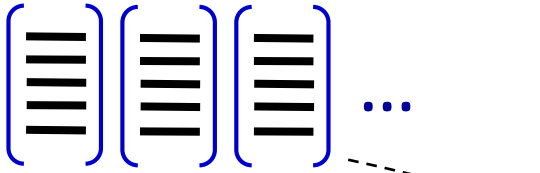
$$\chi^2(h_i, h_j) = \frac{1}{2} \sum_{m=1}^{K} \frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)}$$

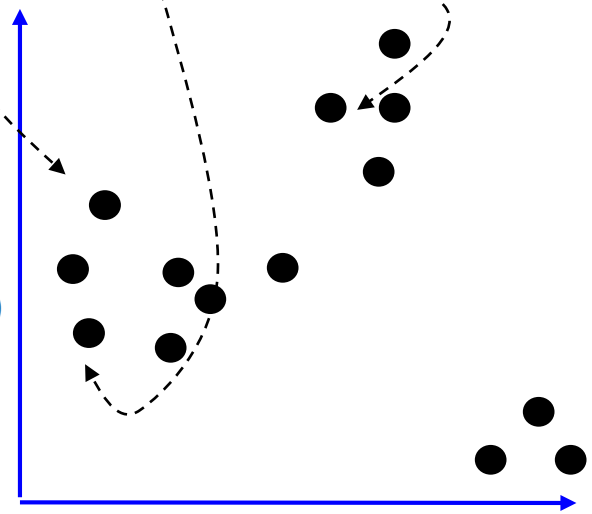
Bag-of-Words for Image Classification

- Training

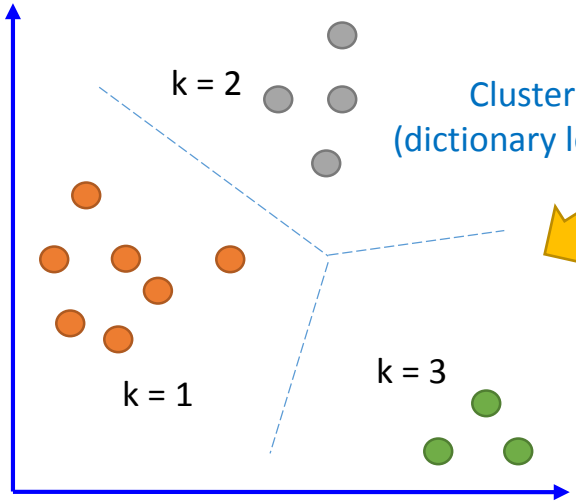
Training images

Interest point detection



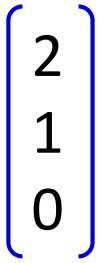


Clustering (dictionary learning)



Quantization (w/ normalization)

Feature Encoding

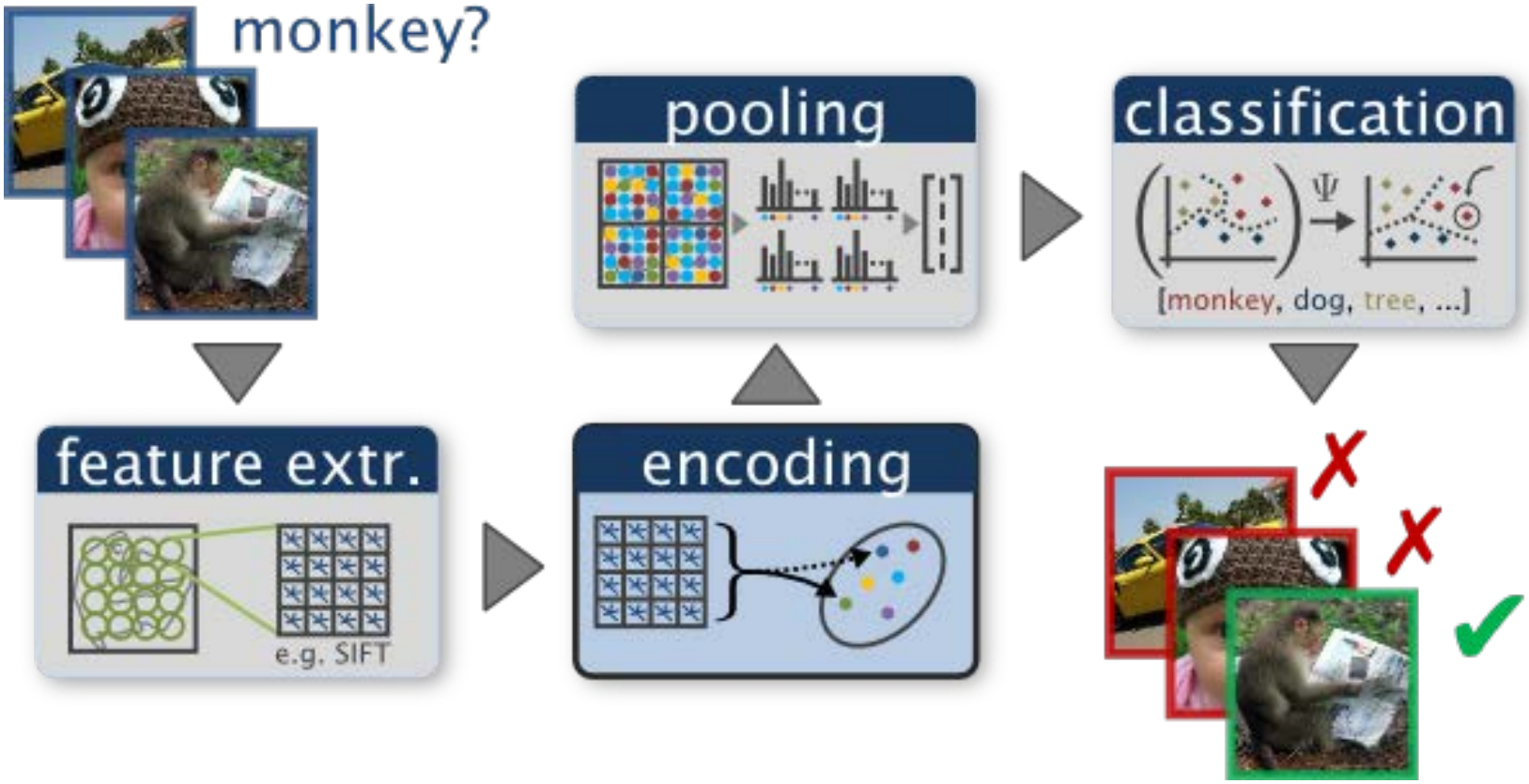


Bag-of-Words for Image Classification

- Testing

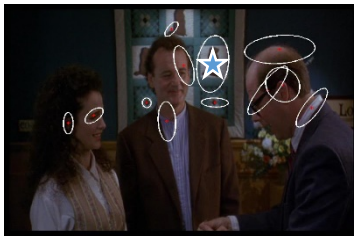
Bag-of-Words for Image Classification

- Overview

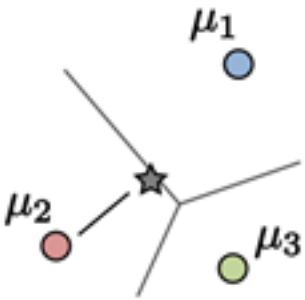


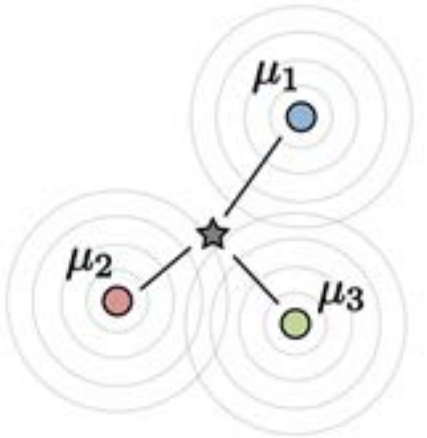
[Chatfieldet al. BMVC 2011]

About Feature Encoding for Bag-of-Words

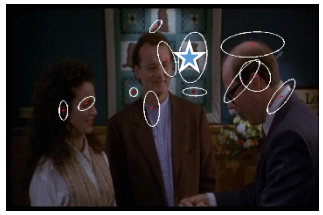


- Hard vs. soft assignments to clusters

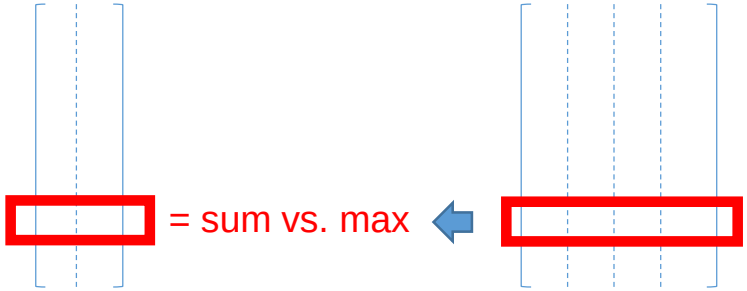


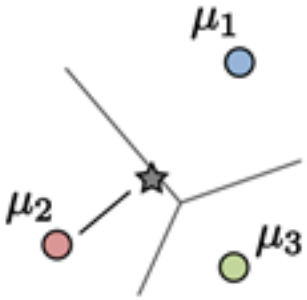


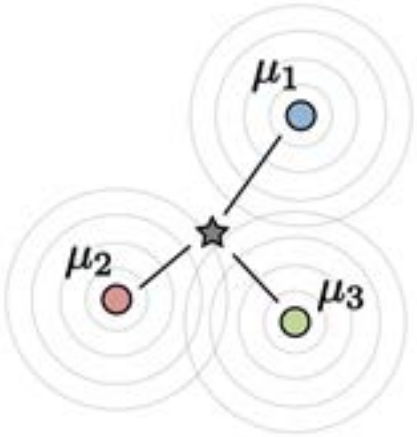
About Feature Encoding for Bag-of-Words



- Sum vs. Max Pooling

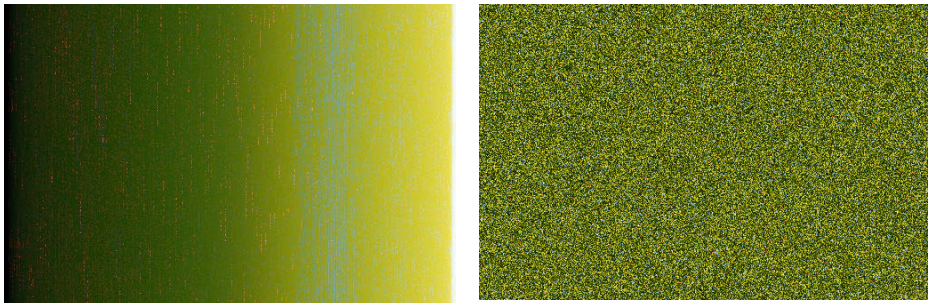


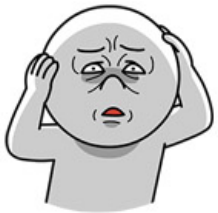


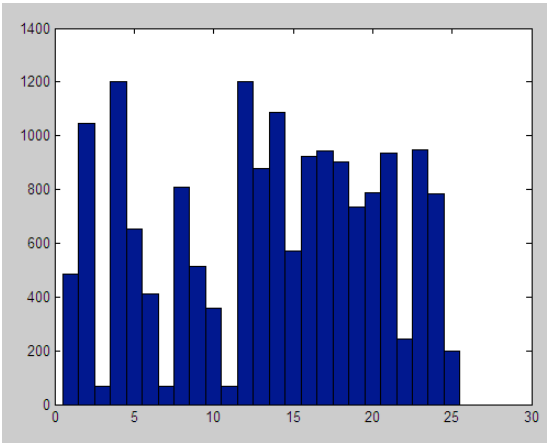


Final Remarks on BoW

- What's the limitation?
  - Loss of…
  - What's the possible solution?

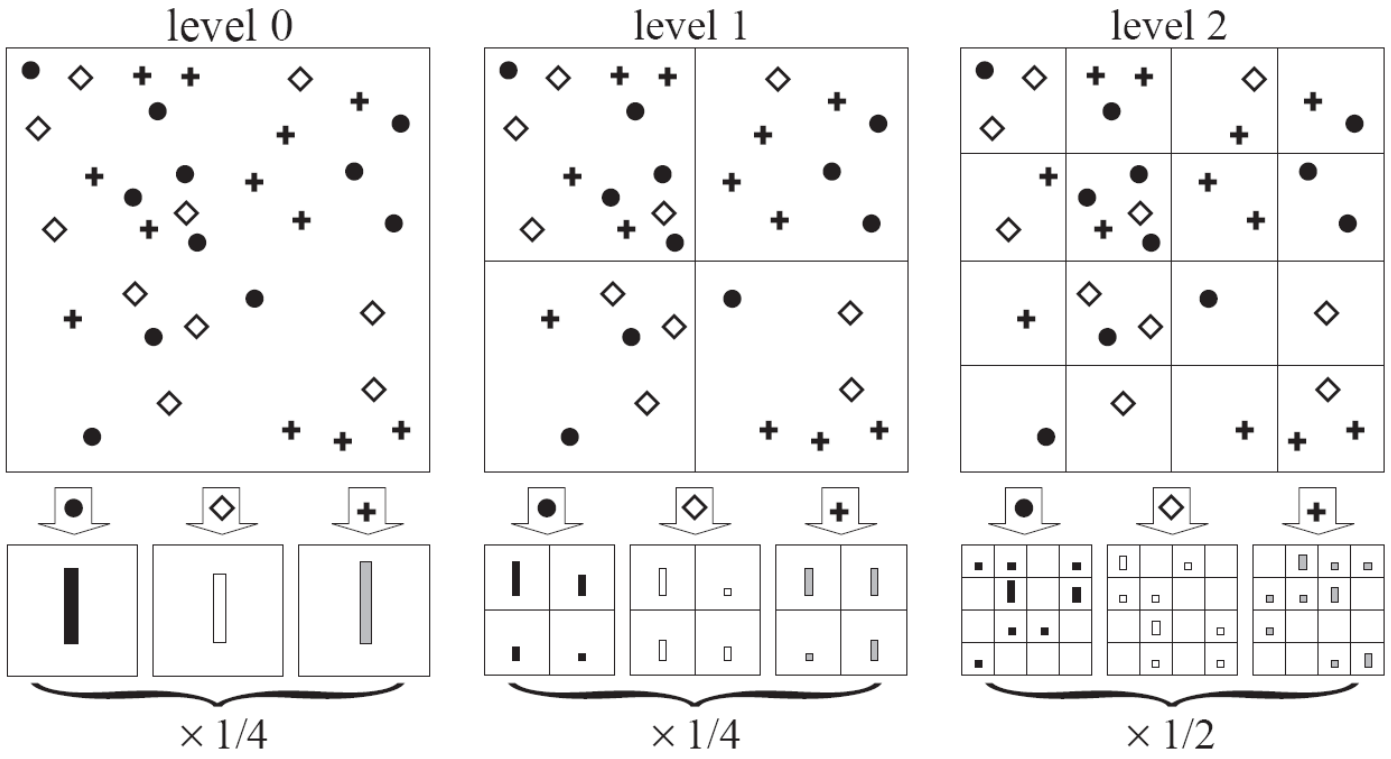






Final Remarks on BoW

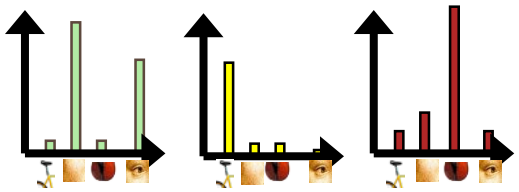
- Spatial pyramid
  - Compute BoW in each spatial grid + concatenation

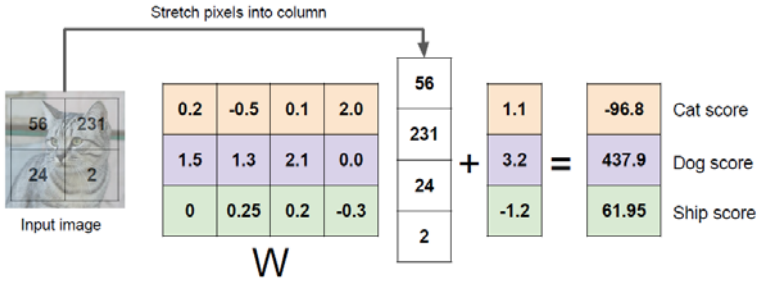


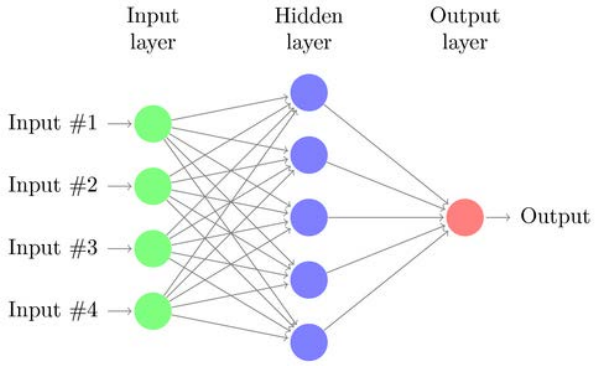
[Lazebnik et al. CVPR 2006]

What's to Be Covered Today…

- General Framework for Visual Classification
  - Bag-of-Words Models as Image Features
- Intro to Neural Networks & CNN
  - Linear Classification
  - Neural Network for Machine Vision
  - Multi-Layer Perceptron
  - Convolutional Neural Networks



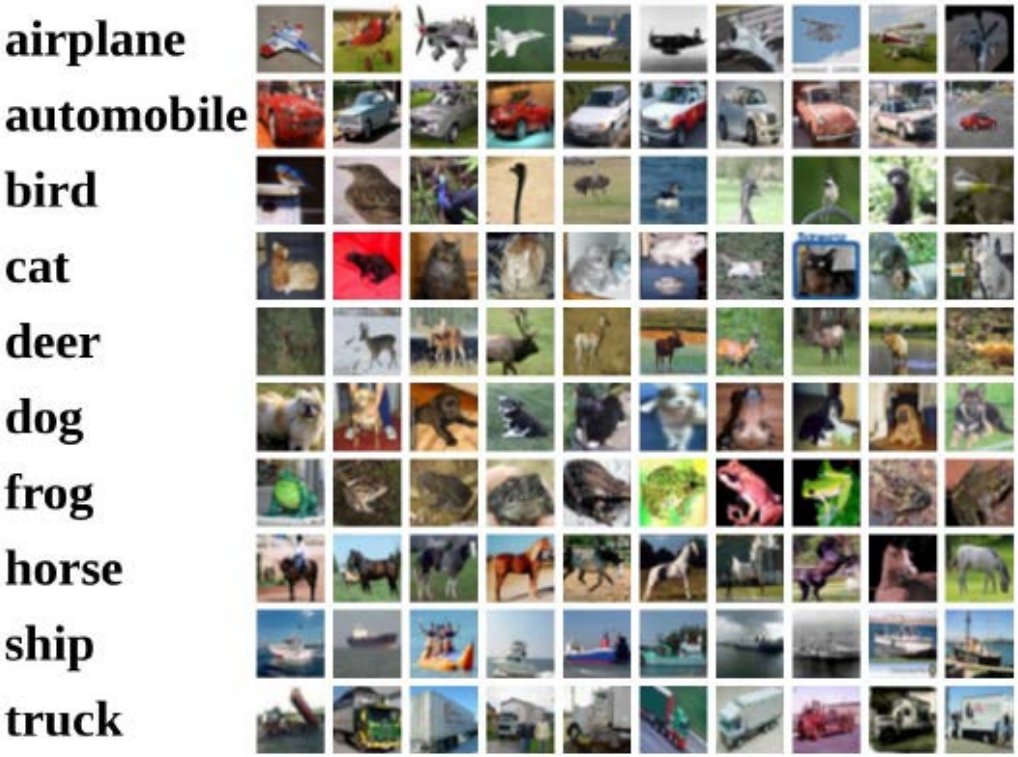




Many slides from Richard Turner, Fei-Fei Li, Yaser Sheikh, Simon Lucey, Kaiming He, and J.-B. Huang

Linear Classification

- Linear Classifier
  - Can be viewed as a parametric approach. Why?
  - Assuming that we need to recognize 10 object categories of interest
  - E.g., CIFAR10 with 50K training & 10K test images of 10 categories.
    And, each image is of size 32 x 32 x 3 pixels.



Linear Classification (cont'd)

- Linear Classifier
  - Can be viewed as a parametric approach. Why?
  - Assuming that we need to recognize 10 object categories of interest (e.g., CIFAR10).
  - Let's take the input image as **x**, and the linear classifier as **W**. We hope to see that **y** = **Wx** + **b** as a 10-dimensional output indicating the score for each class.

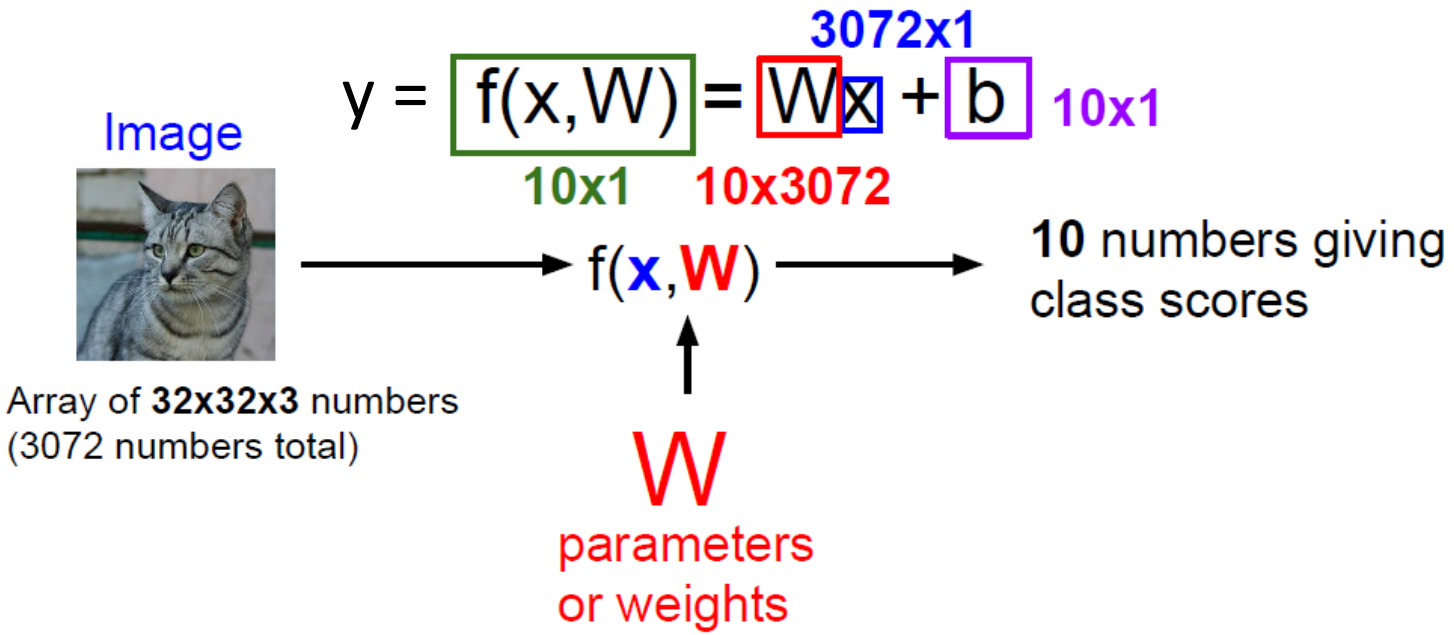


Image credit: Stanford CS231n

Linear Classification (cont'd)

- Linear Classifier
  - Can be viewed as a parametric approach. Why?
  - Assuming that we need to recognize 10 object categories of interest (e.g., CIFAR10).
  - Let's take the input image as **x**, and the linear classifier as **W**. We hope to see that **y** = **Wx** + **b** as a 10-dimensional output indicating the score for each class.
  - Take an image with 2 x 2 pixels & 3 classes of interest as example:
    we need to learn linear transformation/classifer **W** and bias **b**,
    so that desirable outputs **y** = **Wx** + **b** can be expected.

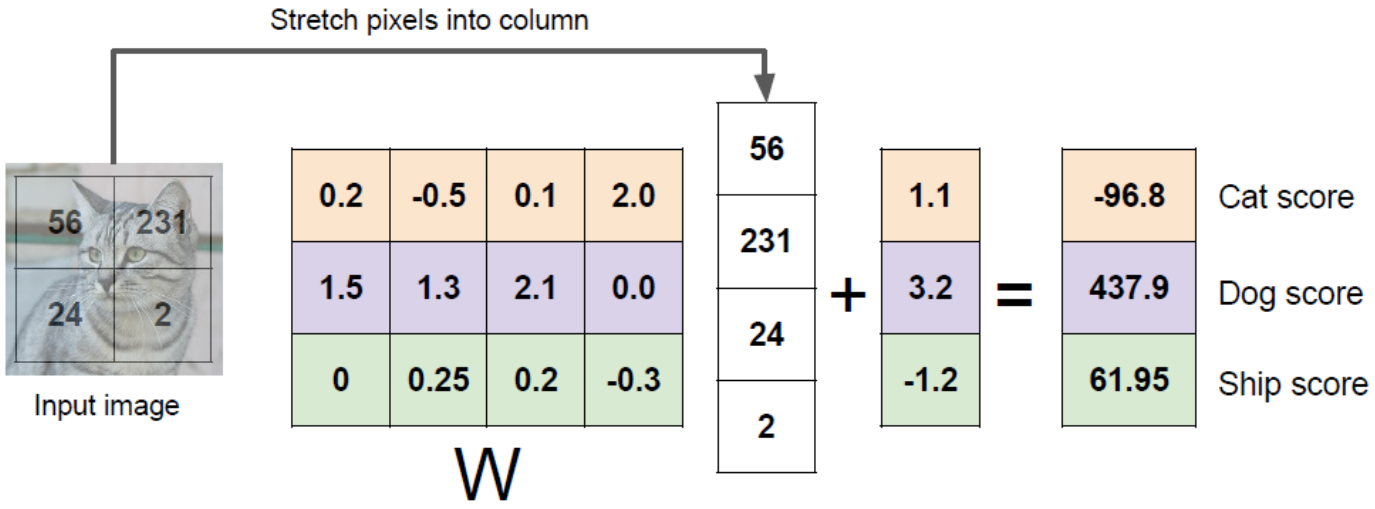
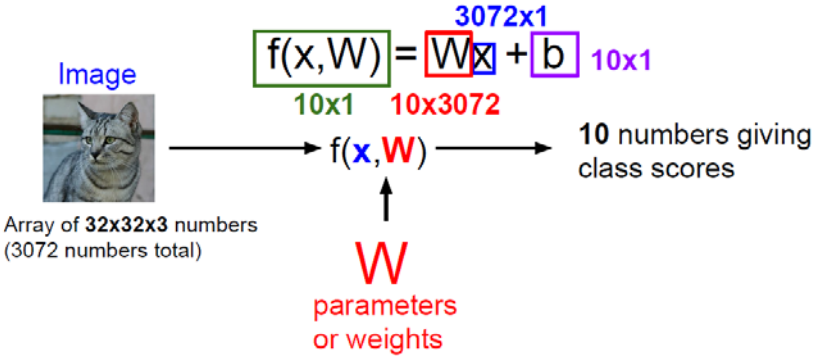
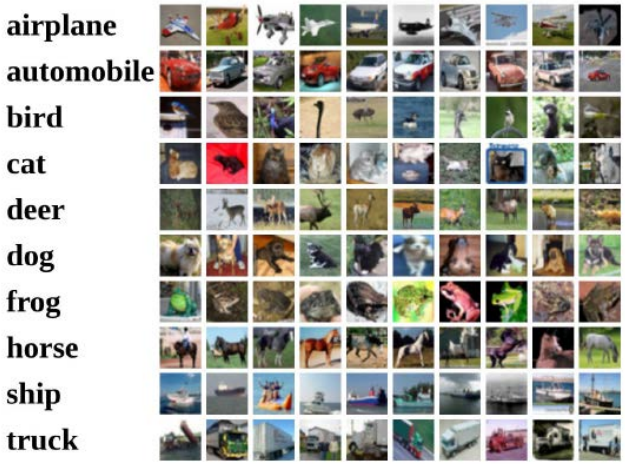


Image credit: Stanford CS231n

Some Remarks



- Interpreting **y** = **Wx** + **b**
  - What can we say about the learned **W**?
  - The weights in **W** are trained by observing training data **X** and their ground truth **Y**.
  - Each column in **W** can be viewed as an exemplar of the corresponding class.
  - Thus, **Wx** basically performs inner product (or correlation) between the input **x** and the exemplar of each class. (Signal & Systems!)



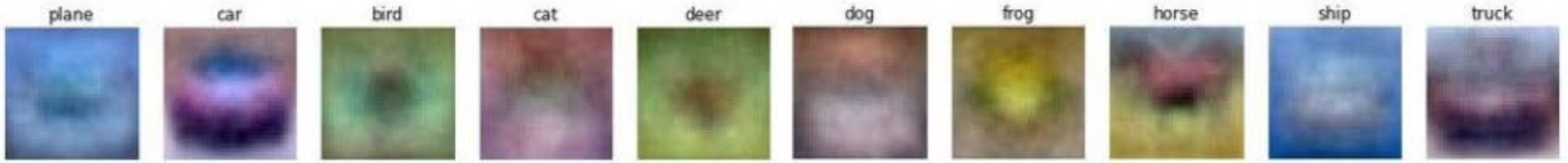


Image credit: Stanford CS231n

Linear Classification

- Remarks
  - Starting points for many multi-class or complex/nonlinear classifier
  - How to determine a proper loss function for matching **y** and **Wx+b**, and thus how to learn the model **W** (including the bias **b**), are the keys to the learning of an effective classification model.

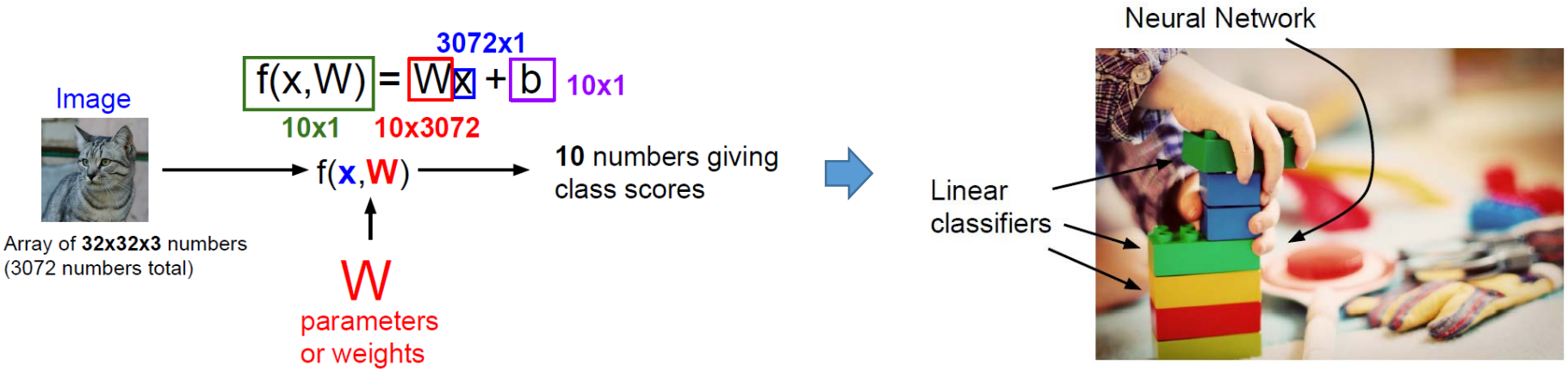
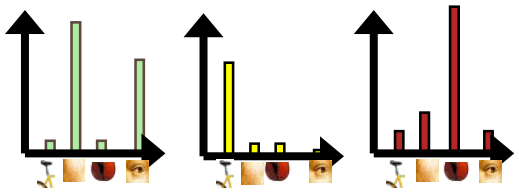
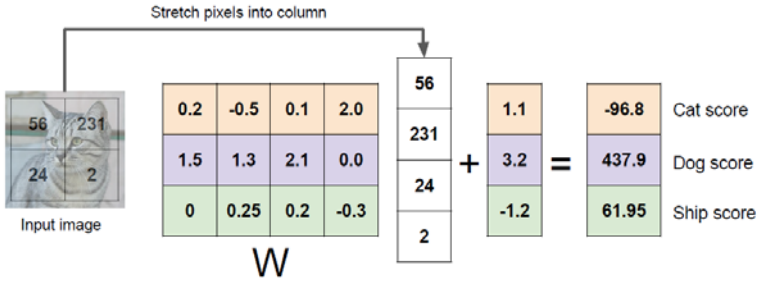


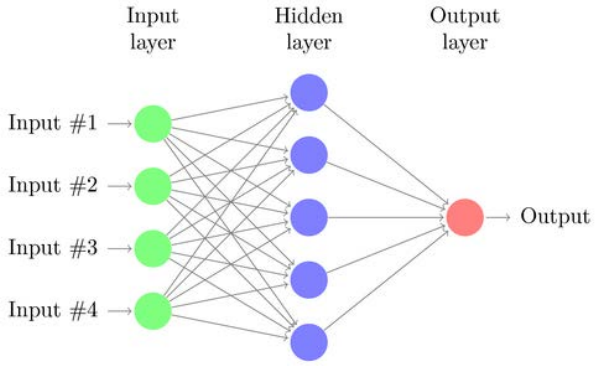
Image credit: Stanford CS231n

What's to Be Covered Today…

- General Framework for Visual Classification
  - Bag-of-Words Models as Image Features
- Intro to Neural Networks & CNN
  - Linear Classification
  - Neural Network for Machine Vision
  - Multi-Layer Perceptron
  - Convolutional Neural Networks

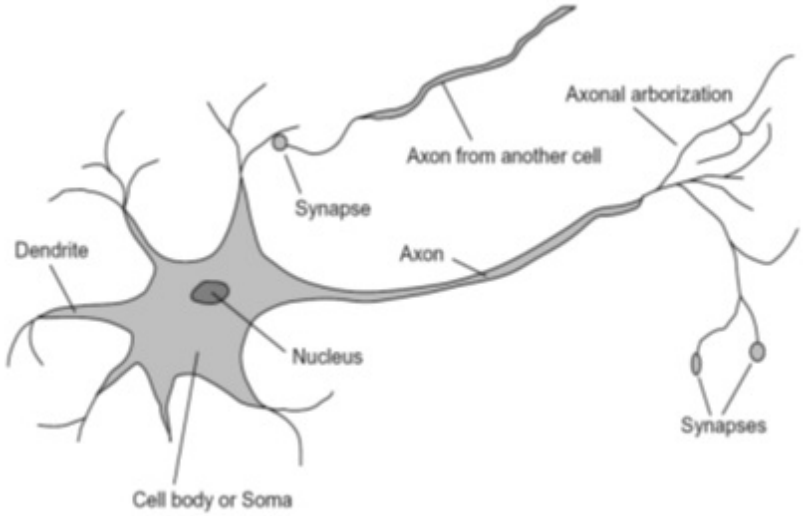




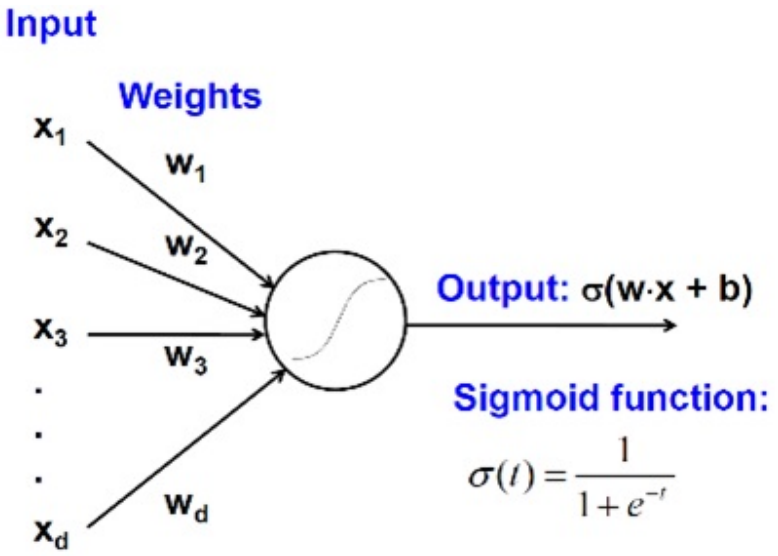


Many slides from Richard Turner, Fei-Fei Li, Yaser Sheikh, Simon Lucey, Kaiming He, and J.-B. Huang

Biological neuron and Perceptrons

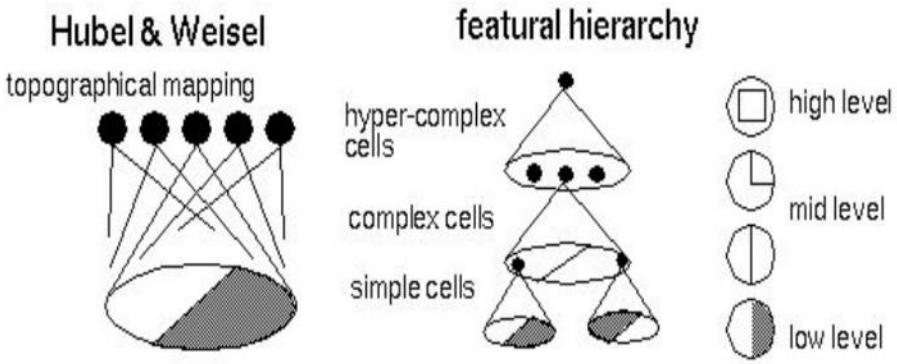


A biological neuron

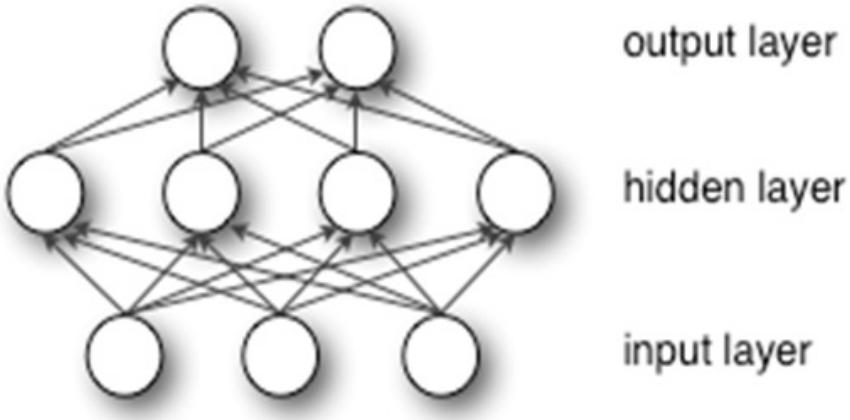


An artificial neuron (Perceptron)
- a linear classifier

Hubel/Wiesel Architecture and Multi-layer Neural Network



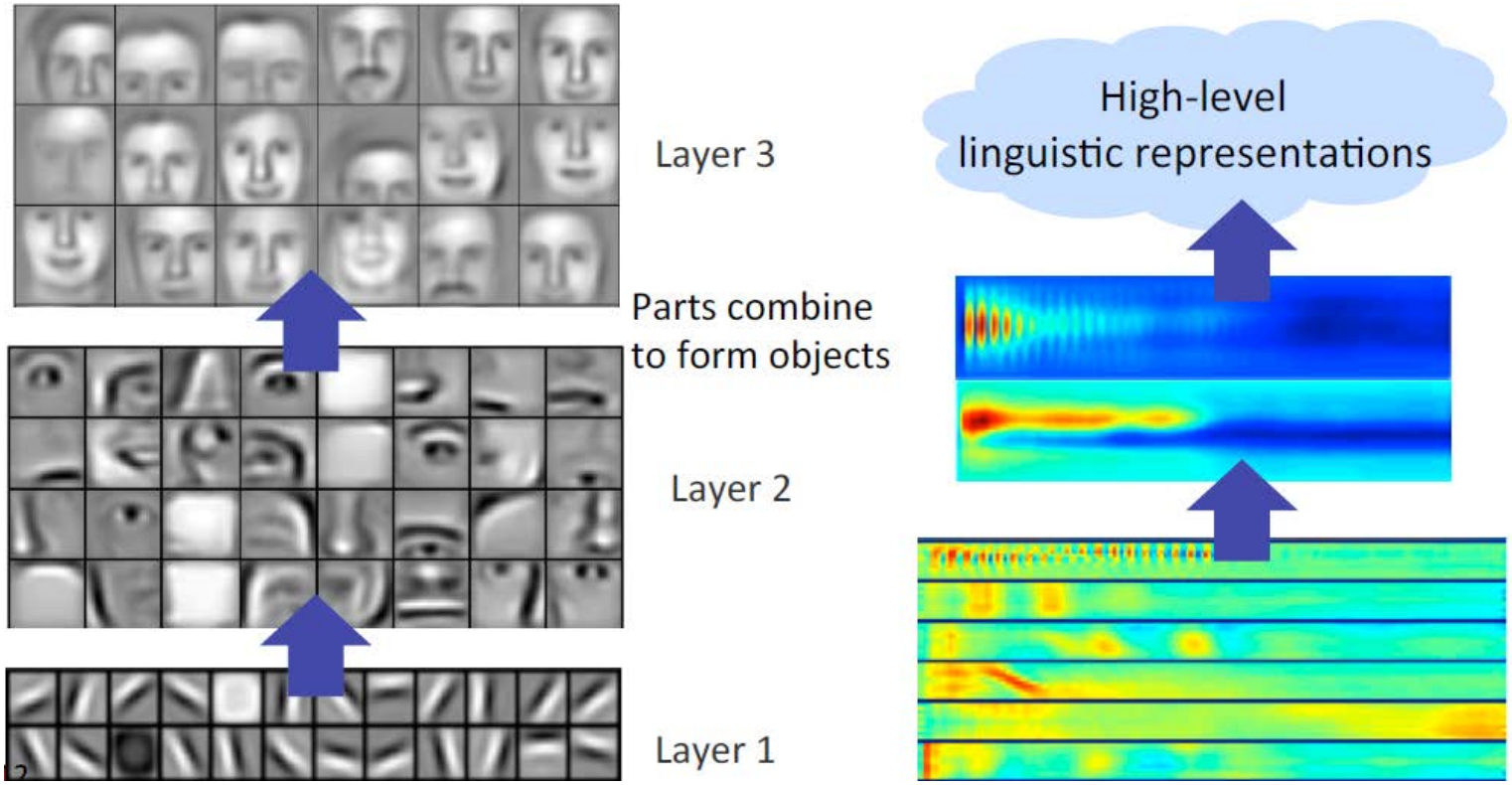
Hubel and Weisel's architecture



Multi-layer Neural Network
- A *non-linear* classifier

Hierarchical Learning

- Successive model layers learn deeper intermediate representations.



Revisit of Linear Classification

- Linear Classifier
  - Can be viewed as a parametric approach. Why?
  - Assuming that we need to recognize 10 object categories of interest (e.g., CIFAR10).
  - Let's take the input image as **x**, and the linear classifier as **W**. We hope to see that **y** = **Wx** + **b** as a 10-dimensional output indicating the score for each class.

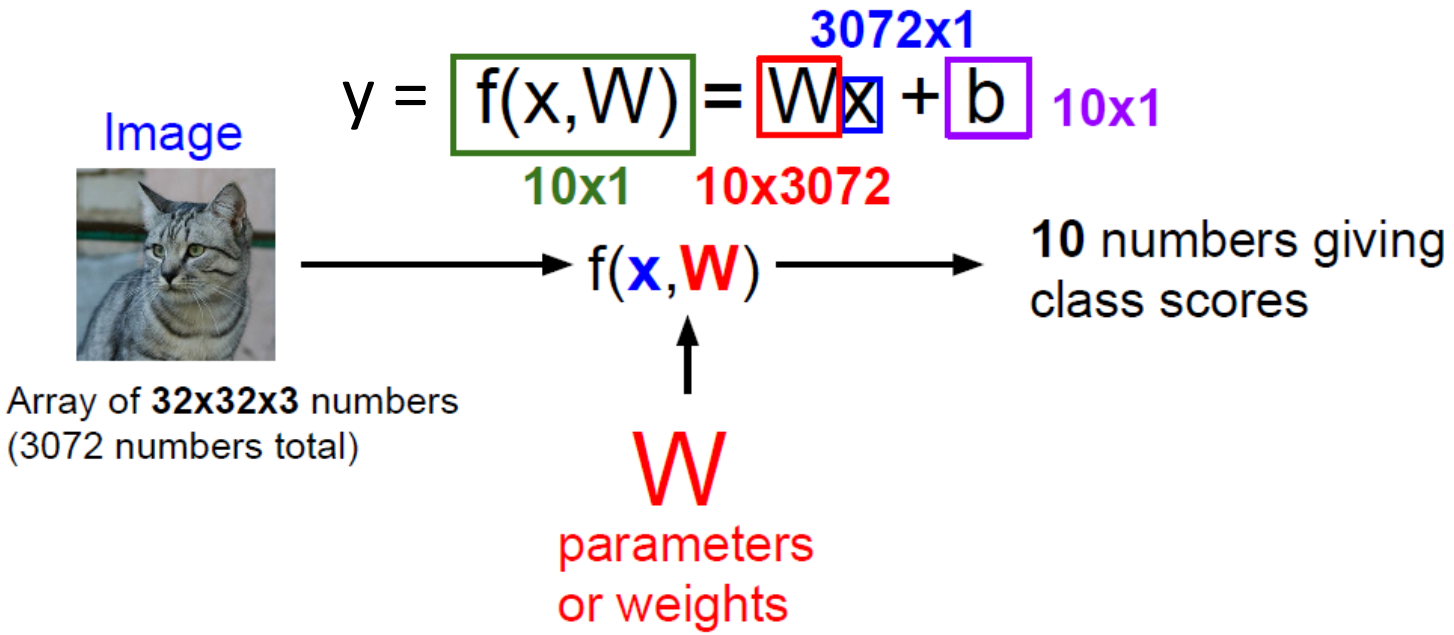
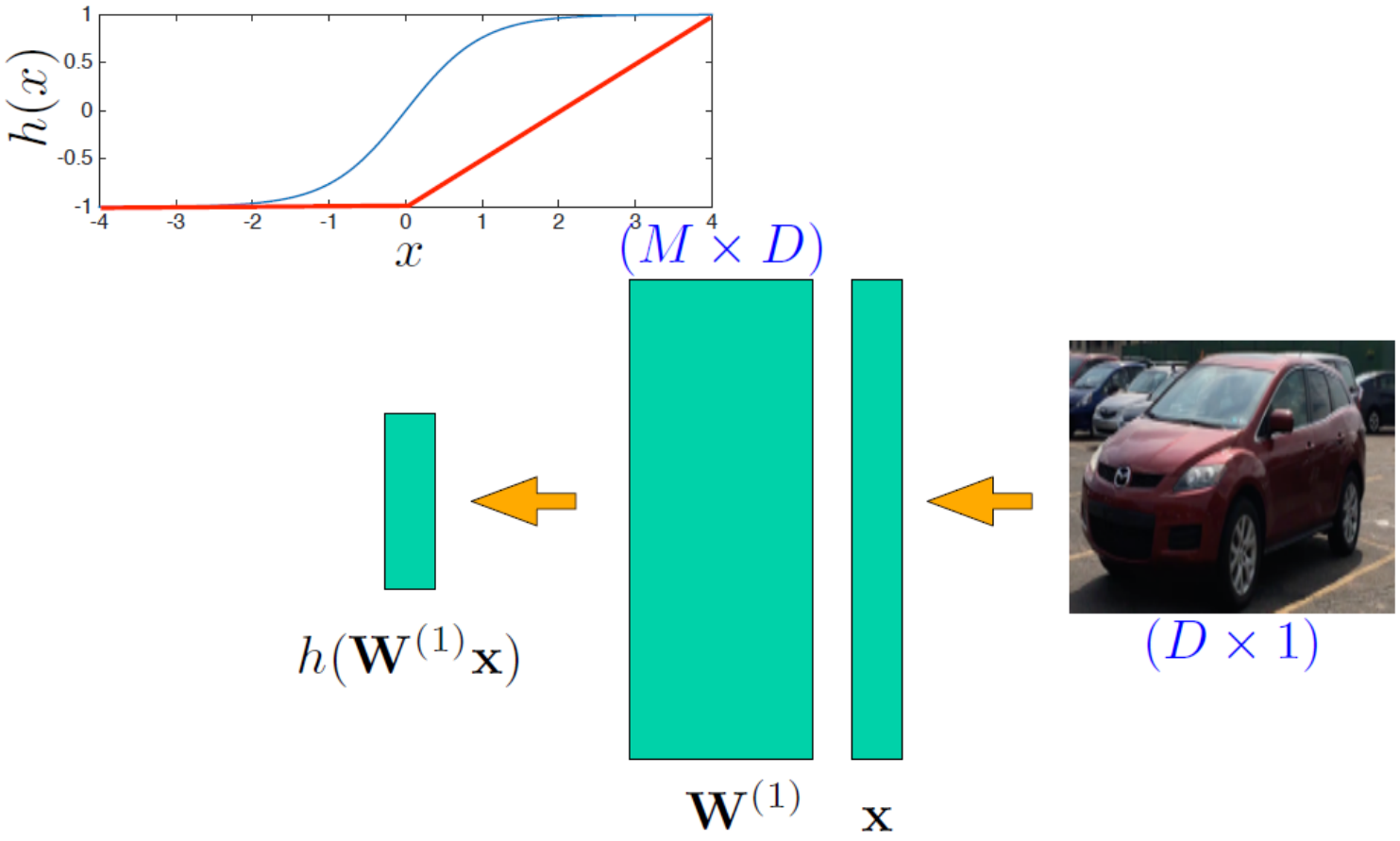
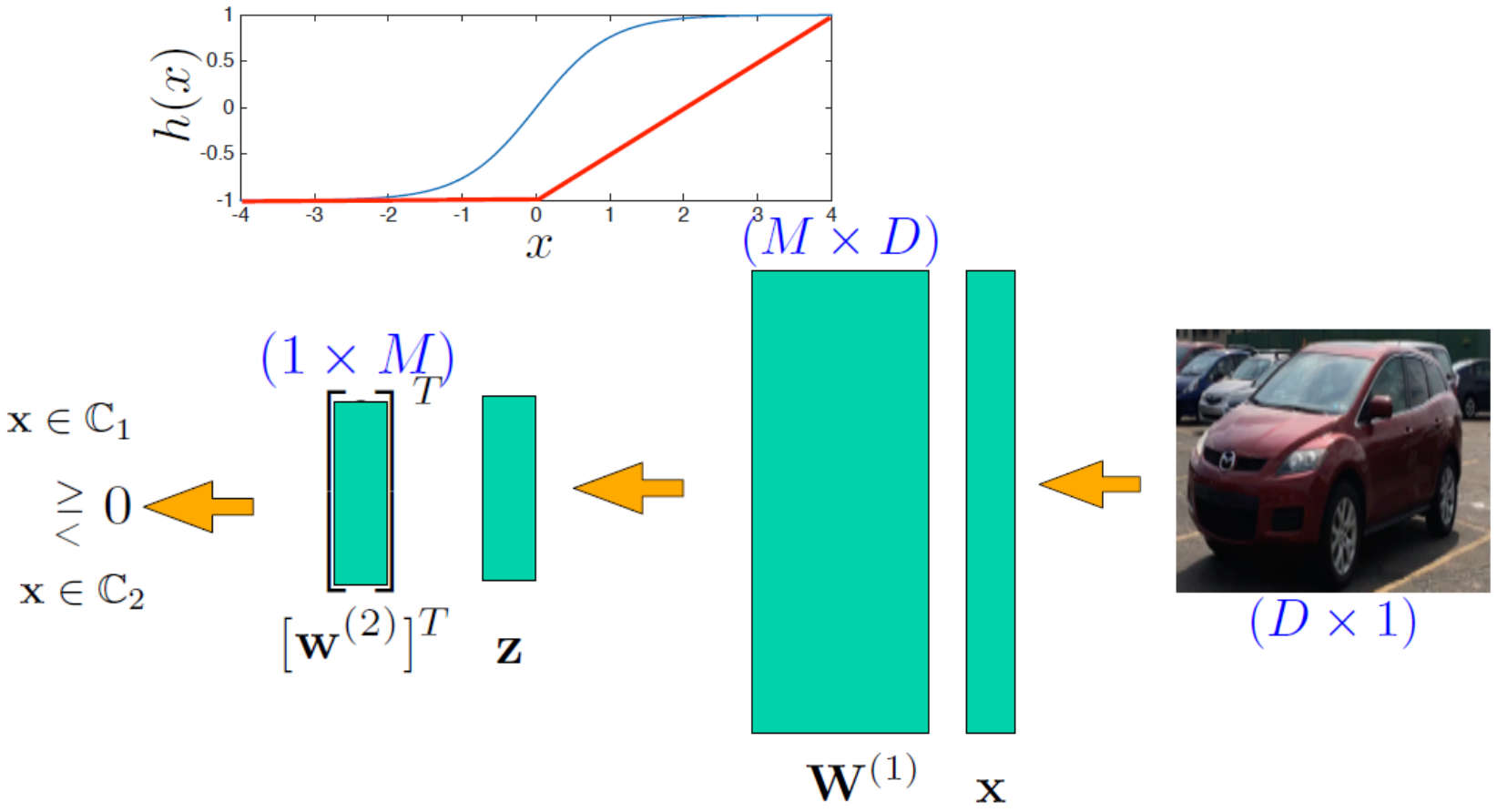


Image credit: Stanford CS231n

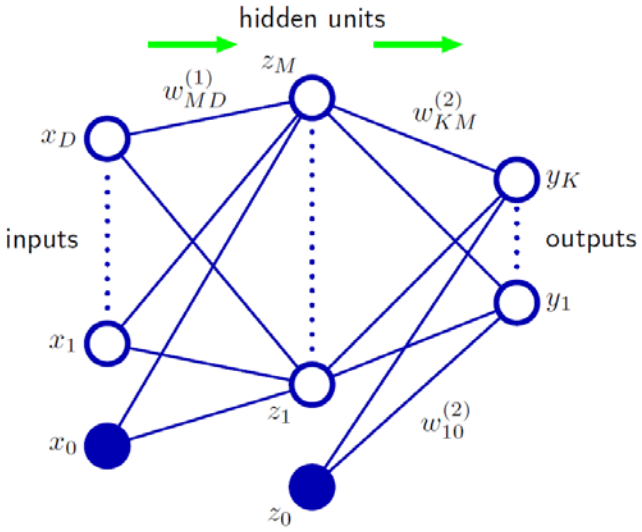
Multi-Layer Perceptron: A Nonlinear Classifier



Multi-Layer Perceptron: A Nonlinear Classifier (cont'd)



Layer 1 in MLP



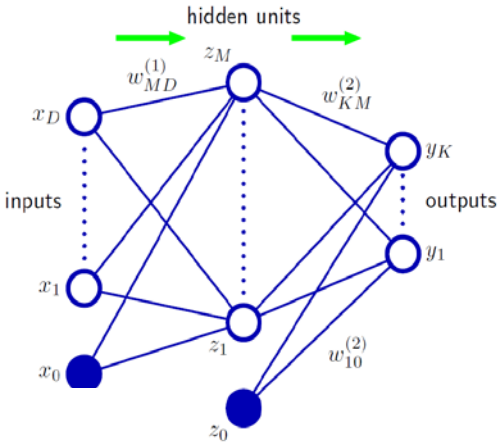
$$\mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_M \end{bmatrix} \leftarrow \begin{bmatrix} h[\mathbf{x}^T \mathbf{w}_1^{(1)}] \\ \vdots \\ h[\mathbf{x}^T \mathbf{w}_M^{(1)}] \end{bmatrix}$$

$h() =$ non-linear function

$[\mathbf{w}_1^{(1)}, \ldots, \mathbf{w}_M^{(1)}] =$ 1st layer's $D \times M$ weights

$\mathbf{x} = D \times 1$ raw input

Layer 2 in MLP



$\mathbf{x} \in \mathbb{R}^D$

Layer 1

$\mathbf{z} \in \mathbb{R}^M$

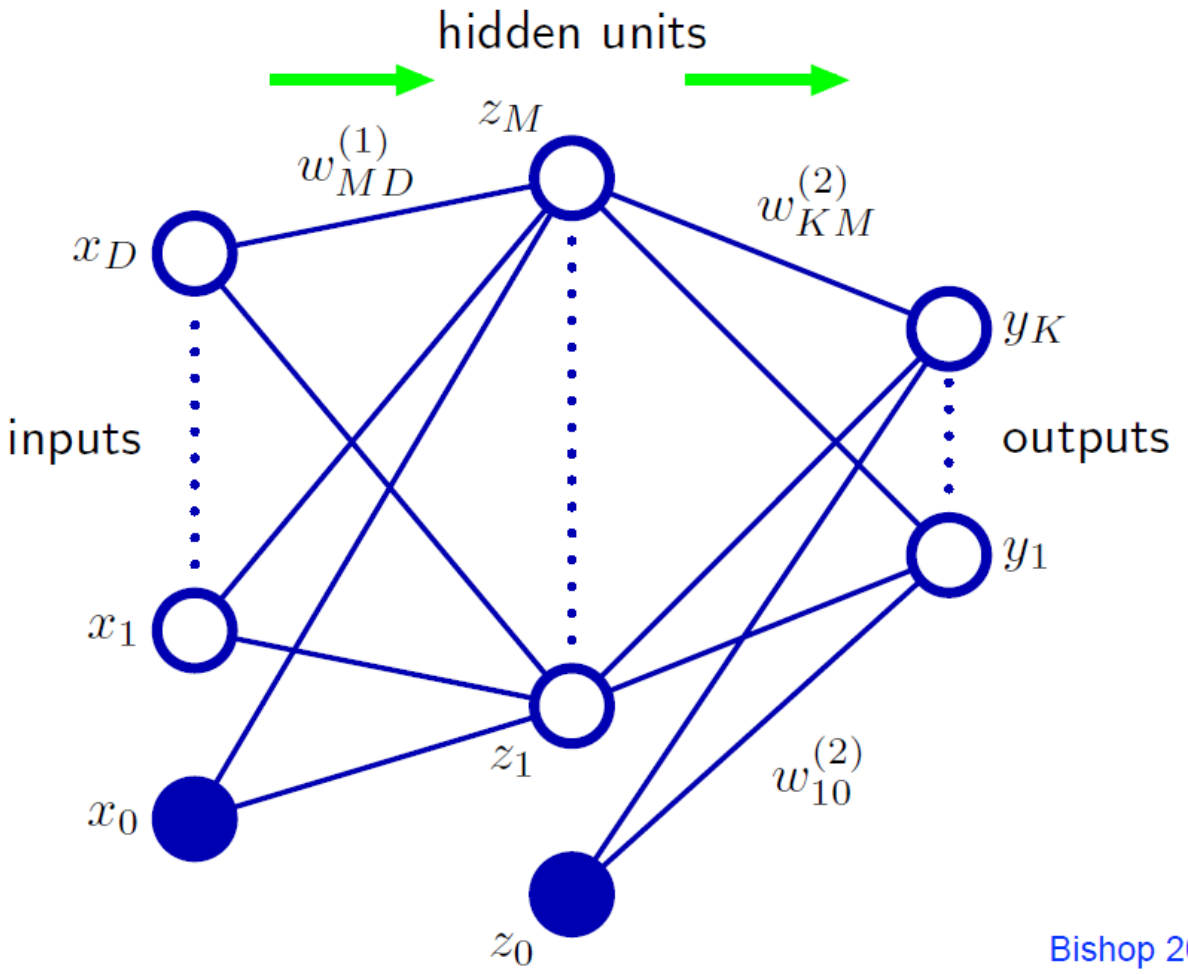
$$\mathbf{z} \in \mathbb{C}_1$$

$$\mathbf{z}^T \mathbf{w}^{(2)} \gtrless 0$$

$$\mathbf{z} \in \mathbb{C}_2$$

$\mathbf{z} = M \times 1$ output of layer 1
$\mathbf{w}^{(2)} = 2$nd layer's $M \times 1$ weight vector

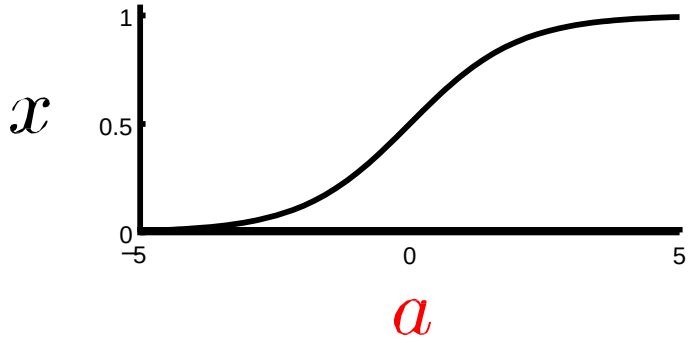
Multi-Layer Perceptron: A Nonlinear Classifier (cont'd)

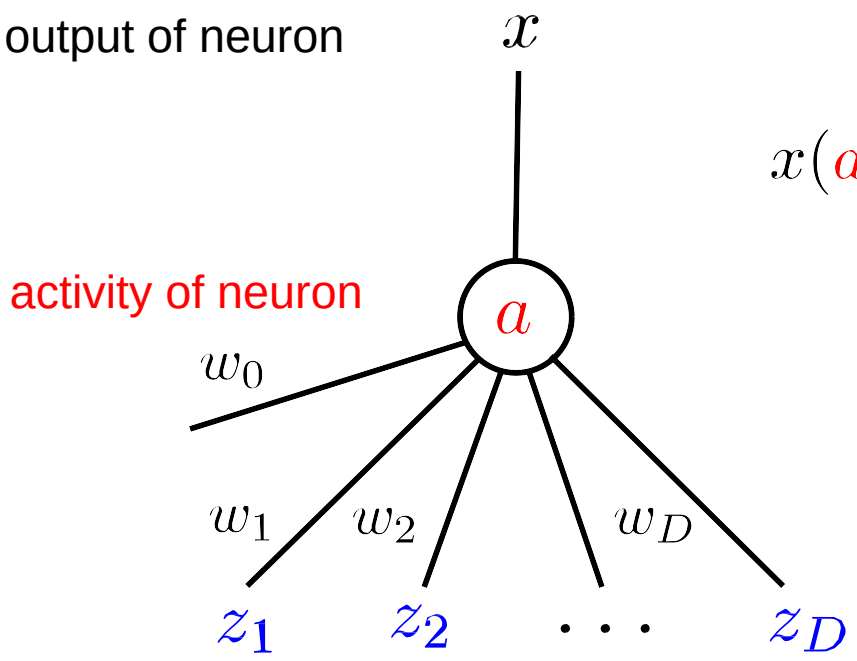


Bishop 2006

Let's Get a Closer Look…

- A single neuron

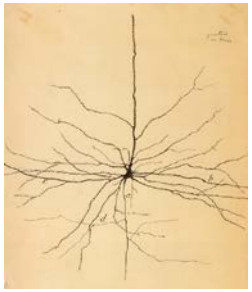




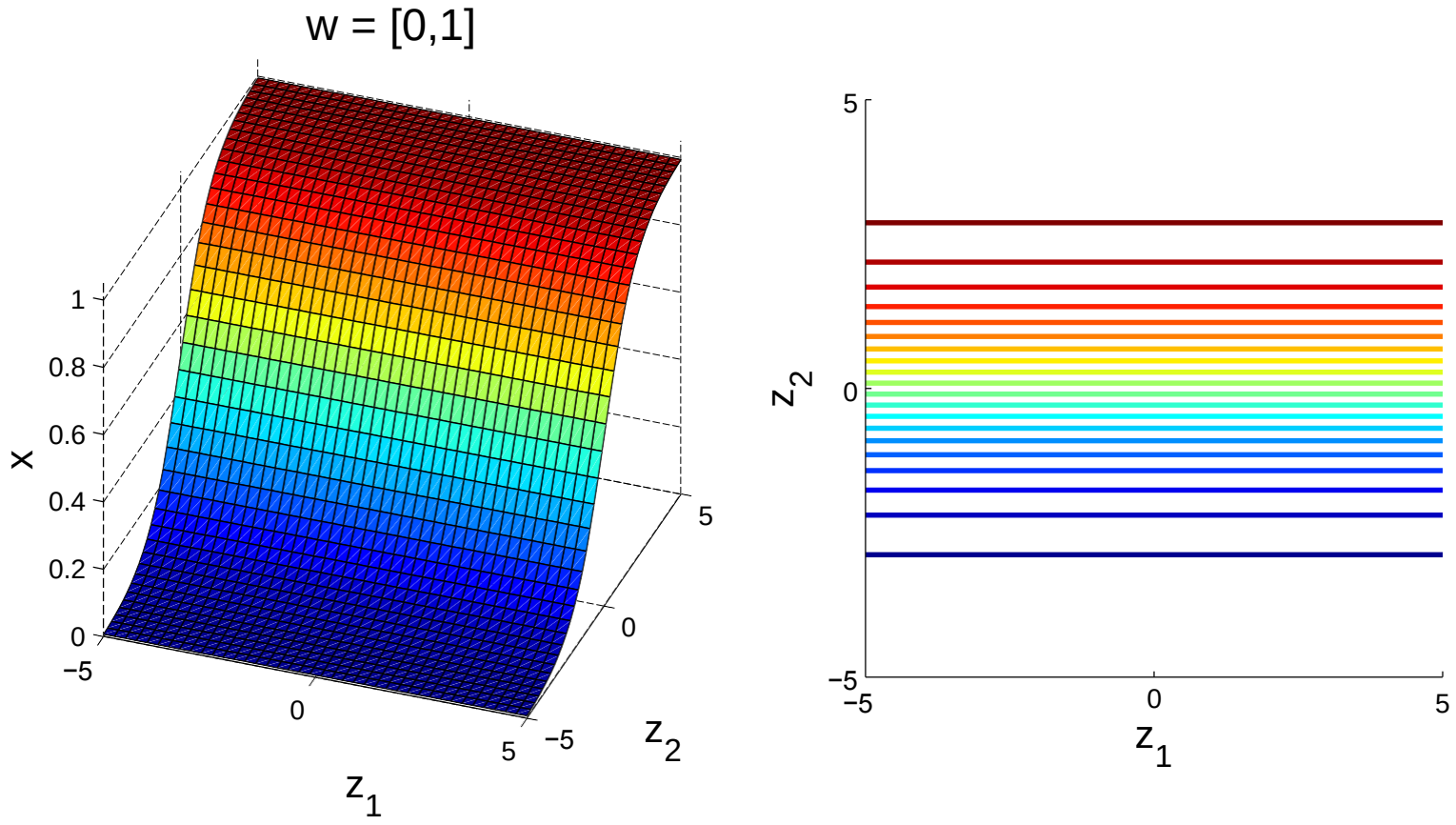
$$x(a) = \frac{1}{1+\exp(-a)} \qquad x \in (0,1)$$

$$a = w_0 + \sum_{d=1}^{D} w_d z_d$$

$$= \sum_{d=0}^{D} w_d z_d$$

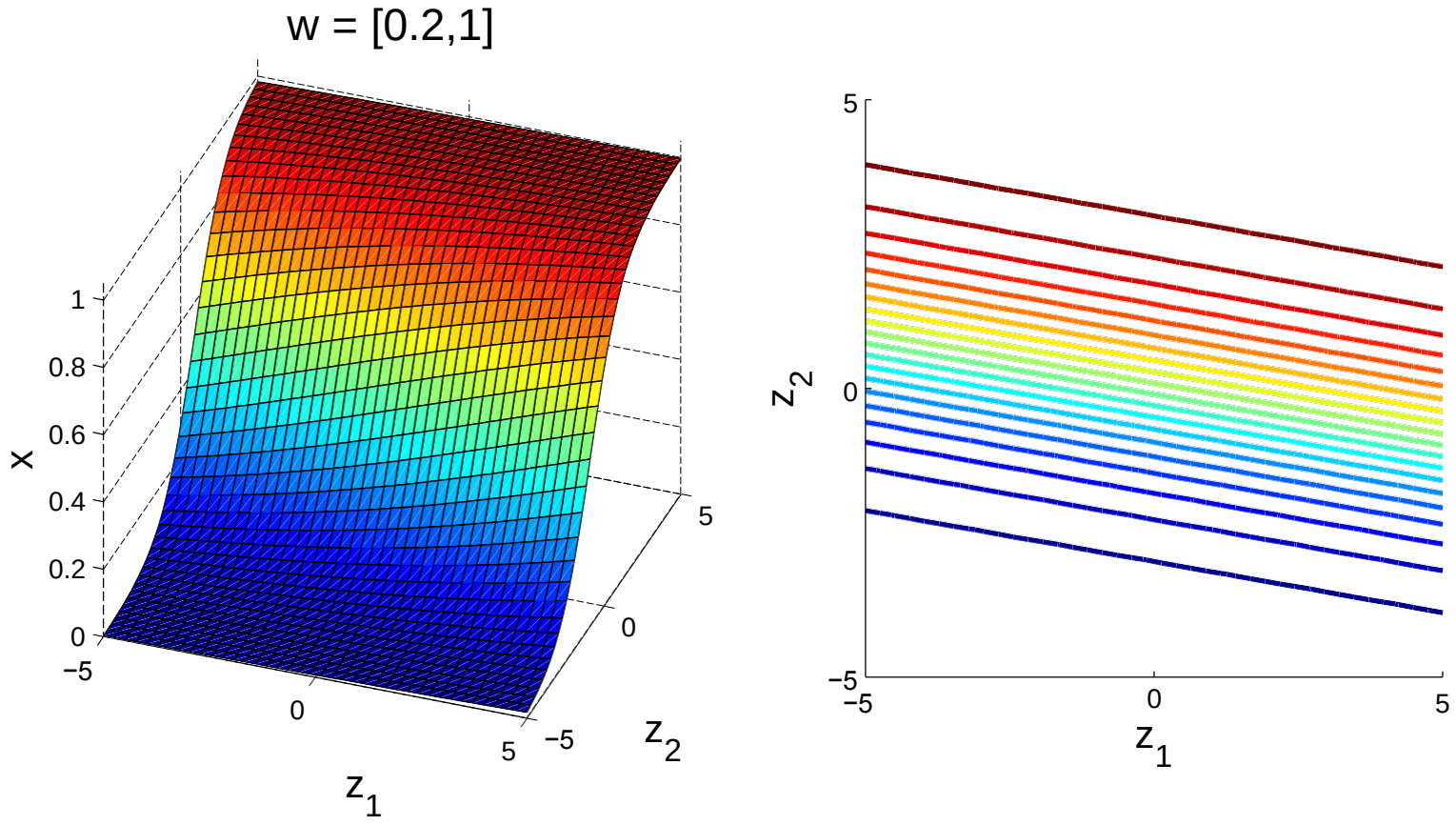


Input-Output Function of a Single Neuron



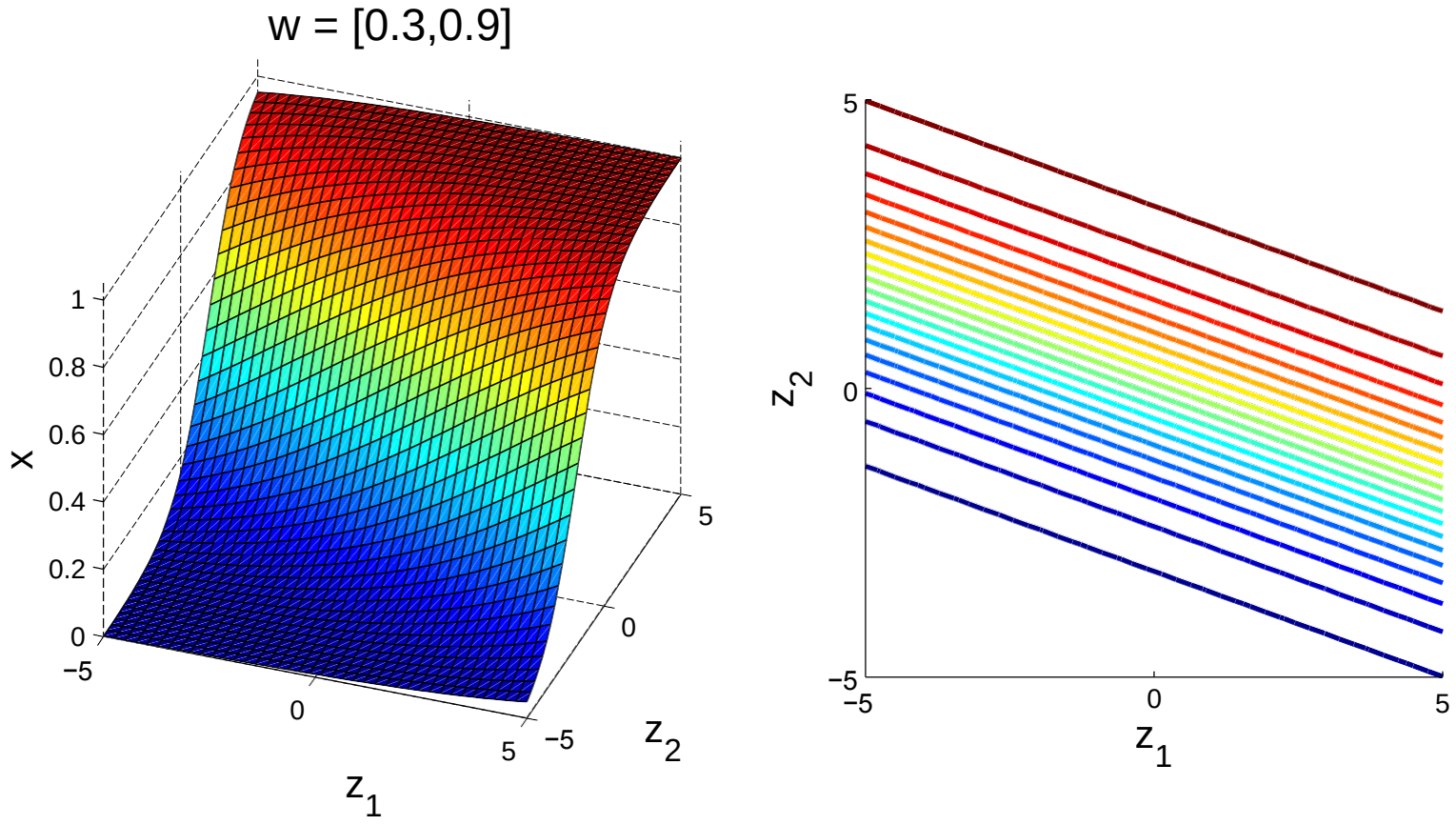
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron



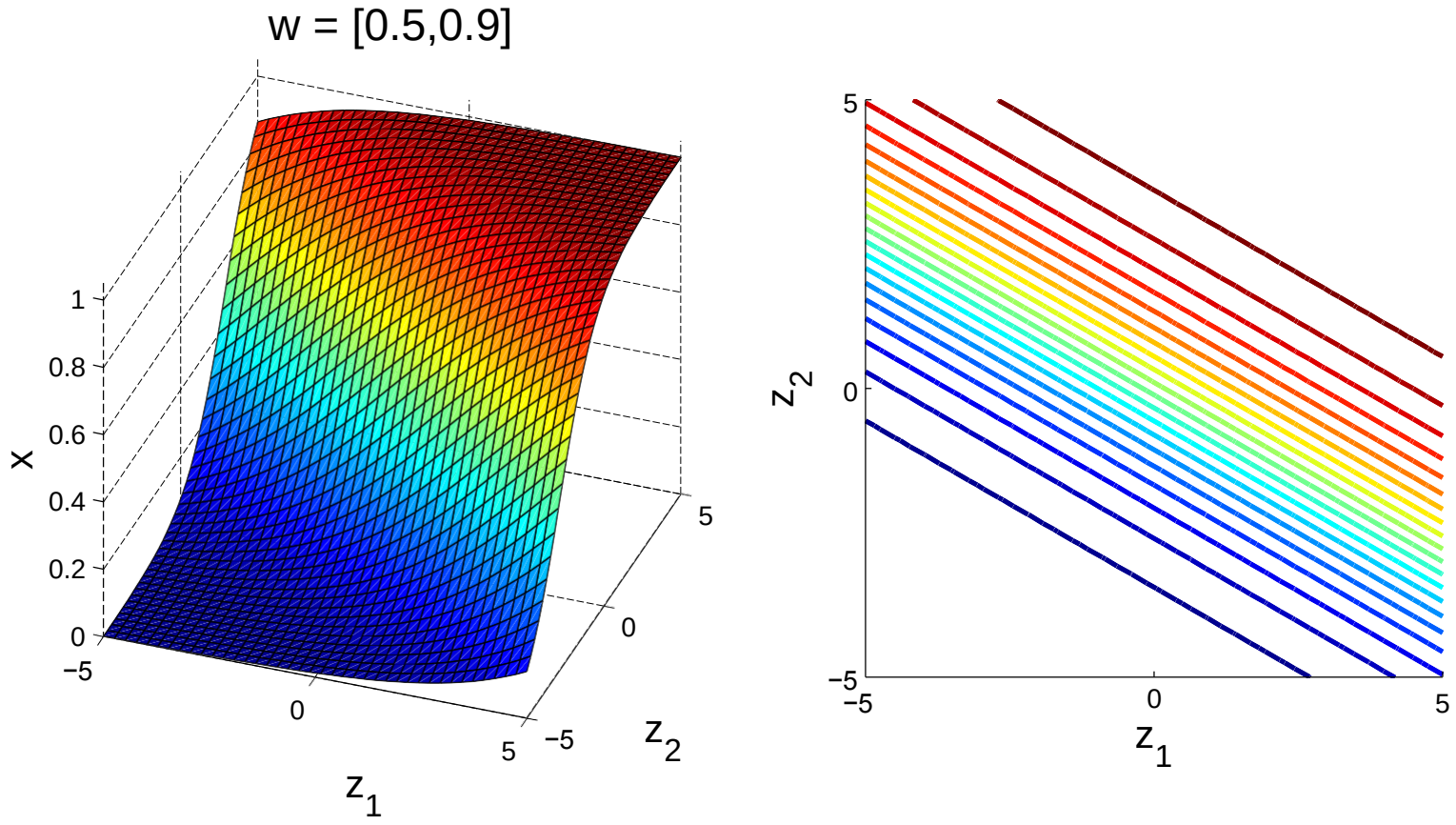
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron



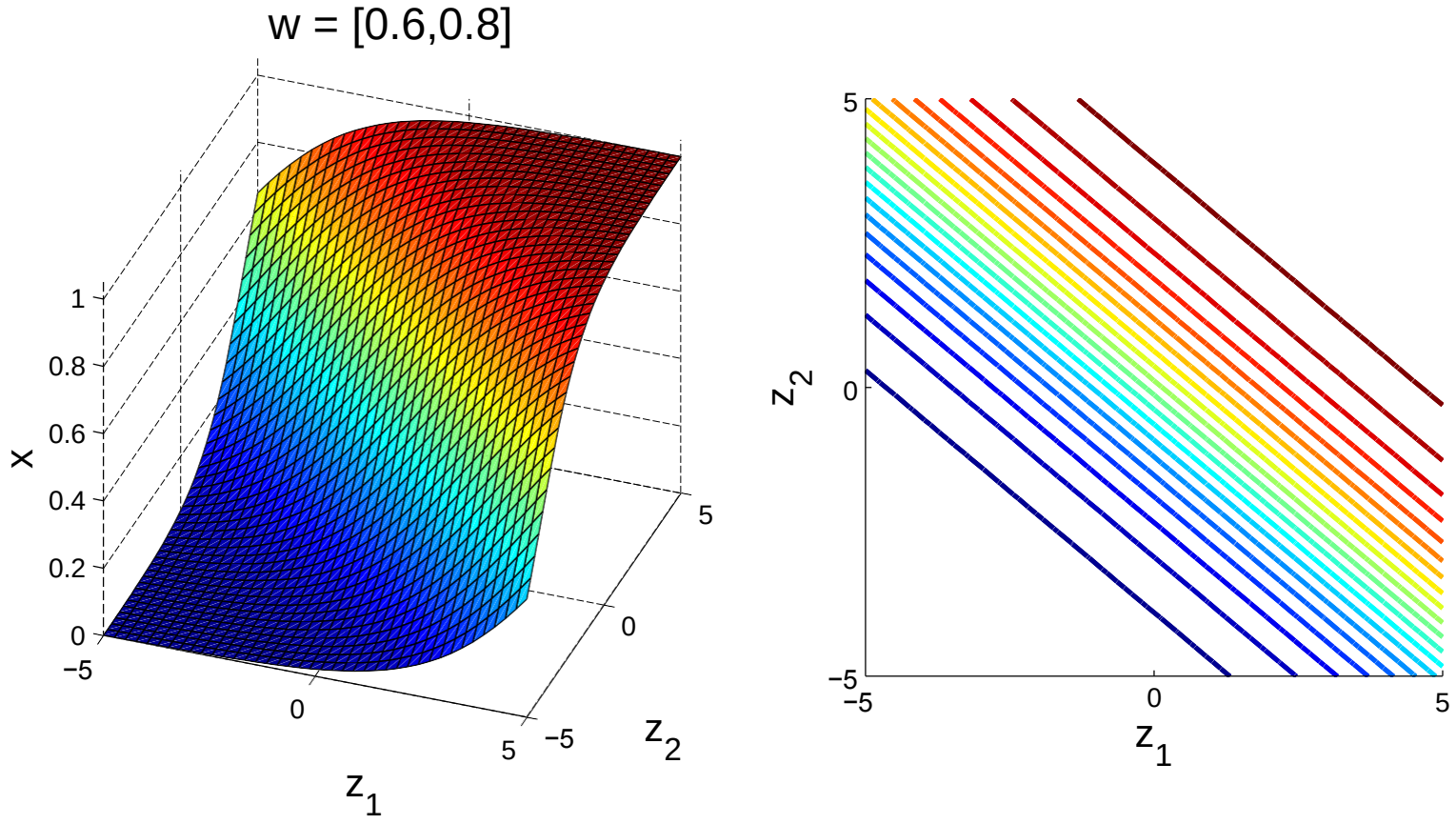
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron



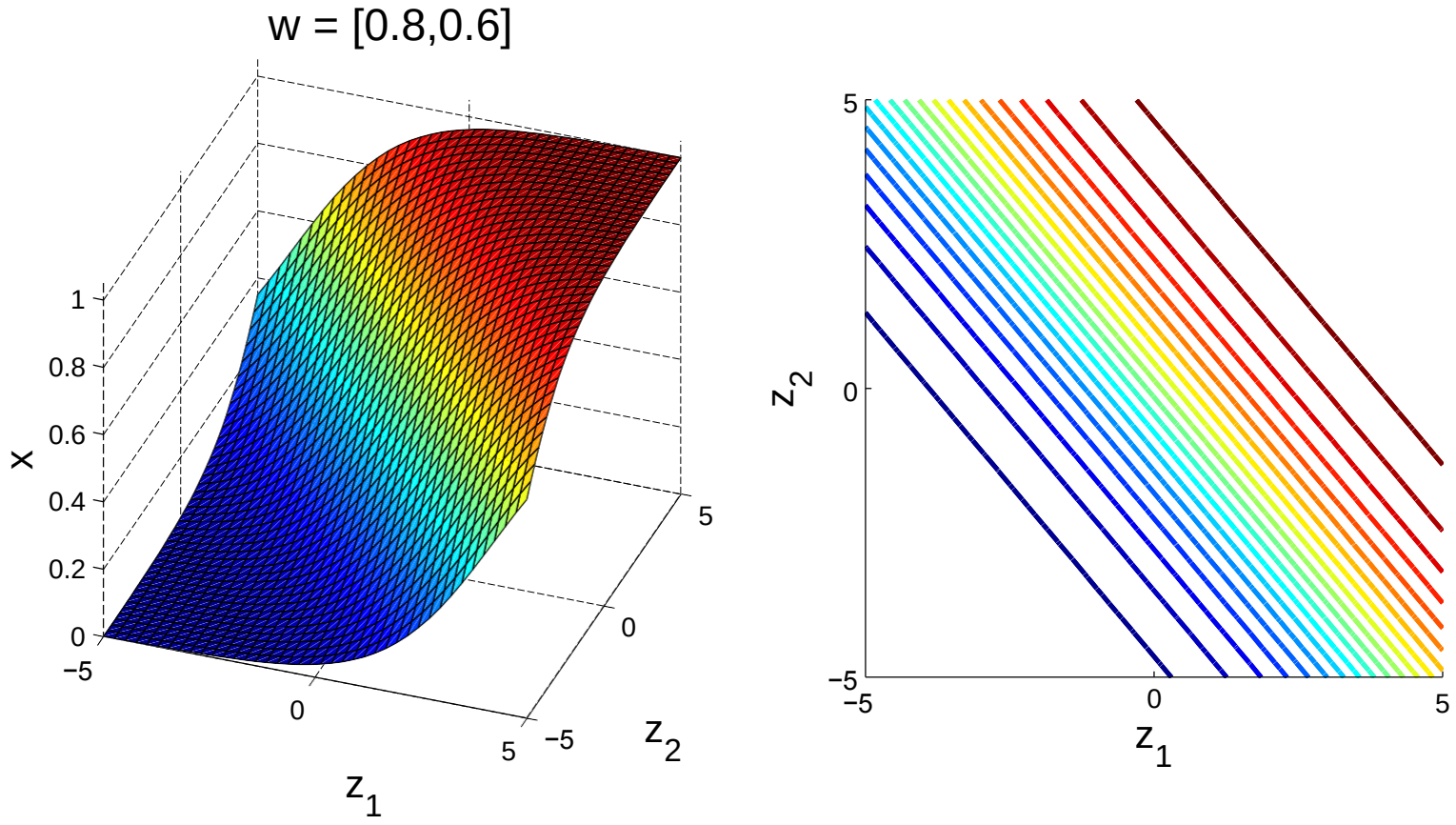
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron



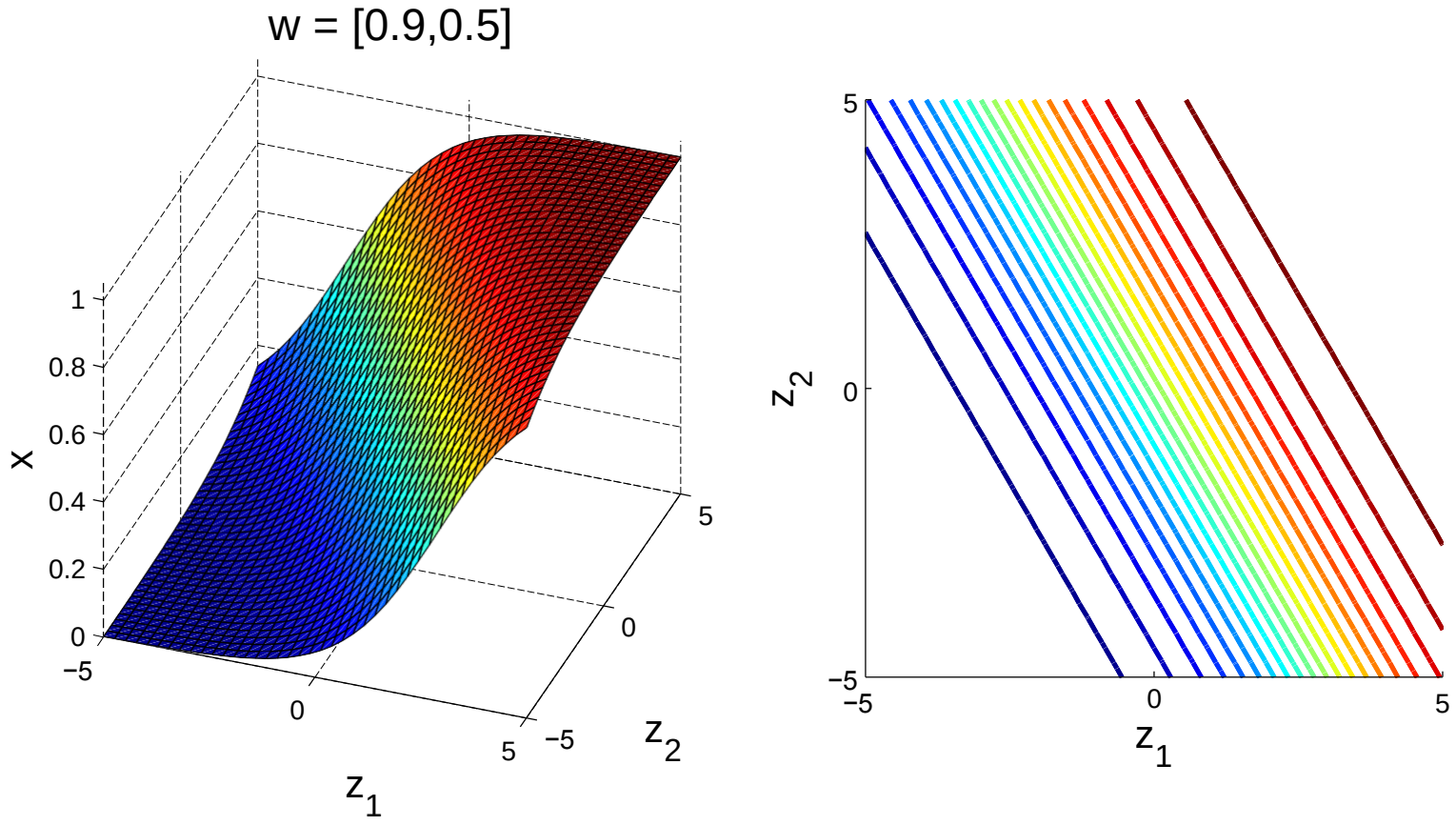
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron



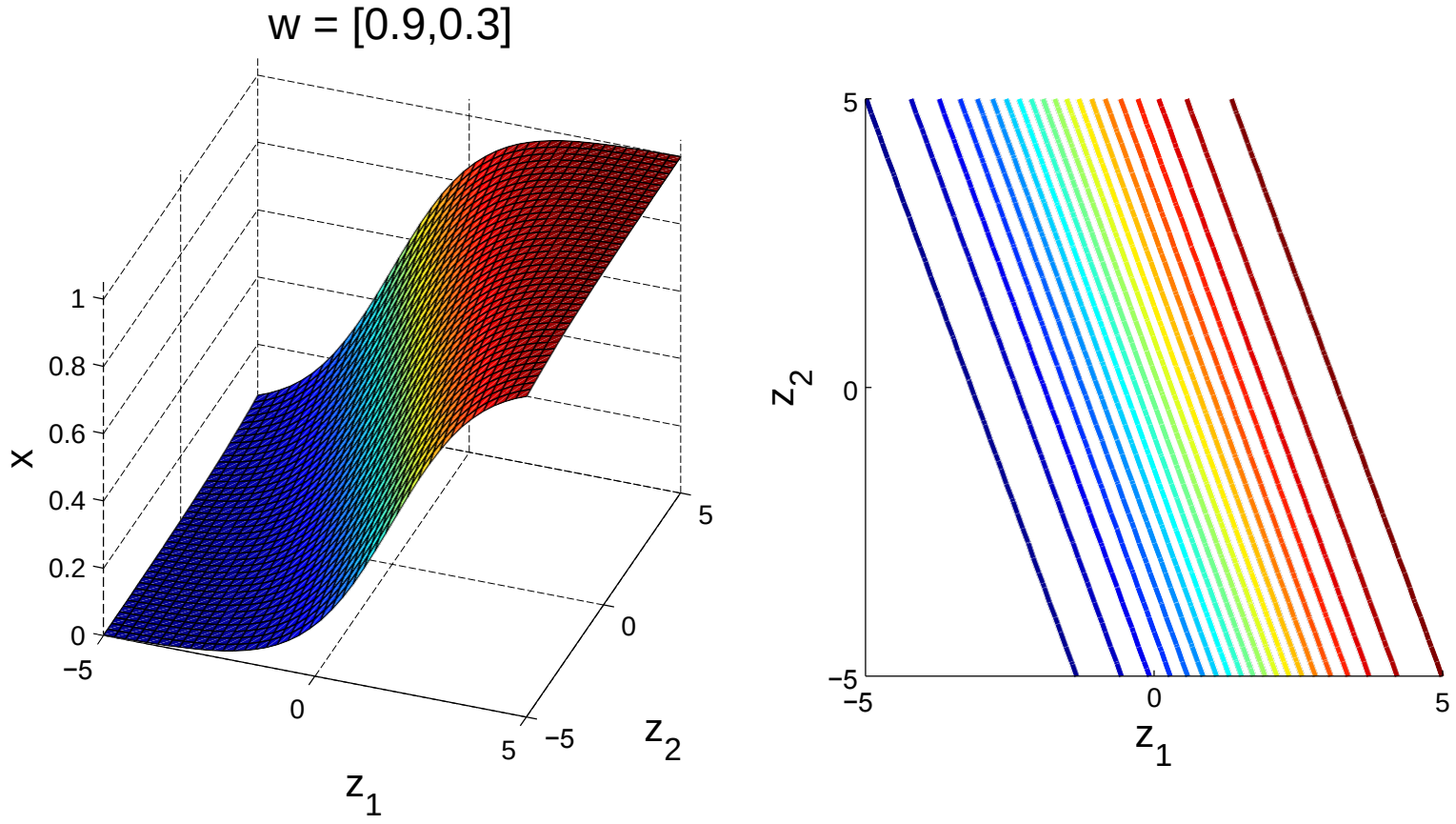
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron



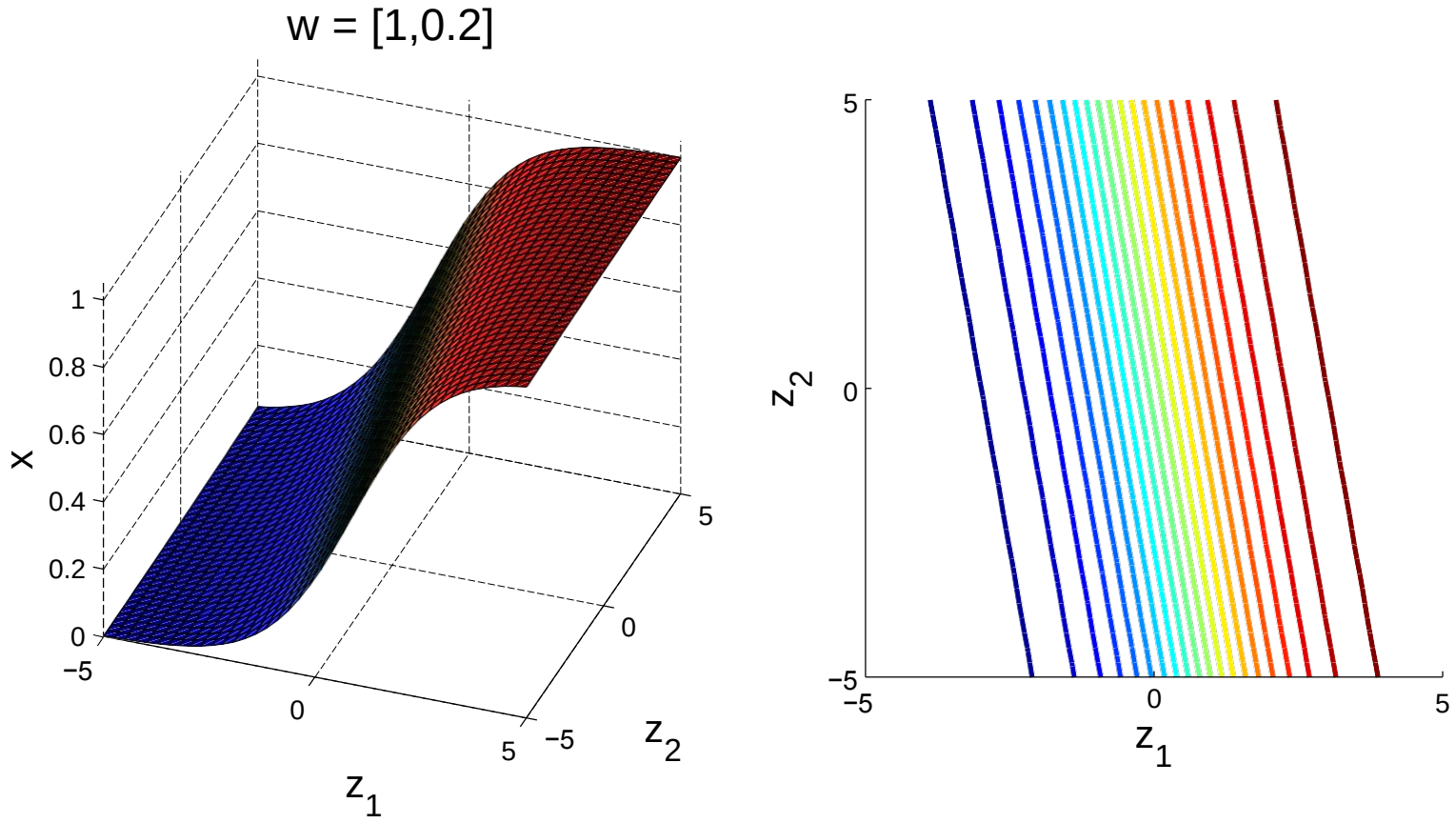
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron



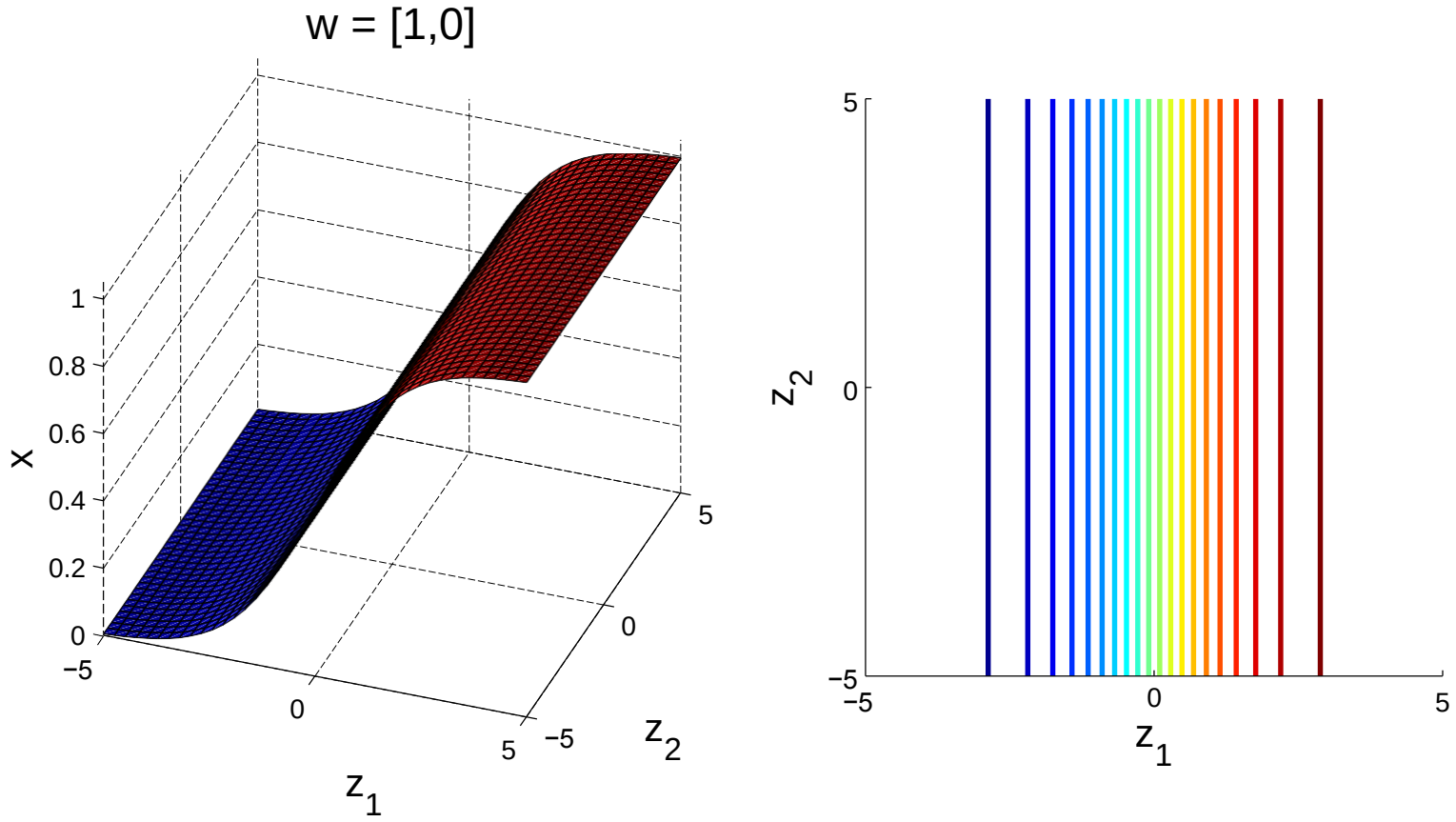
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron



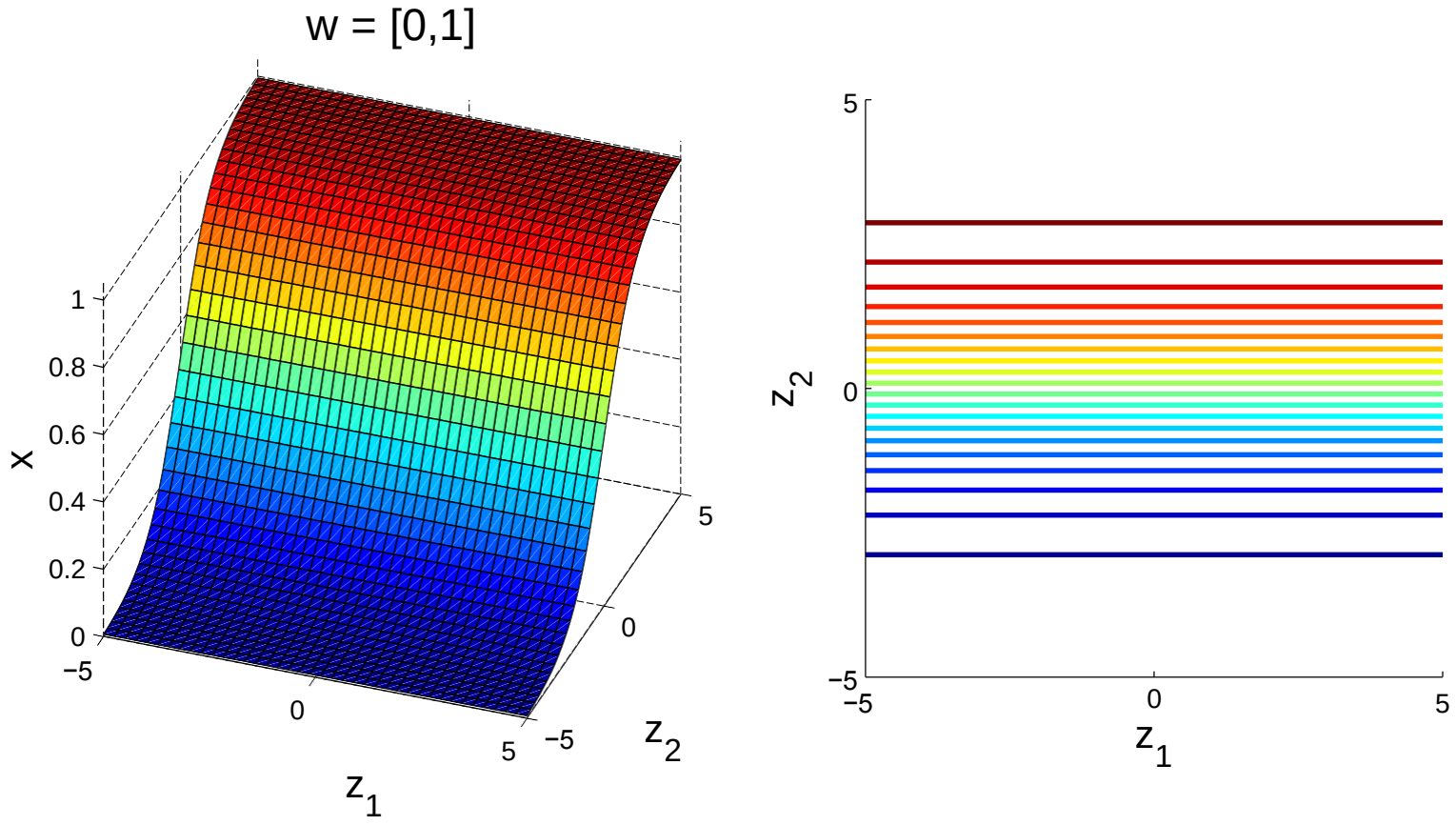
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron



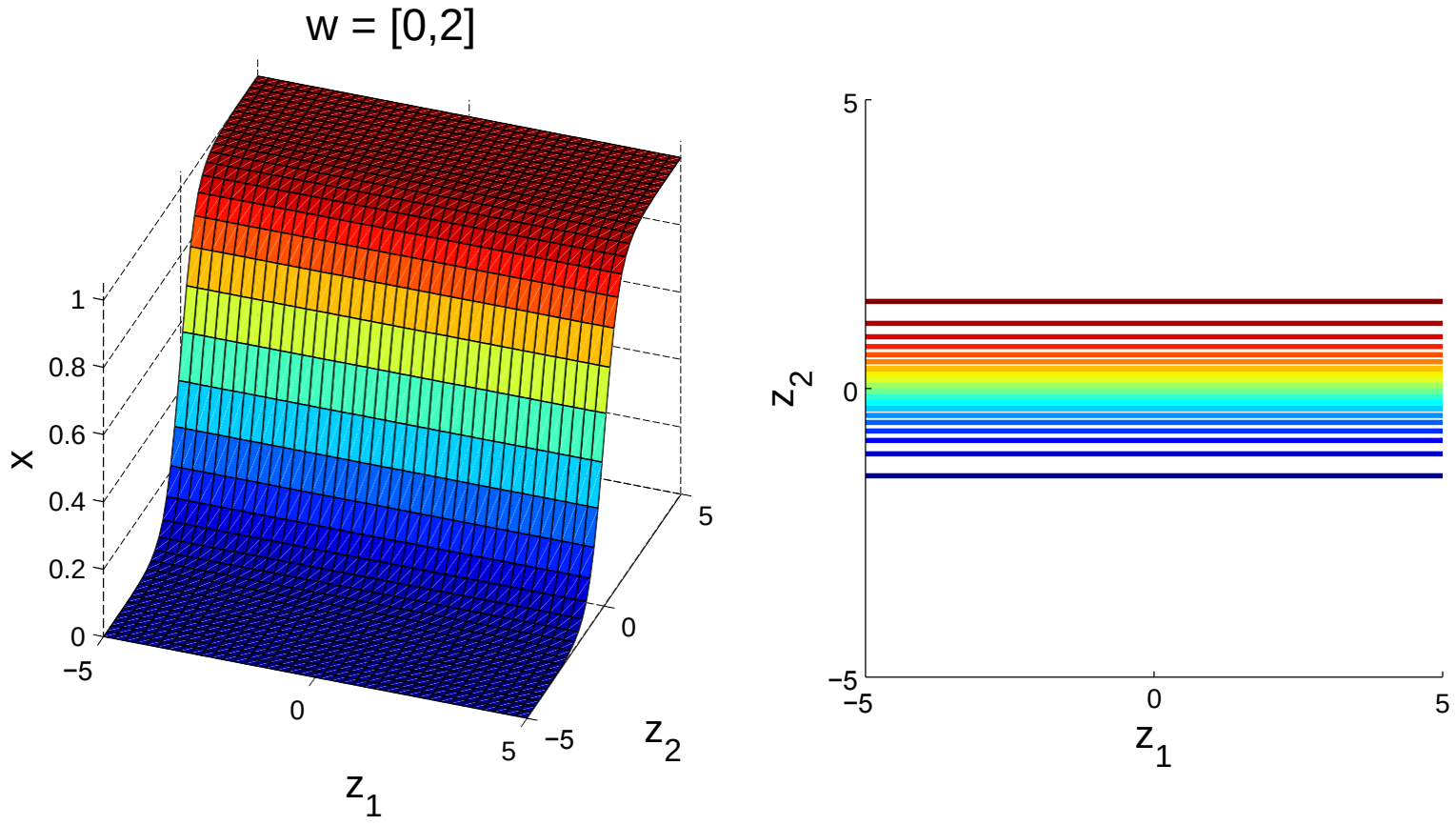
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron (cont'd)



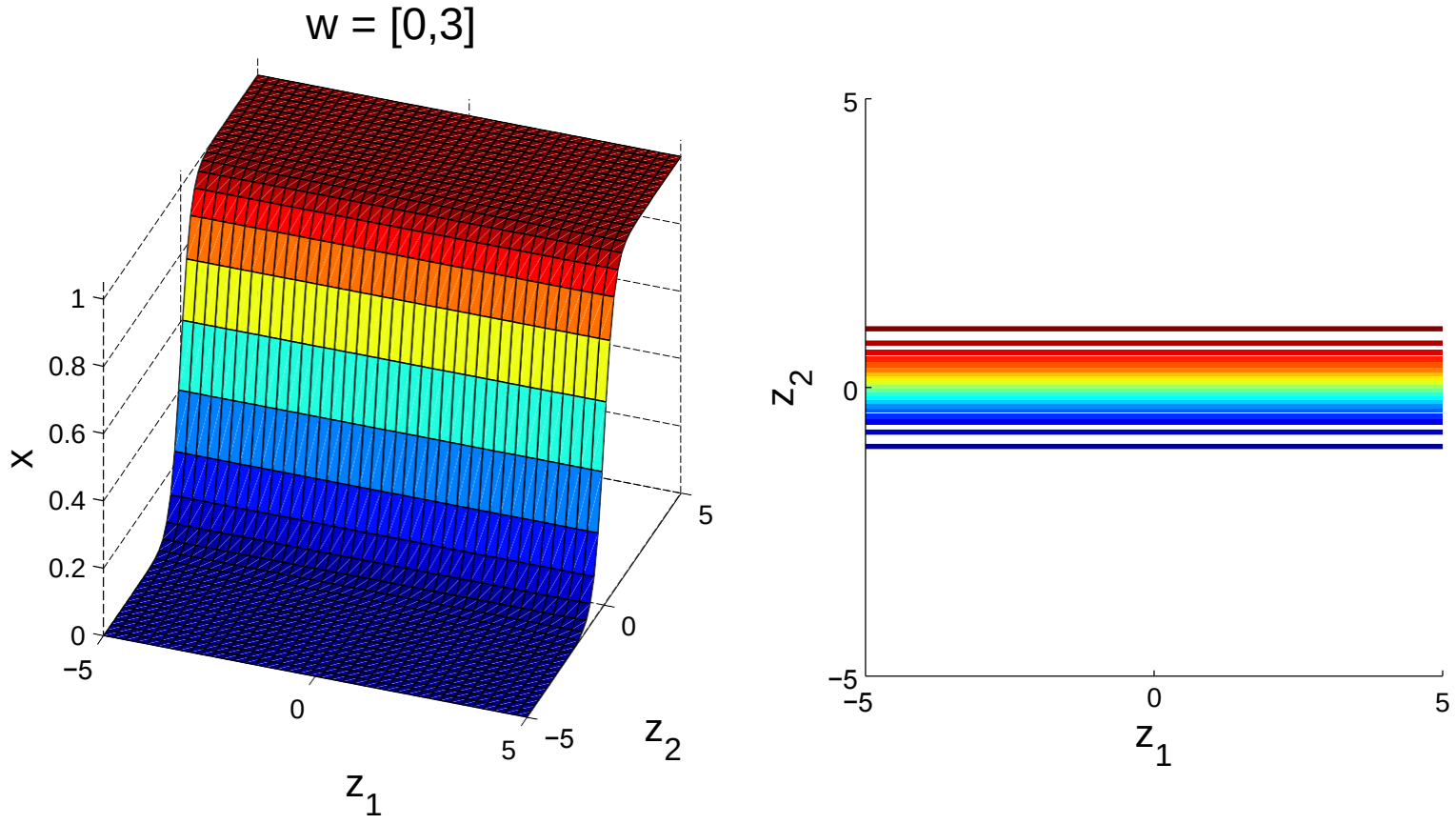
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron (cont'd)



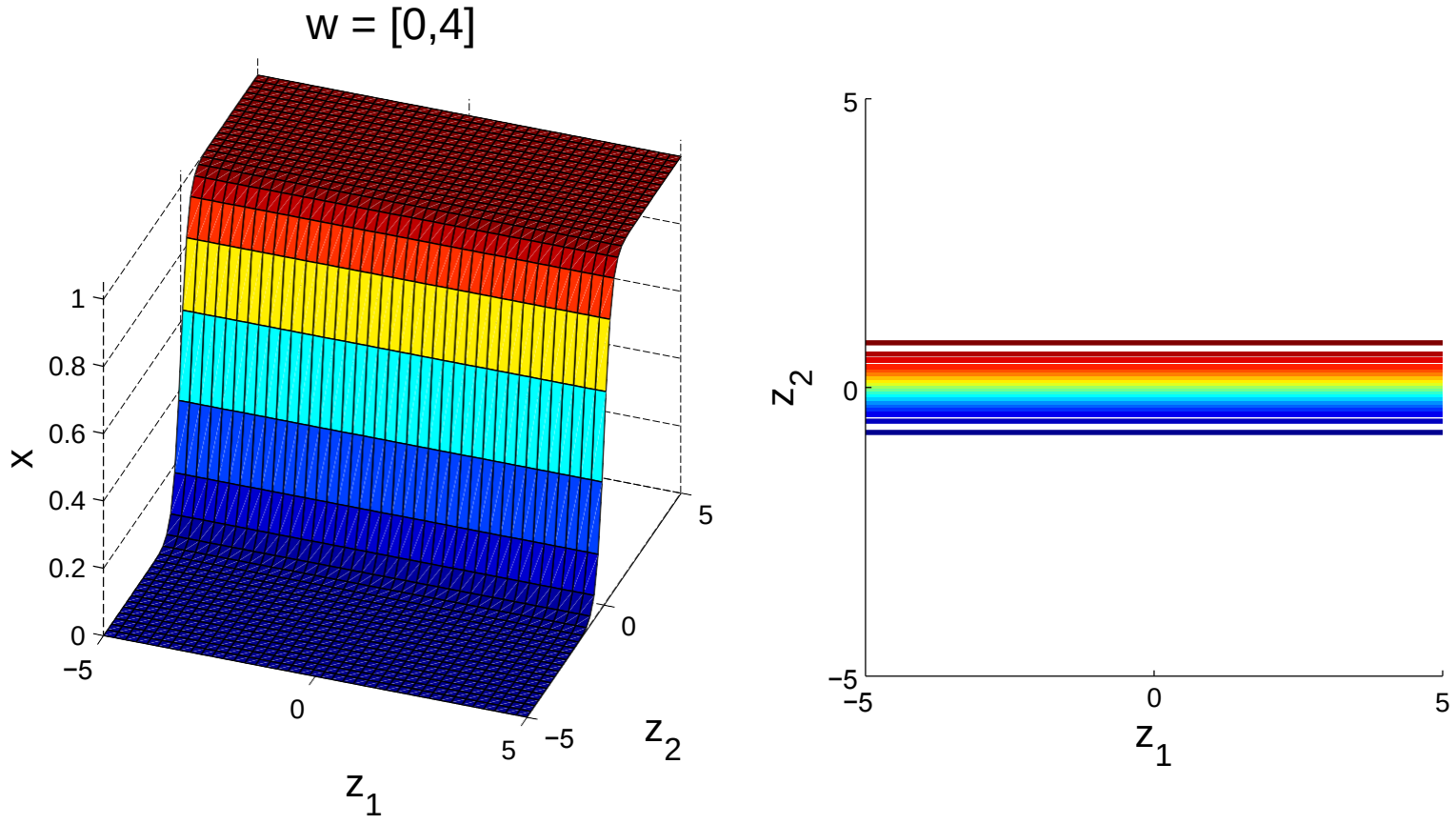
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron (cont'd)



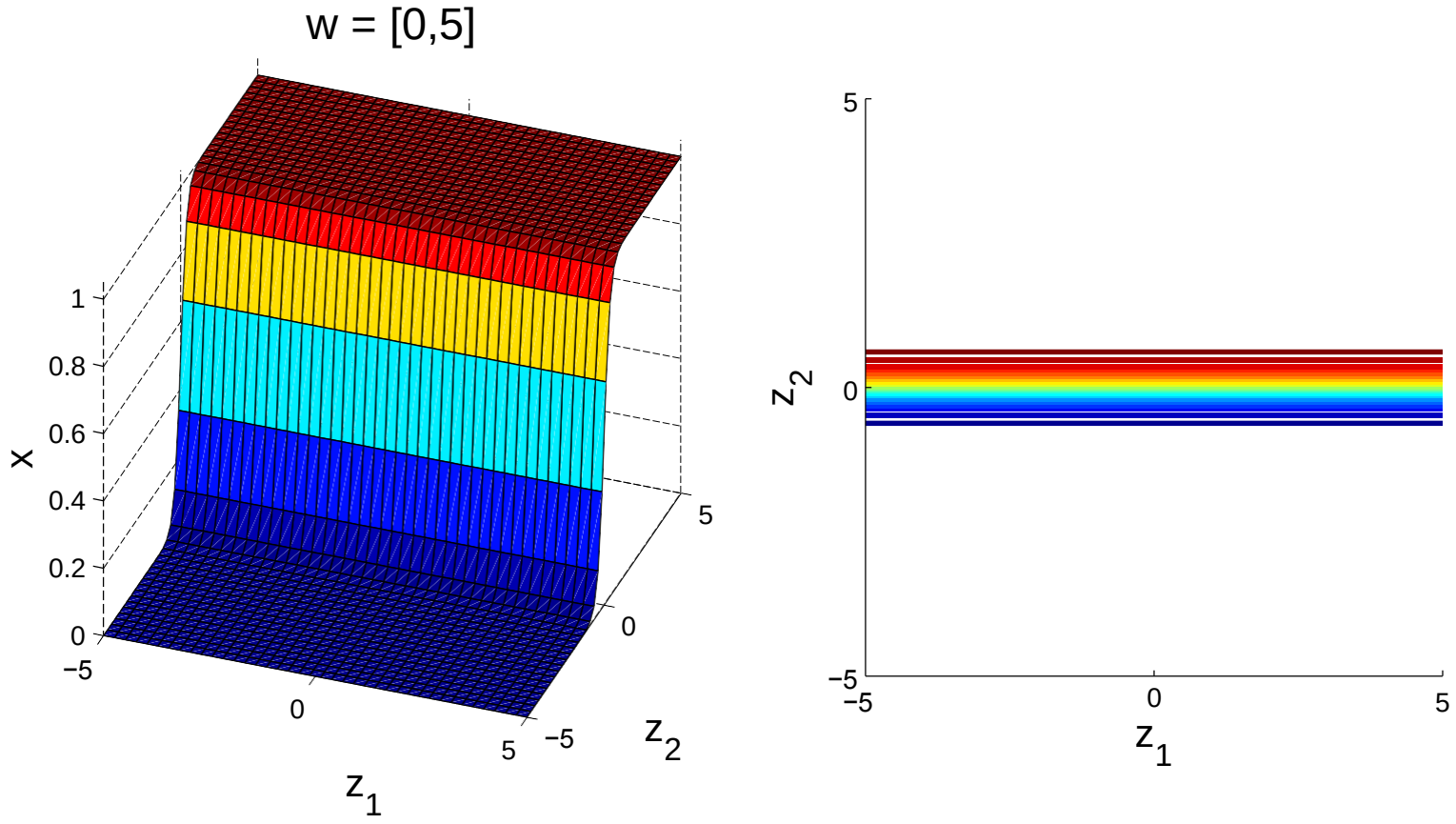
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Input-Output Function of a Single Neuron (cont'd)



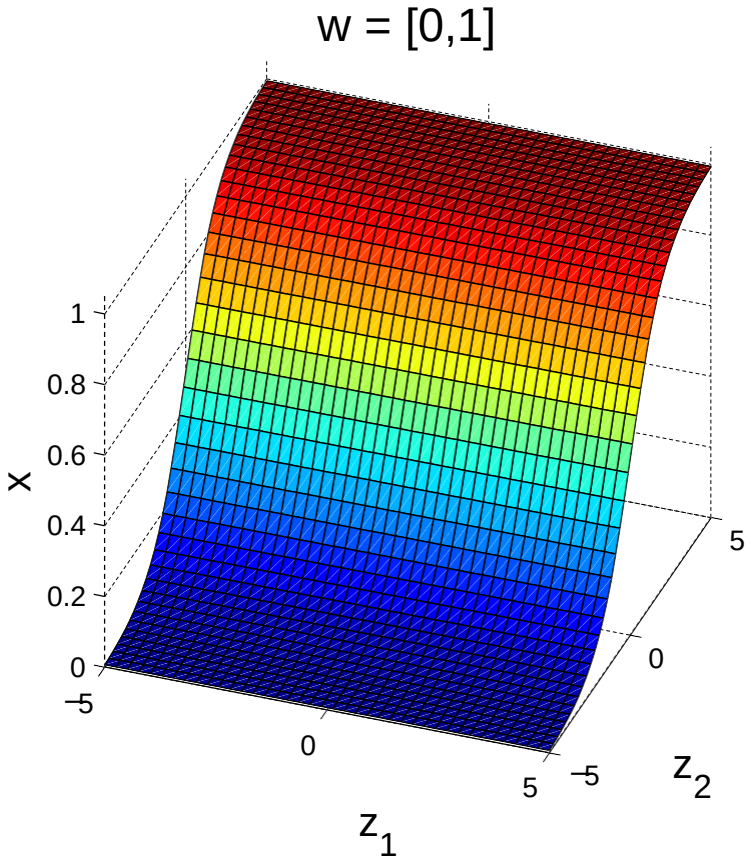
$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

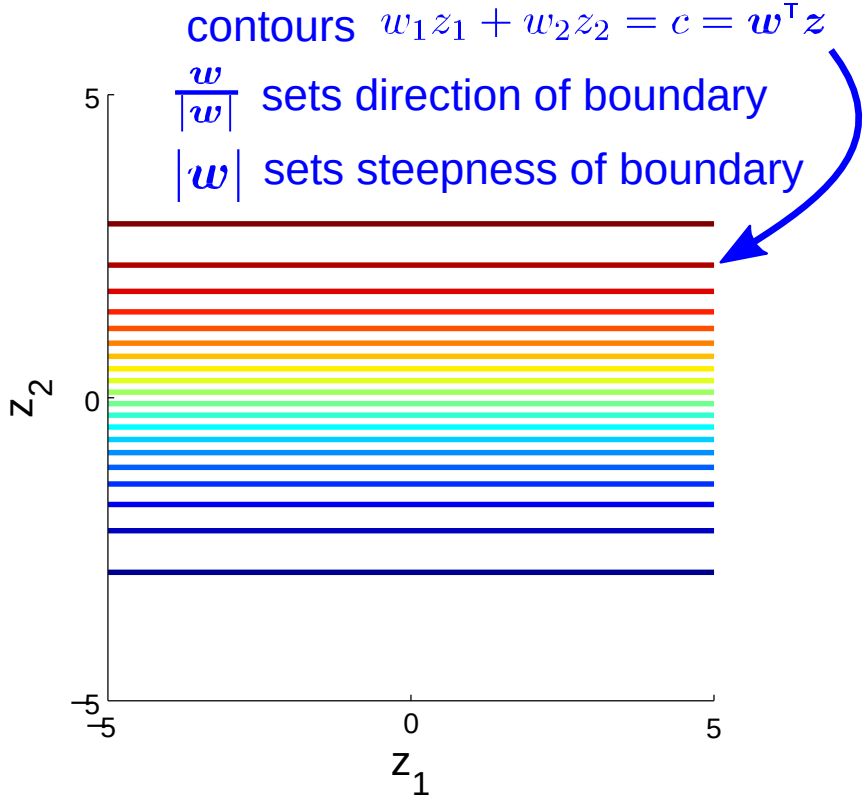
Input-Output Function of a Single Neuron (cont'd)



$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

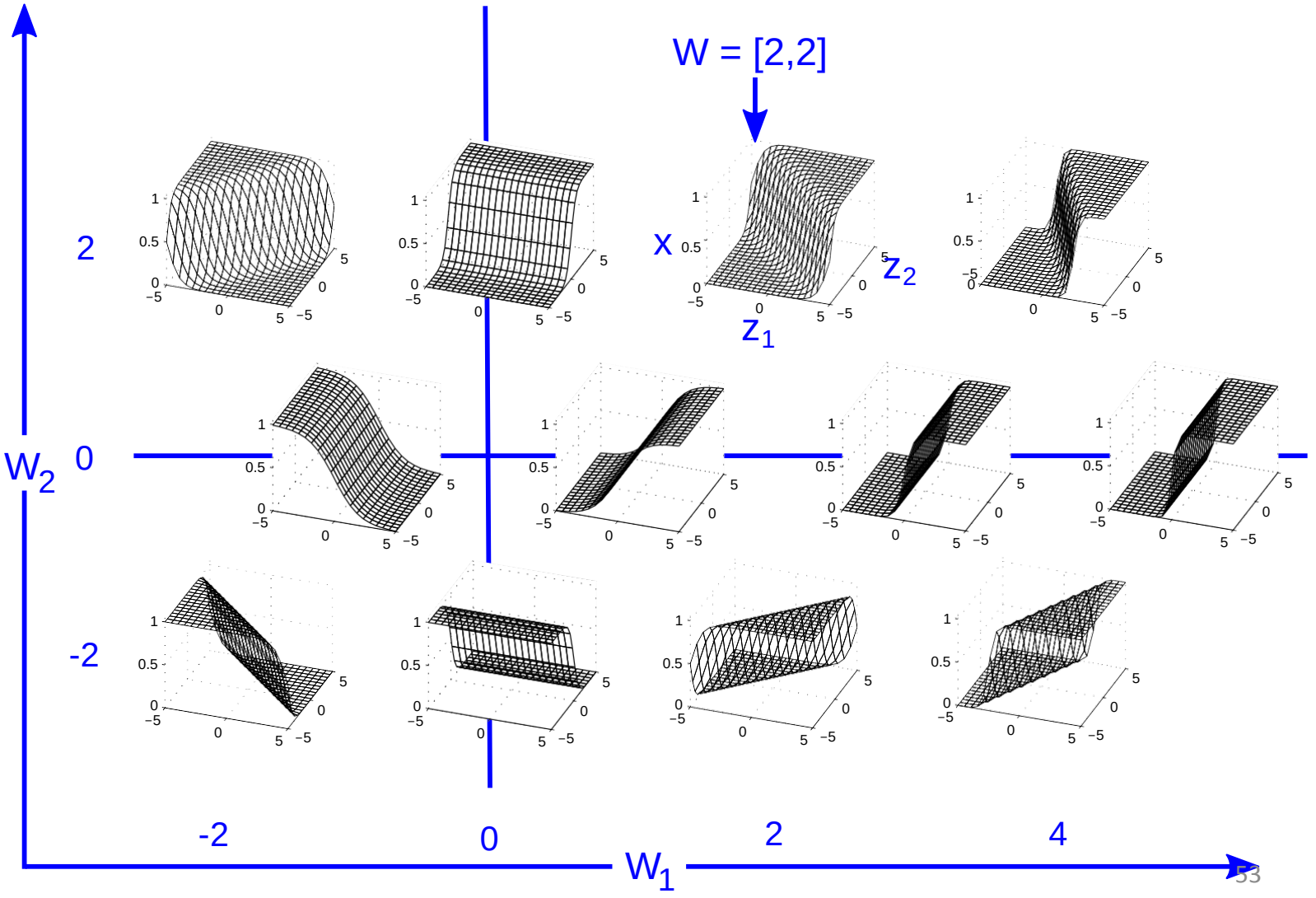
Input-Output Function of a Single Neuron (cont'd)



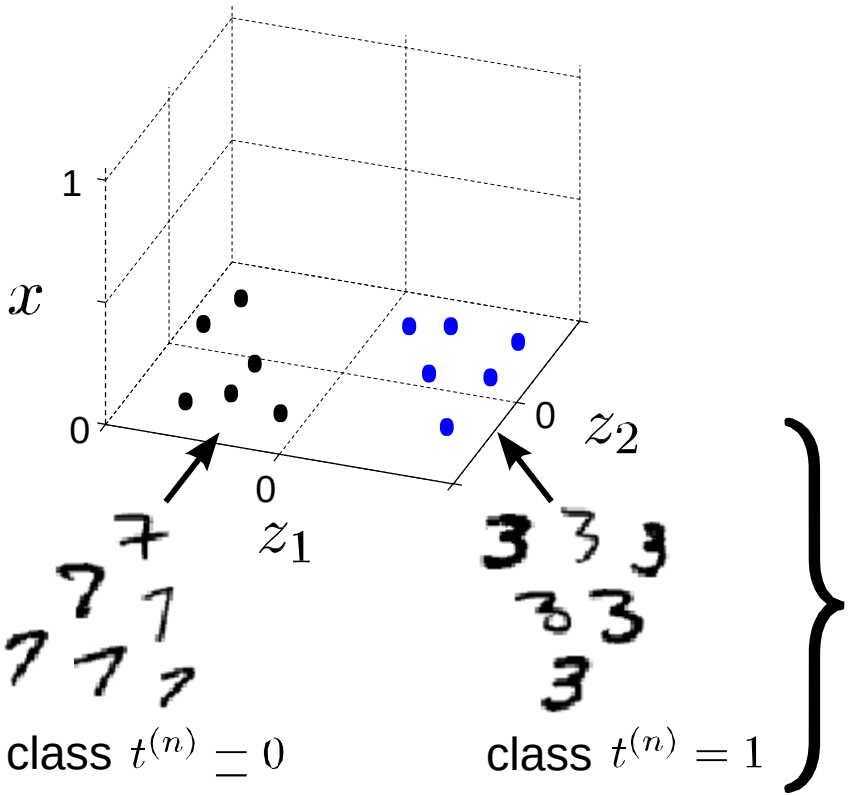


$$x(z_1, z_2) = \frac{1}{1+\exp(-w_1 z_1 - w_2 z_2)}$$

Weight Space of a Single Neuron



Training a Single Neuron

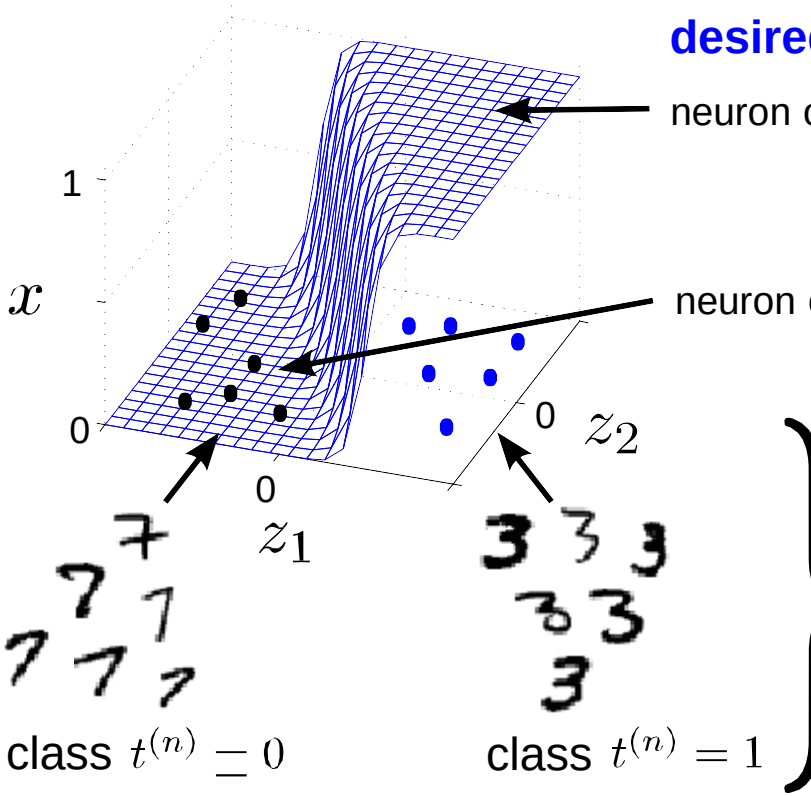


training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

Training a Single Neuron



desired result of training:

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 1$ for $t^{(n)} = 1$

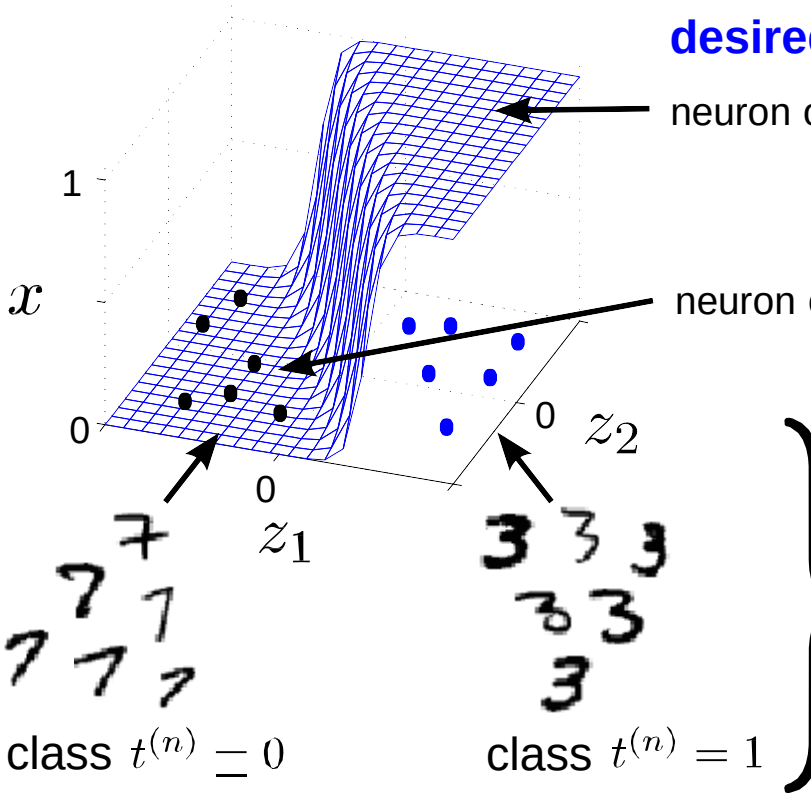
neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 0$ for $t^{(n)} = 0$

training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

Training a Single Neuron



desired result of training:

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 1$ for $t^{(n)} = 1$

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 0$ for $t^{(n)} = 0$

training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

objective function:

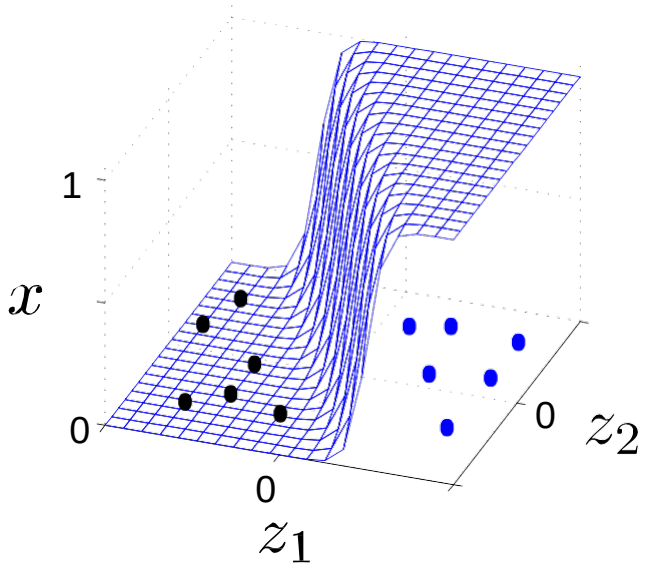
$$G(\boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - t^n) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right)\right] \geq 0$$

surprise $-\log p(\text{outcome})$ when observing $t^{(n)}$

relative entropy between $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})$ and $t^{(n)}$

} encourages neuron output to match training data

Training a Single Neuron



training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

objective function:

$$G(\boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - t^n) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right)\right] \quad \geq 0$$

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} G(\boldsymbol{w})$$

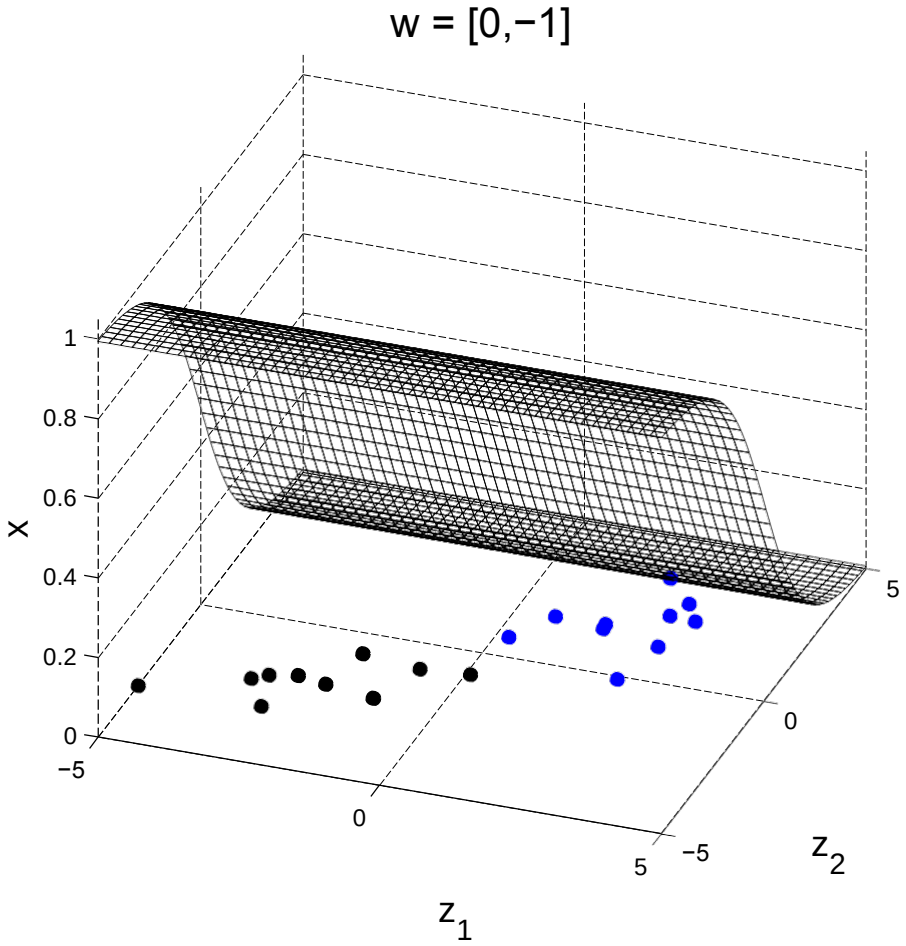
choose the weights that minimise the network's surprise about the training data

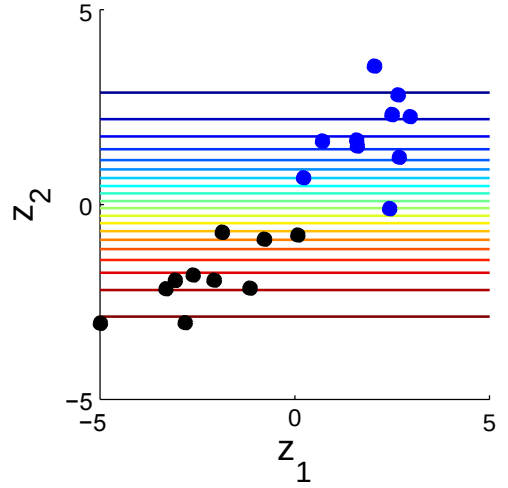
$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}} G(\boldsymbol{w}) = \sum_n \frac{\mathrm{d}G(\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}\boldsymbol{w}} = -\sum_n (t^{(n)} - x^{(n)}) \boldsymbol{z}^{(n)}$$ = prediction error x feature

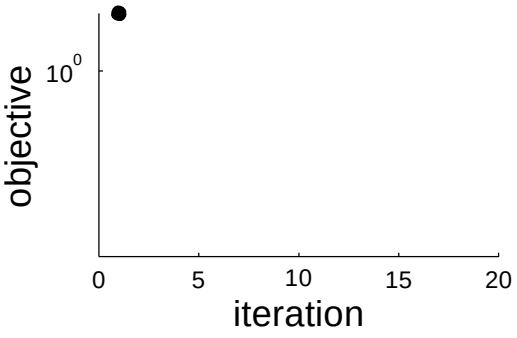
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}} G(\boldsymbol{w})$$

iteratively step down the objective (gradient points up hill)

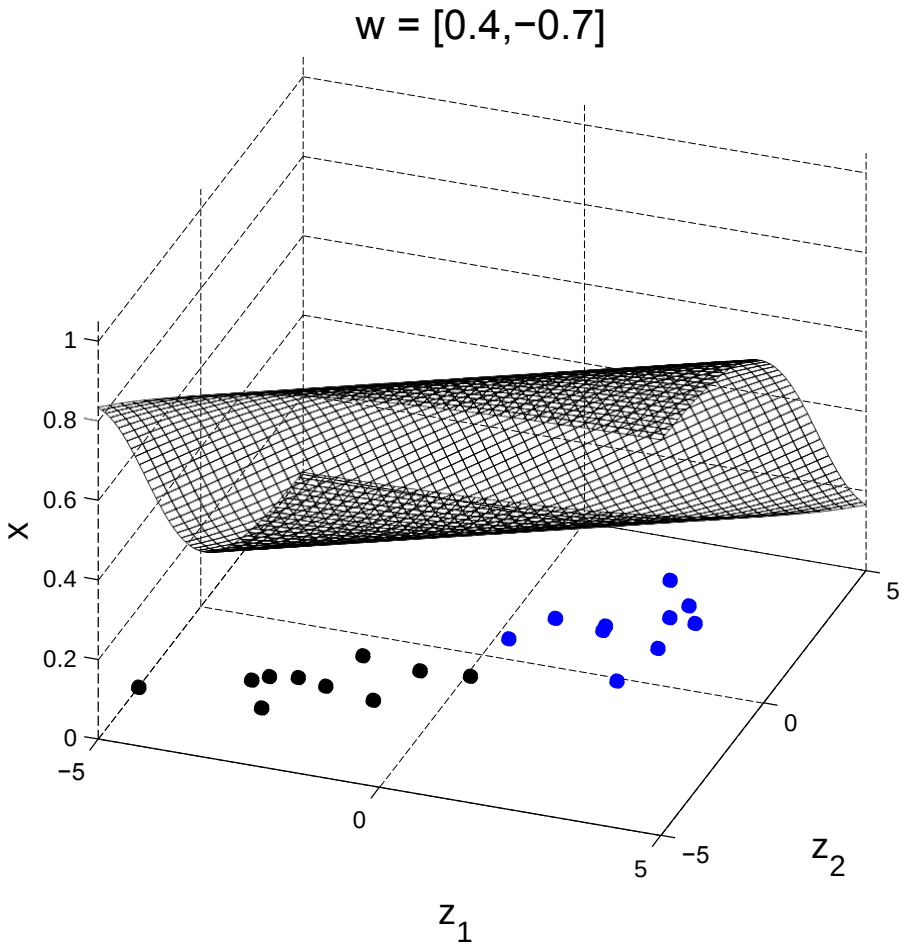
Training a Single Neuron

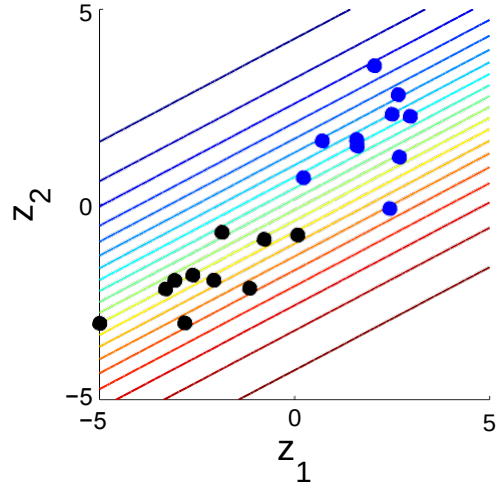


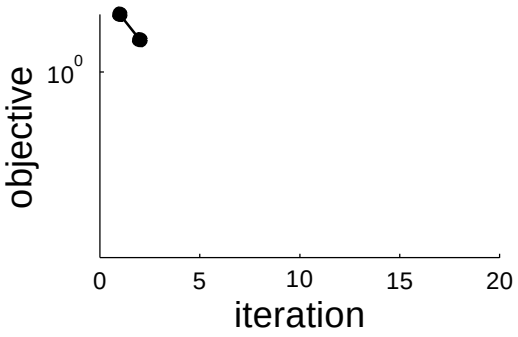




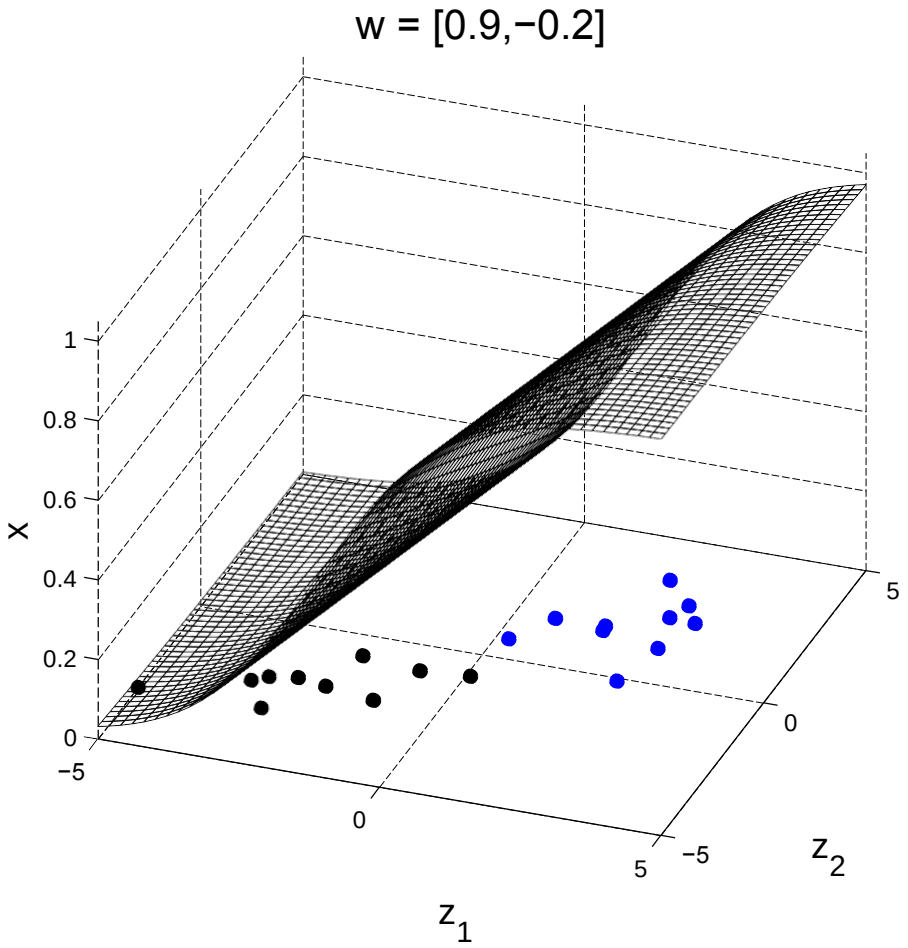
Training a Single Neuron

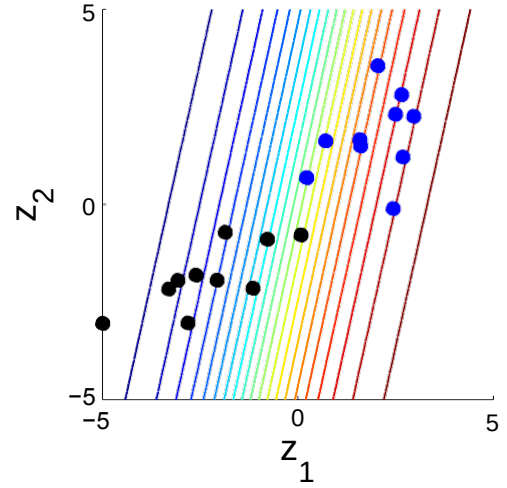


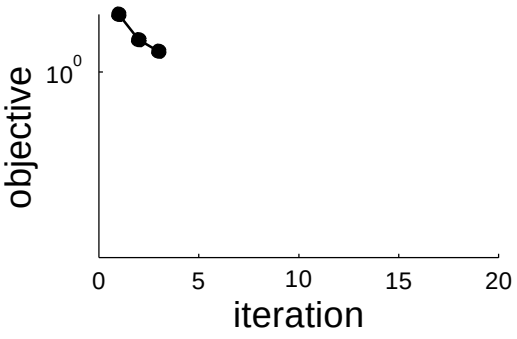




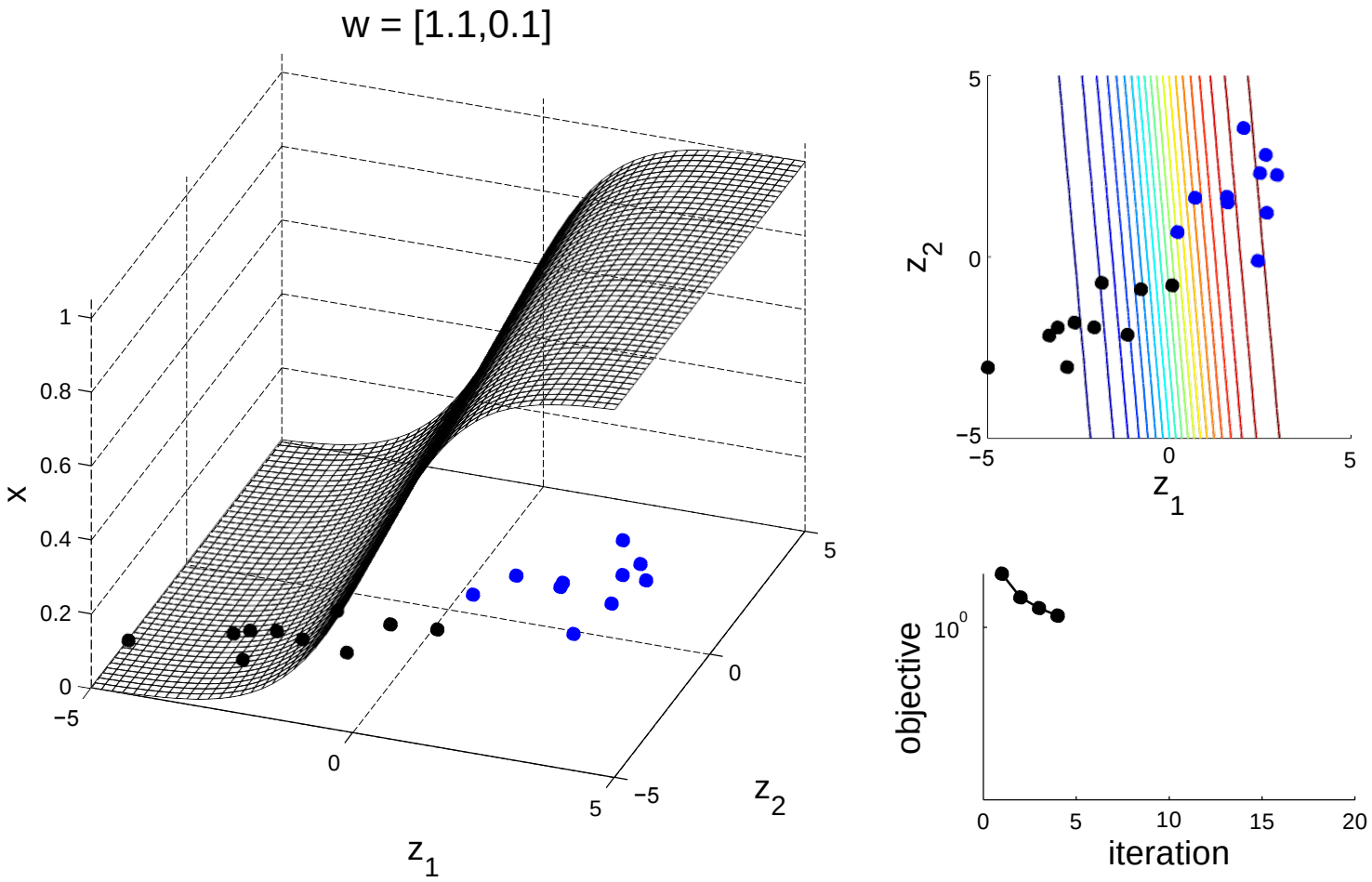
Training a Single Neuron



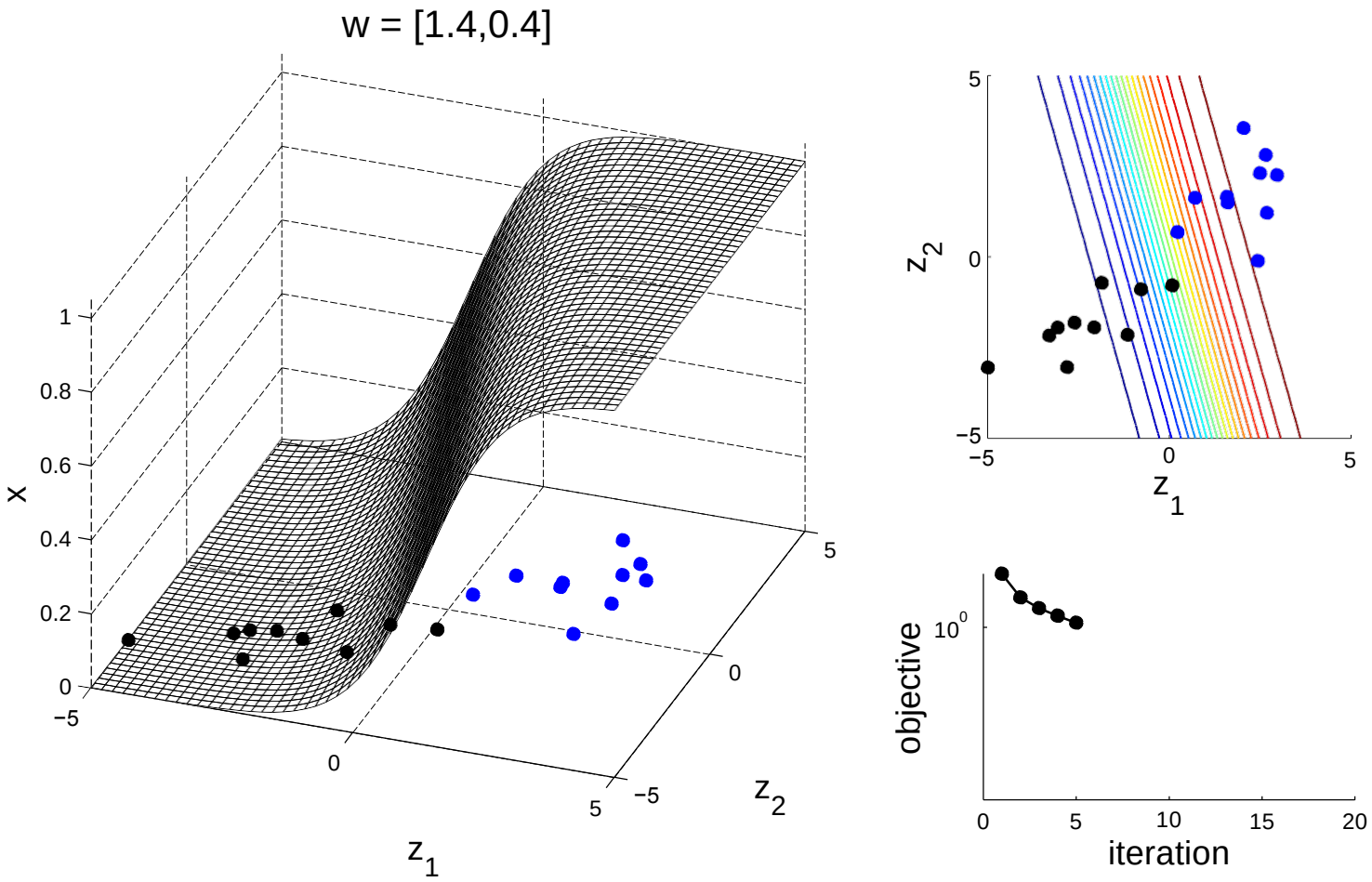




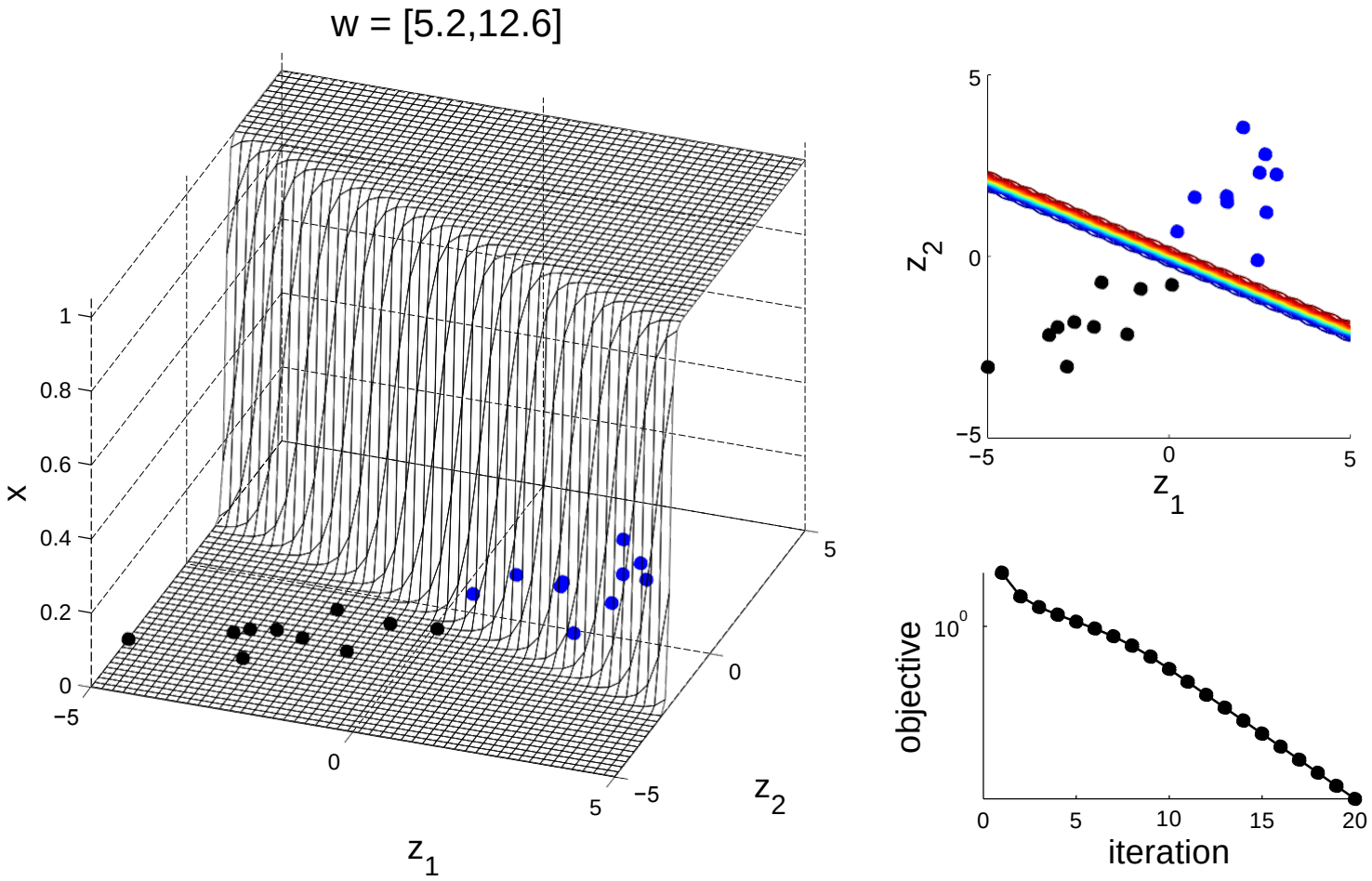
Training a Single Neuron



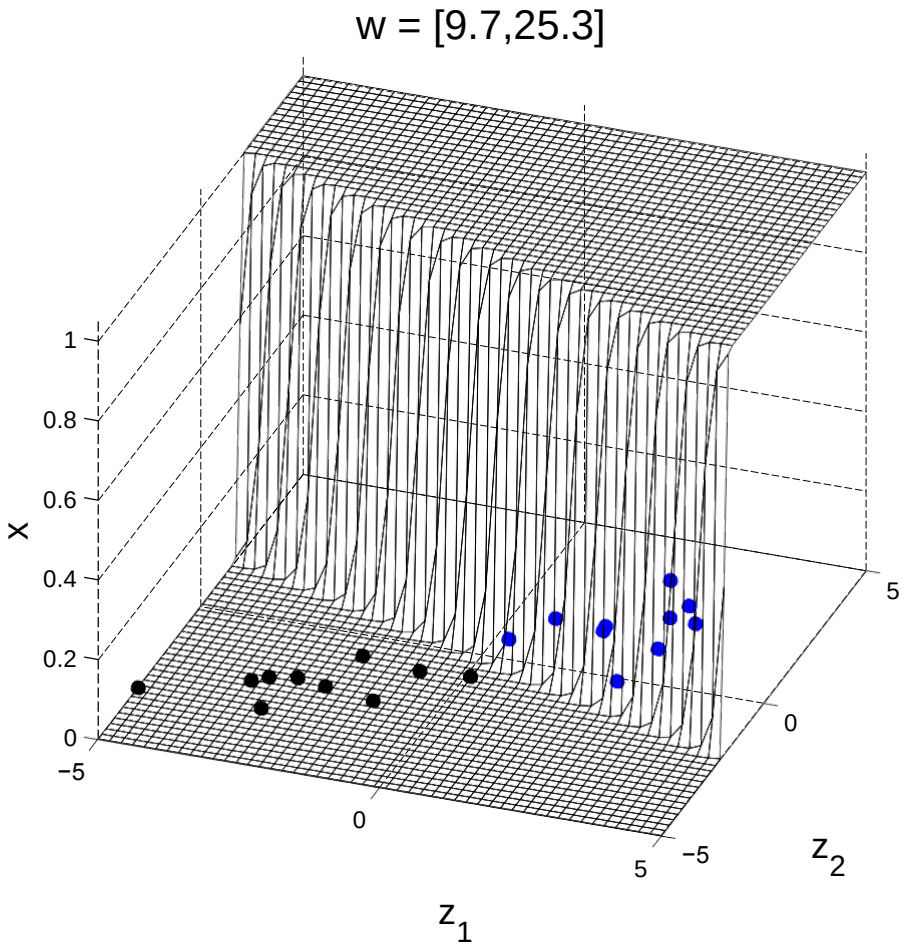
Training a Single Neuron

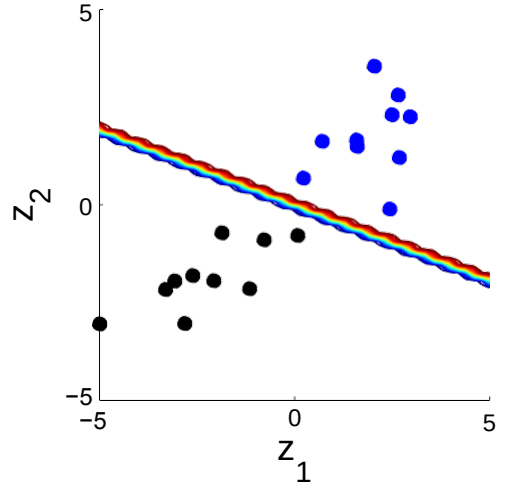


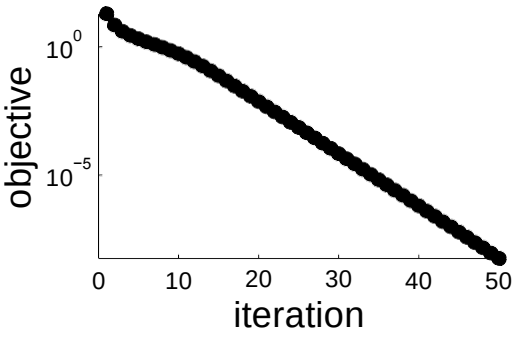
Training a Single Neuron



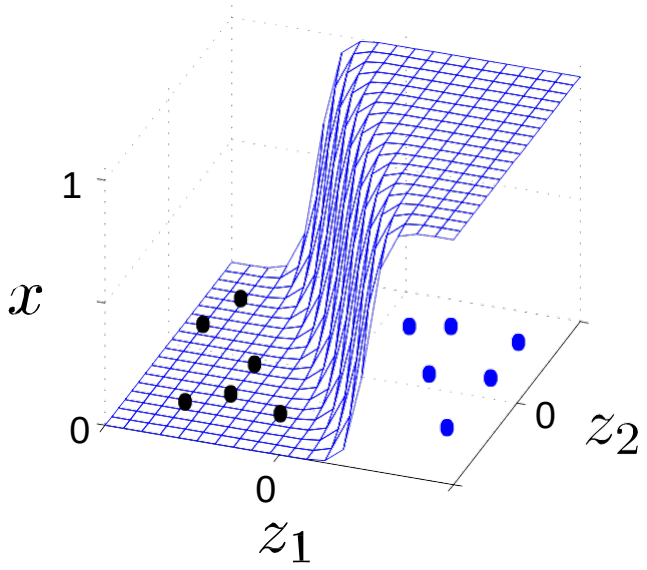
Training a Single Neuron







Overfitting and Weight Decay



training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

objective function:

$$G(\boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - t^n) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right)\right]$$

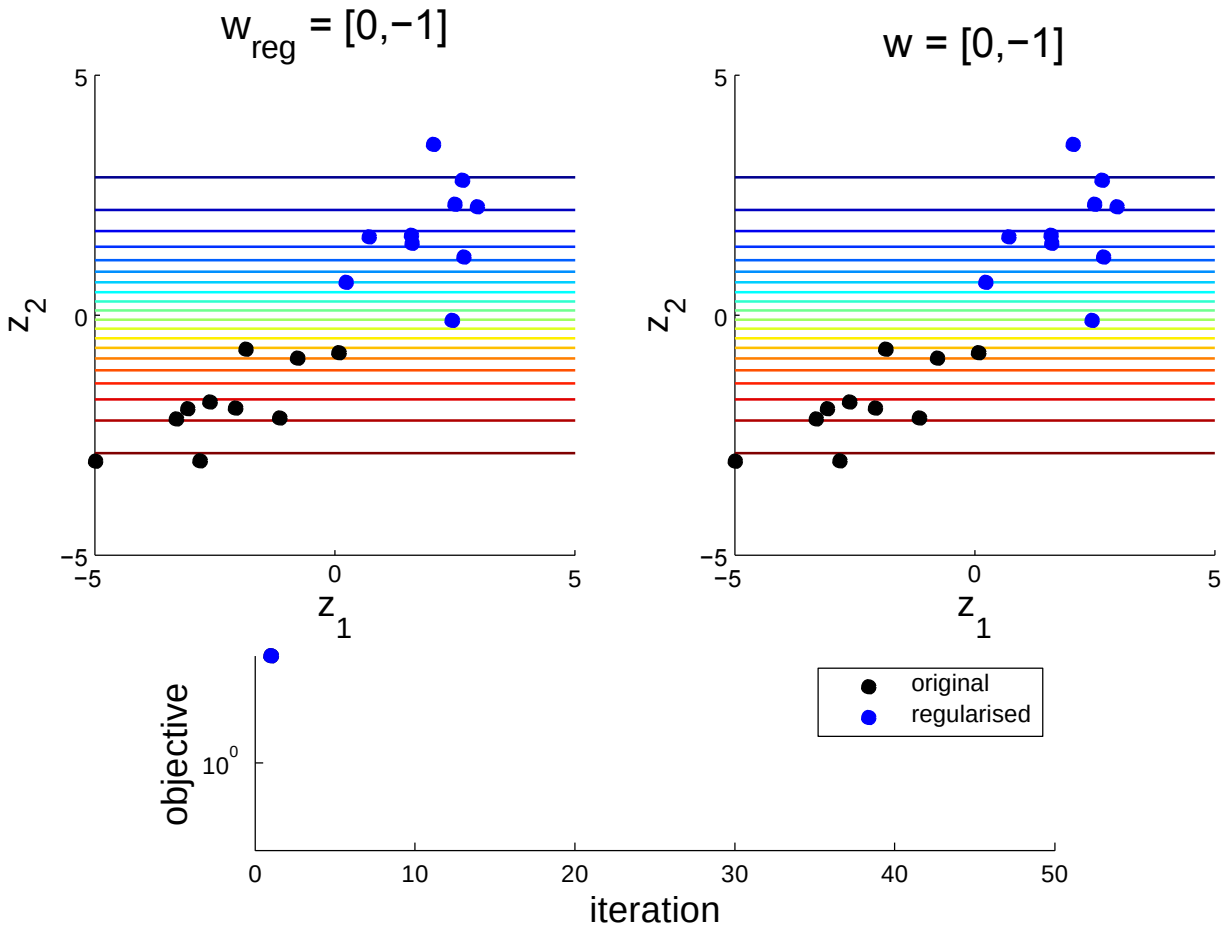
$E(\boldsymbol{w}) = \frac{1}{2} \sum_i w_i^2$ regulariser discourages the network using extreme weights

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} M(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left[G(\boldsymbol{w}) + \alpha E(\boldsymbol{w})\right]$$

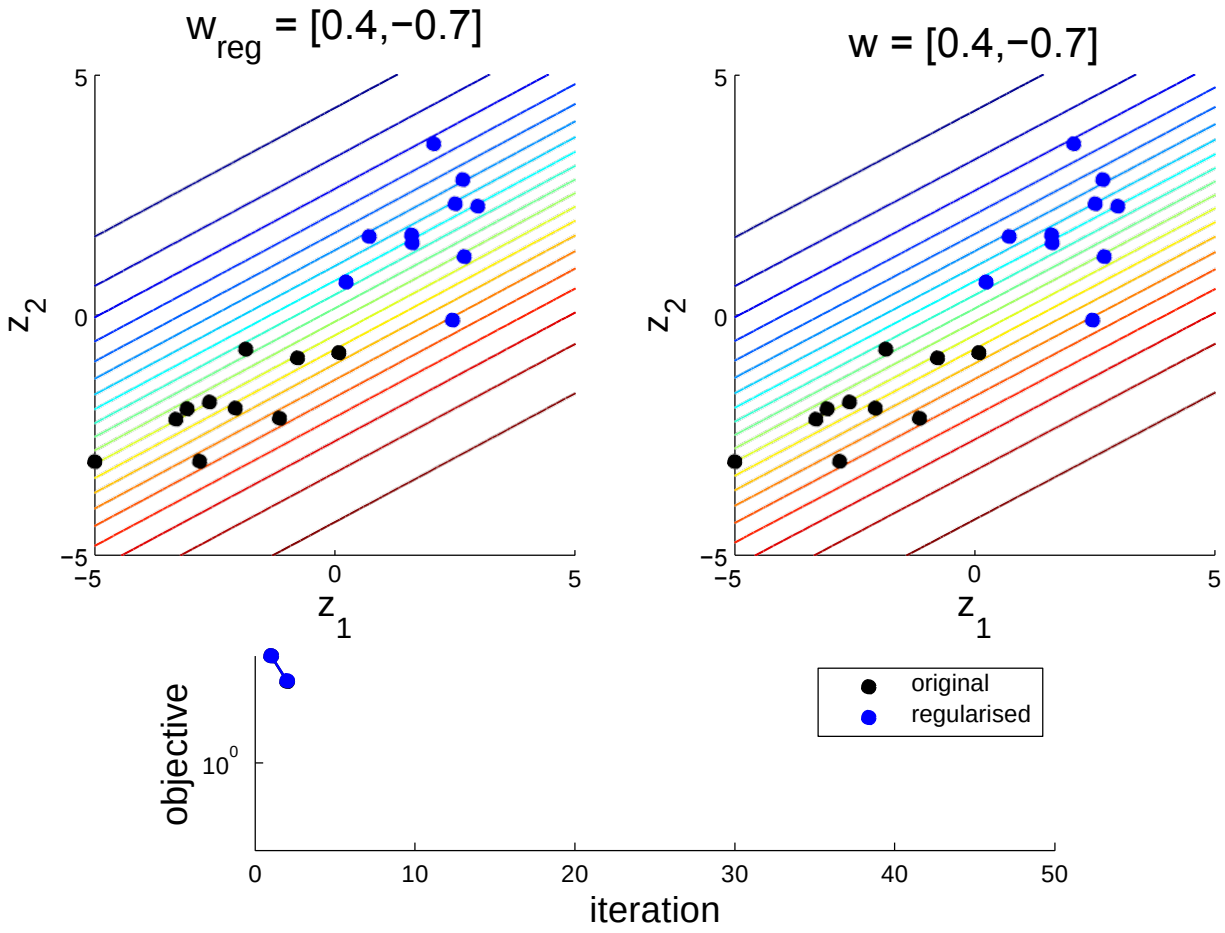
$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}} M(\boldsymbol{w}) = -\sum_n (t^{(n)} - x^{(n)}) \boldsymbol{z}^{(n)} + \alpha \boldsymbol{w}$$

weight decay - shrinks weights towards zero

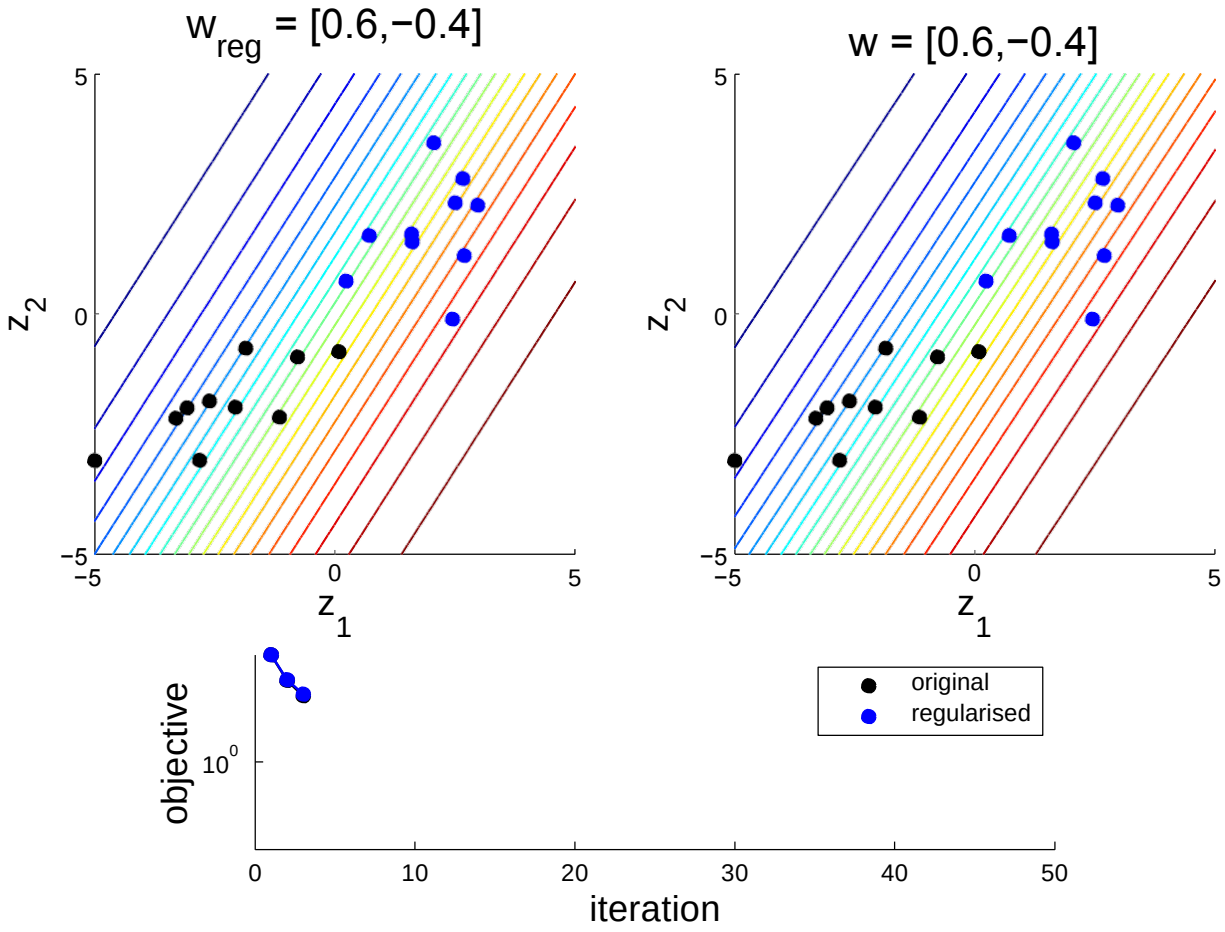
Training a Single Neuron (cont'd)



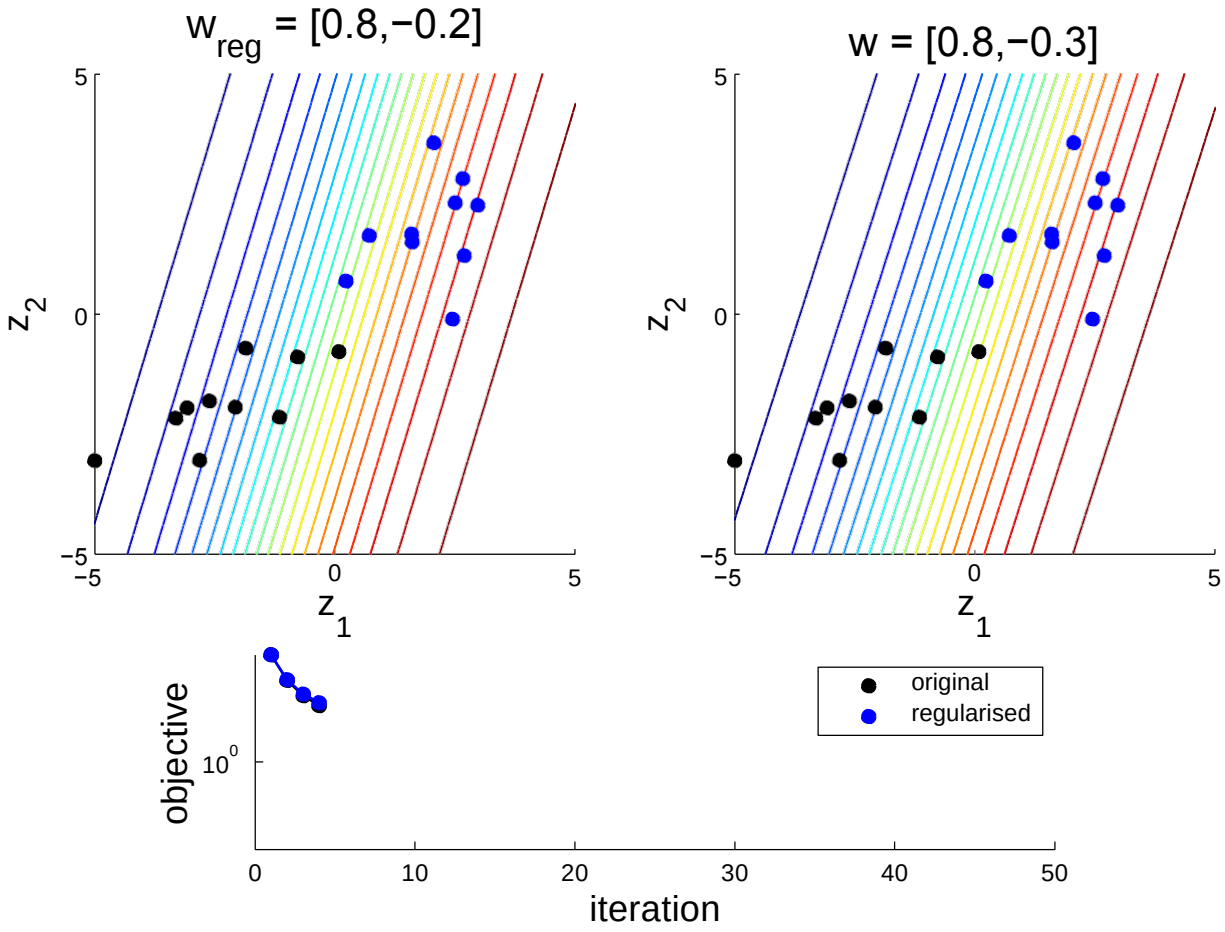
Training a Single Neuron (cont'd)



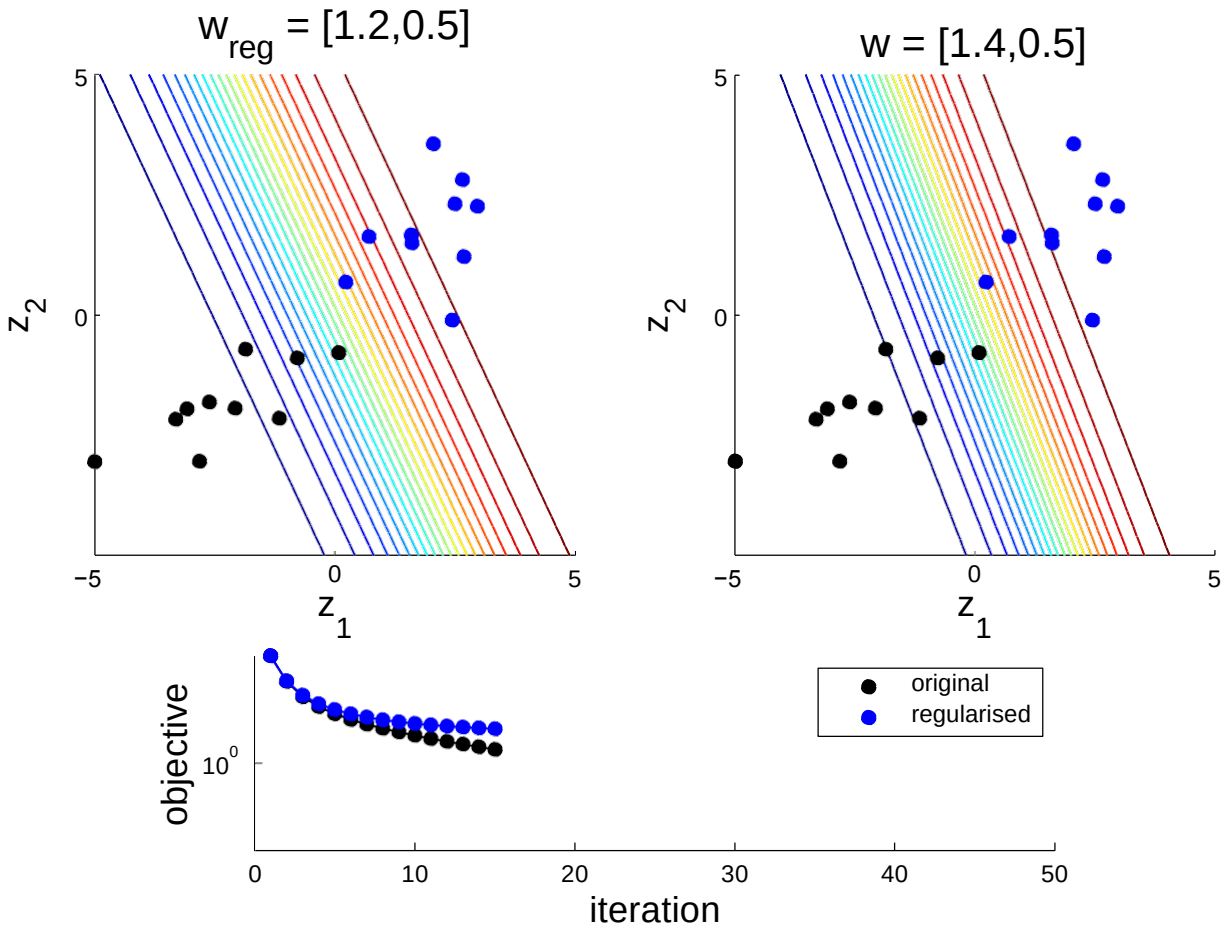
Training a Single Neuron (cont'd)



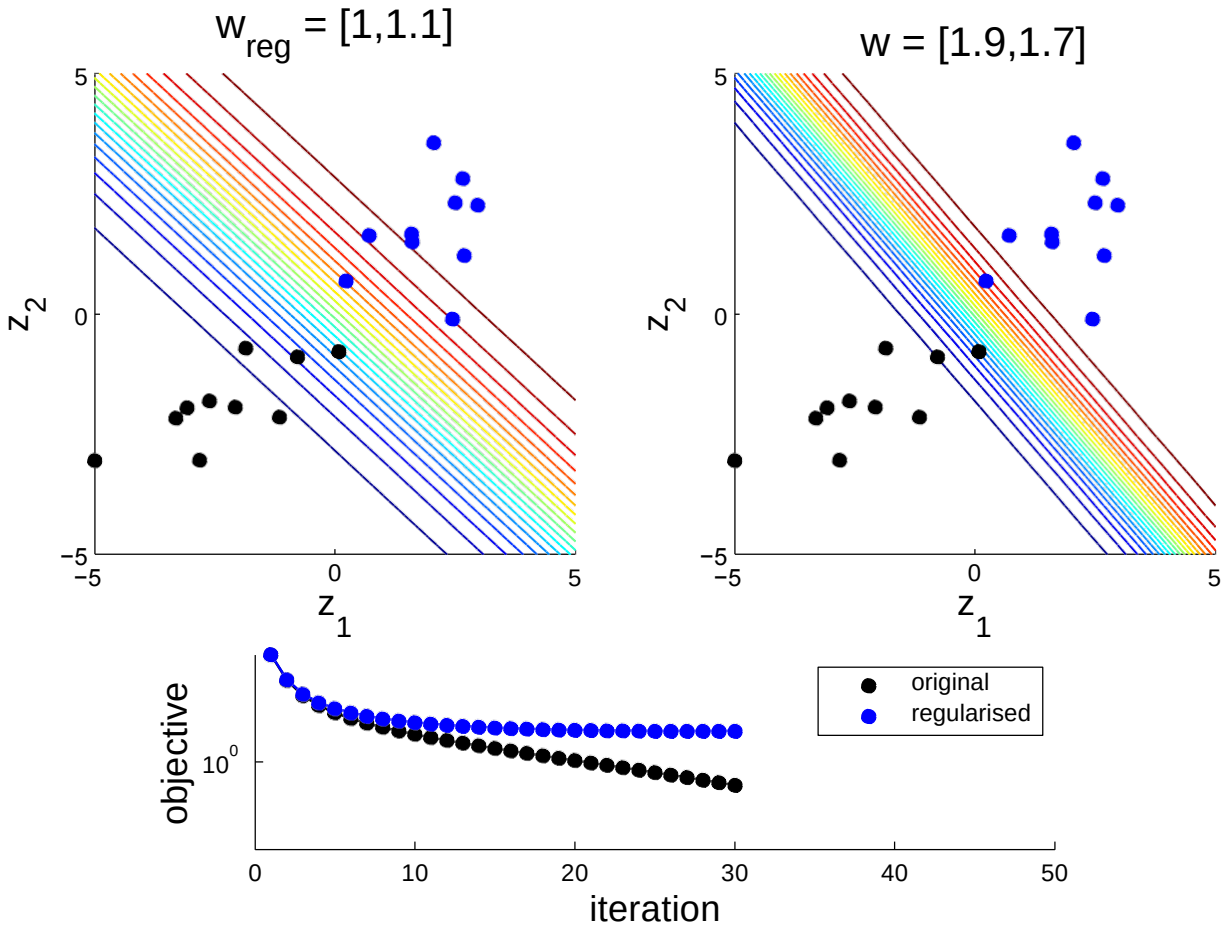
Training a Single Neuron (cont'd)



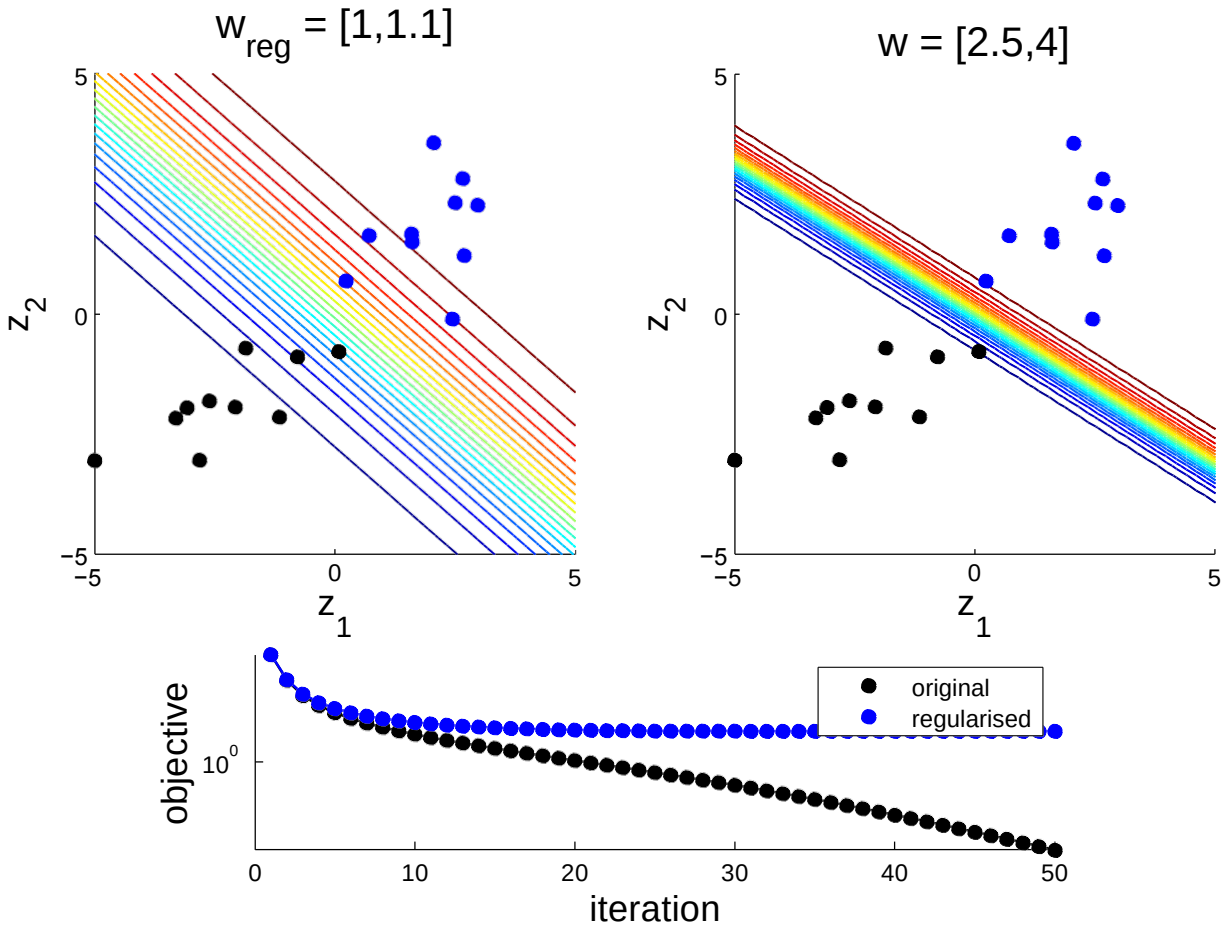
Training a Single Neuron (cont'd)



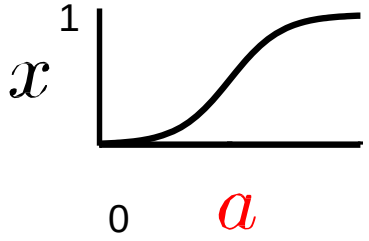
Training a Single Neuron (cont'd)

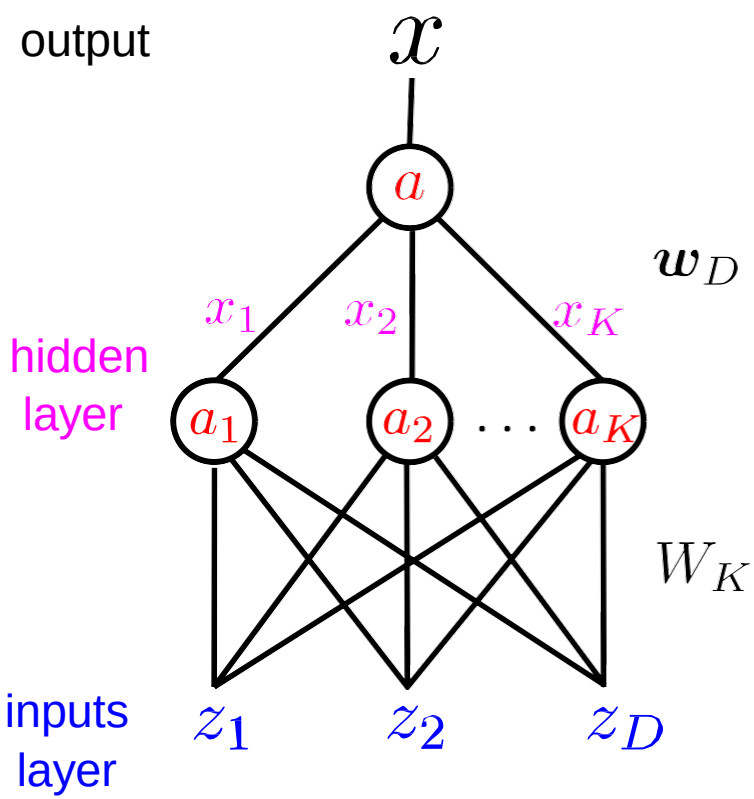


Training a Single Neuron (cont'd)



Single Hidden Layer Neural Networks





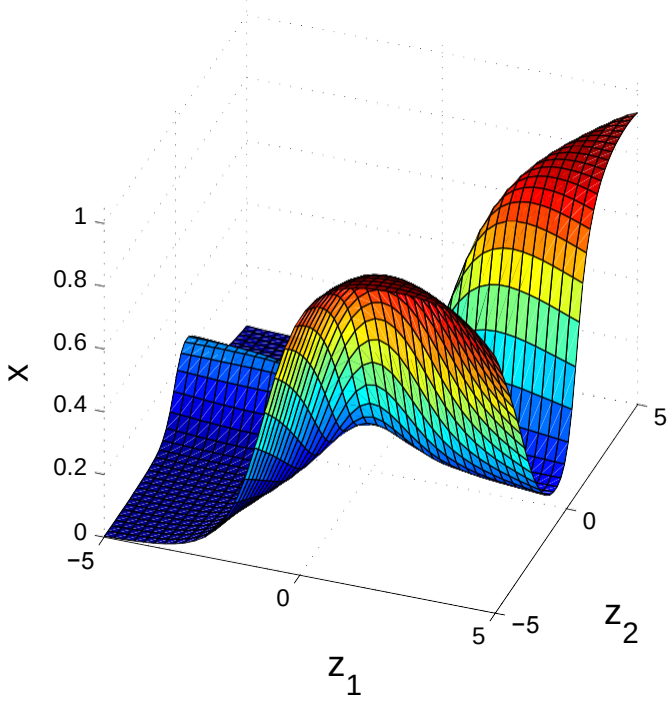
$$x(a) = \frac{1}{1+\exp(-a)}$$

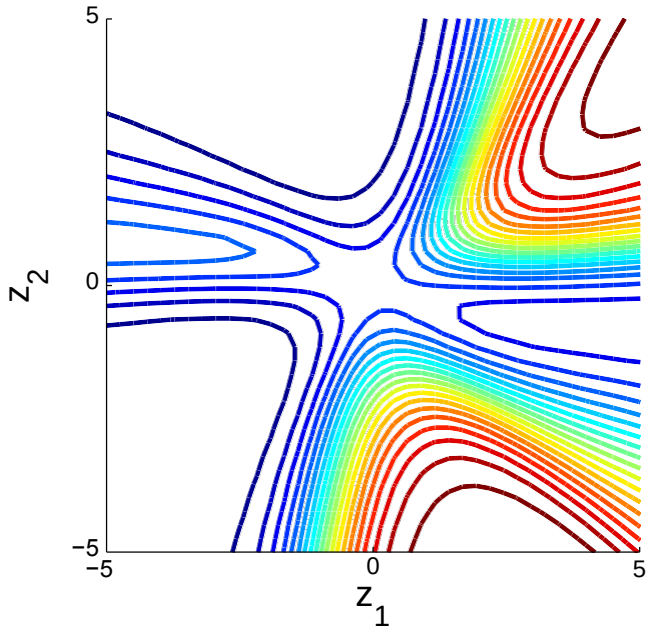
$$a = \sum_{k=1}^{K} w_k x_k$$

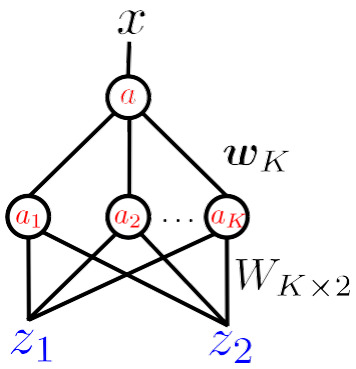
$$x(a_k) = \frac{1}{1+\exp(-a_k)}$$

$$a_k = \sum_{d=1}^{D} W_{k,d} z_d$$

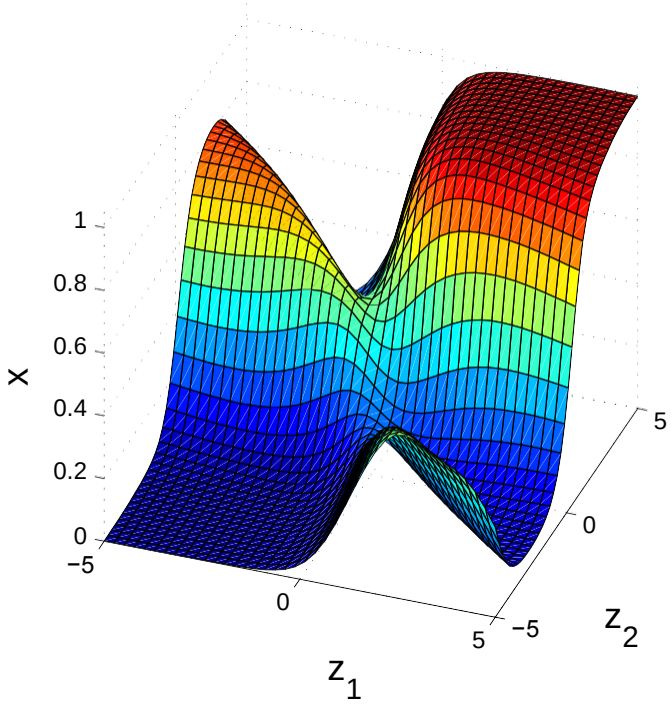
Sampling Random Neural Network Classifiers

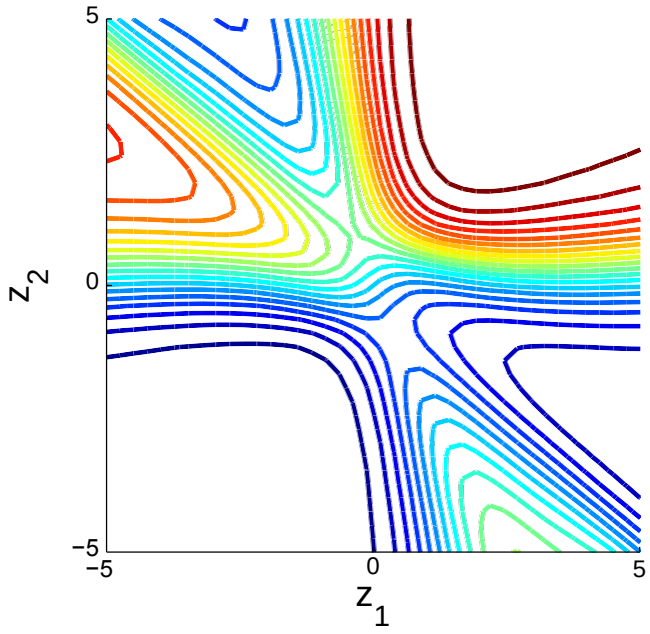


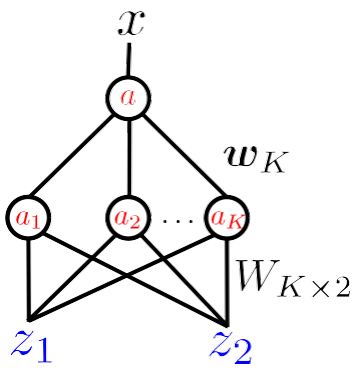




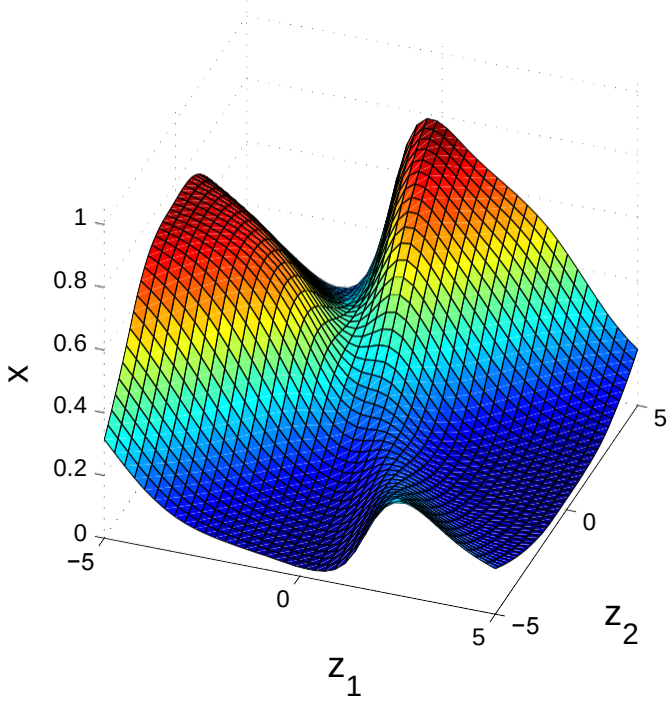
Sampling Random Neural Network Classifiers

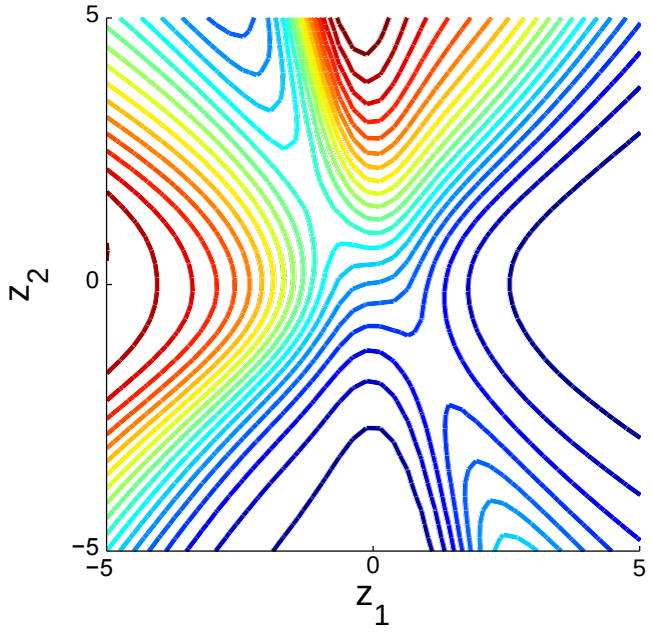


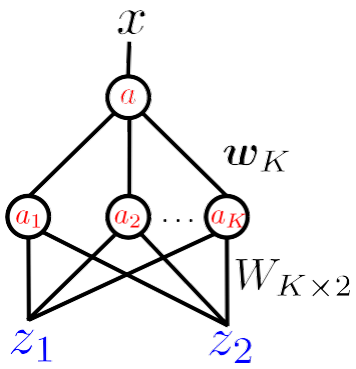




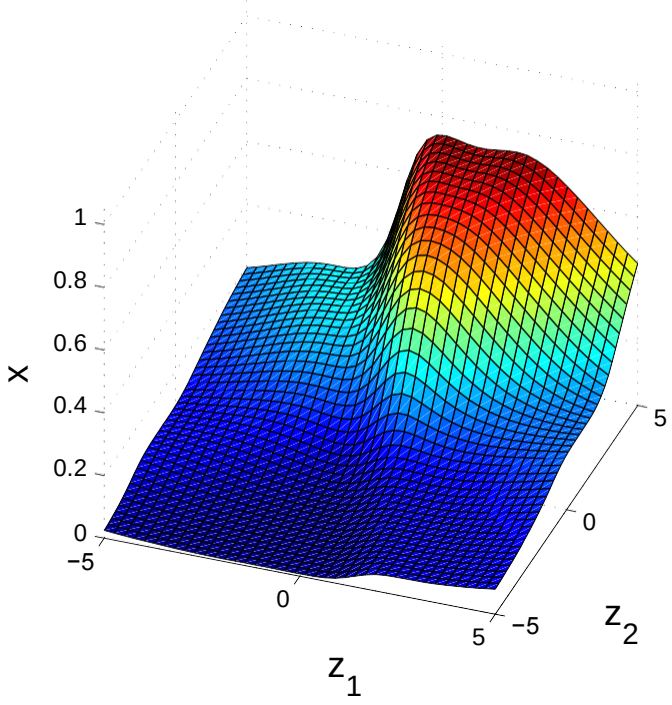
Sampling Random Neural Network Classifiers

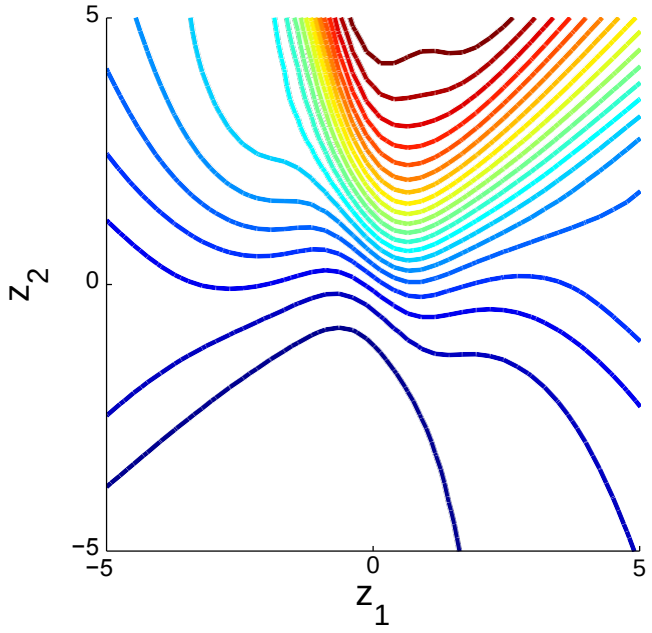


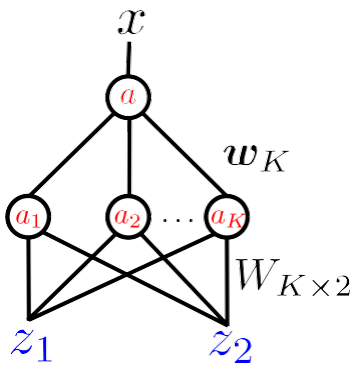




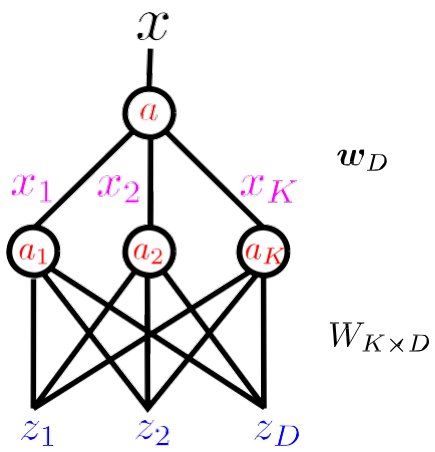
Sampling Random Neural Network Classifiers







Training a Neural Network with a Single Hidden Layer



$$x(a) = \frac{1}{1+\exp(-a)}$$

$$a = \sum_{k=1}^{K} w_k x_k$$

$$x(a_k) = \frac{1}{1+\exp(-a_k)}$$

$$a_k = \sum_{d=1}^{D} W_{k,d} z_d$$

objective function:

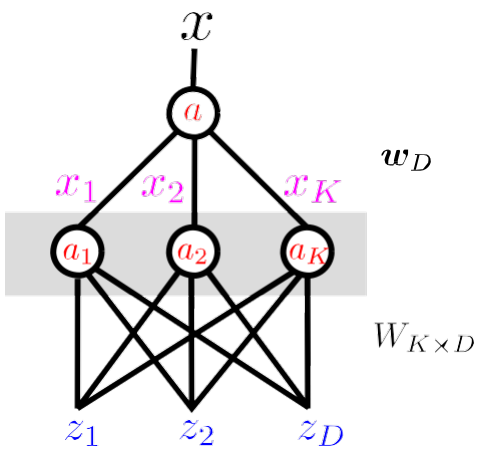
$$G(W, \boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}^{(n)} + (1 - t^n) \log \left(1 - \mathrm{x}^{(n)}\right)\right]$$ likelihood same as before

$$E(W, \boldsymbol{w}) = \tfrac{1}{2} \sum_i w_i^2 + \tfrac{1}{2} \sum_{ij} W_{ij}^2$$ regulariser discourages extreme weights

$$\{W, \boldsymbol{w}^*\} = \arg\min_{W,\boldsymbol{w}} M(W, \boldsymbol{w}) = \arg\min_{W,\boldsymbol{w}} \left[G(W, \boldsymbol{w}) + \alpha E(W, \boldsymbol{w})\right]$$

Training a Neural Network with a Single Hidden Layer

Networks with hidden layers can be fit using gradient descent using an algorithm called back-propagation.



$$x(a) = \frac{1}{1+\exp(-a)}$$

$$a = \sum_{k=1}^{K} w_k x_k$$

$$x(a_k) = \frac{1}{1+\exp(-a_k)}$$

$$a_k = \sum_{d=1}^{D} W_{k,d} z_d$$

objective function:

$$G(W, \boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}^{(n)} + (1 - t^n) \log \left(1 - \mathrm{x}^{(n)}\right)\right]$$ likelihood same as before

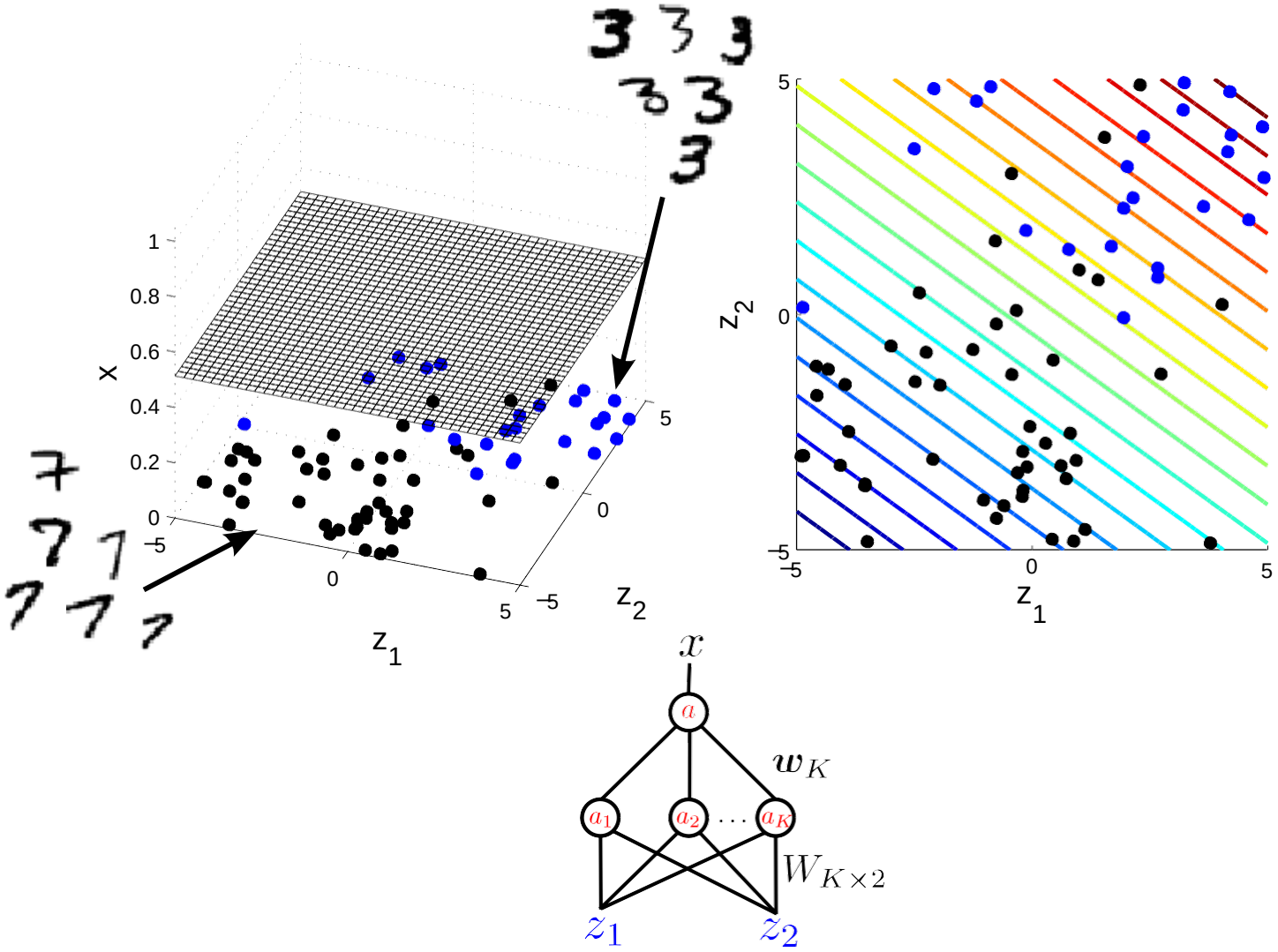
$$E(W, \boldsymbol{w}) = \tfrac{1}{2} \sum_i w_i^2 + \tfrac{1}{2} \sum_{ij} W_{ij}^2$$ regulariser discourages extreme weights

$$\{W, \boldsymbol{w}^*\} = \arg\min_{W,\boldsymbol{w}} M(W, \boldsymbol{w}) = \arg\min_{W,\boldsymbol{w}} \left[G(W, \boldsymbol{w}) + \alpha E(W, \boldsymbol{w})\right]$$

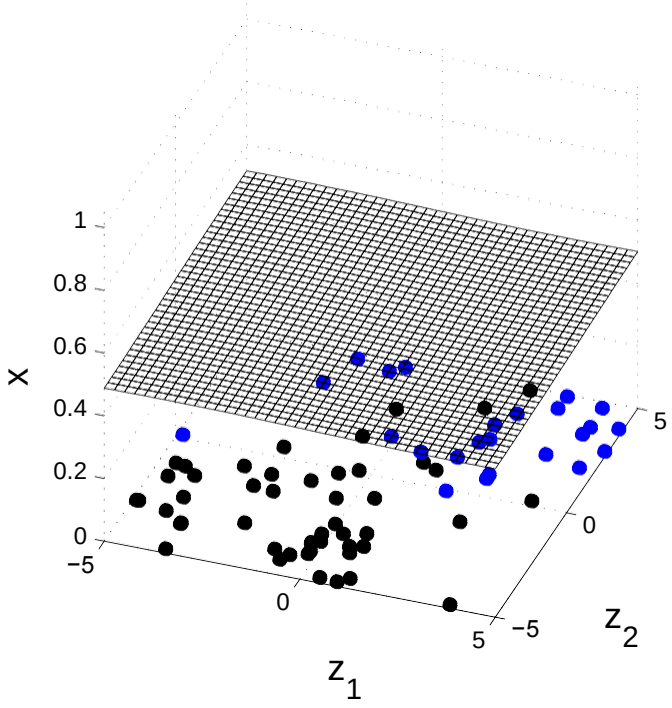
$$\frac{\mathrm{d}G(W,\boldsymbol{w})}{\mathrm{d}W_{ij}} = \sum_n \frac{\mathrm{d}G(W,\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}W_{ij}} = \sum_n \frac{\mathrm{d}G(W,\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}a^{(n)}}{\mathrm{d}W_{ij}}$$

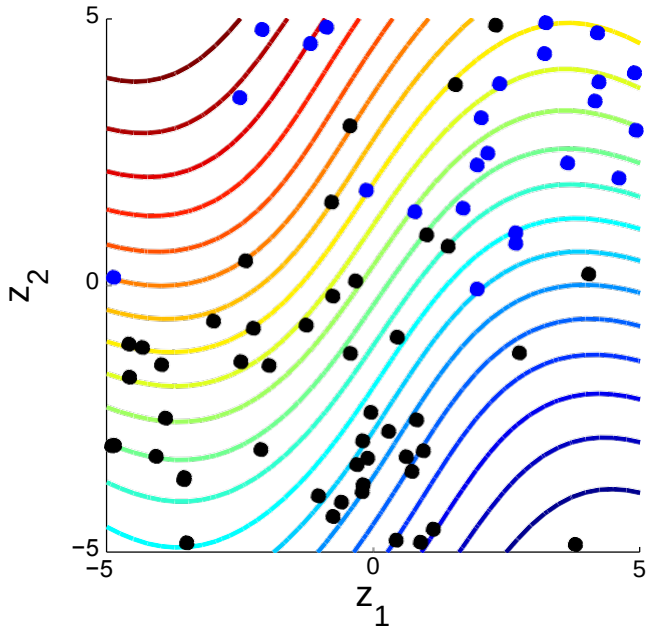
$$= \sum_n \frac{\mathrm{d}G(W,\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}a^{(n)}} \frac{\mathrm{d}a^{(n)}}{\mathrm{d}x_i^{(n)}} \frac{\mathrm{d}x_i^{(n)}}{\mathrm{d}W_{ij}} = \sum_n \frac{\mathrm{d}G(W,\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}a^{(n)}} \frac{\mathrm{d}a^{(n)}}{\mathrm{d}x_i^{(n)}} \frac{\mathrm{d}a_i^{(n)}}{\mathrm{d}W_{ij}}$$

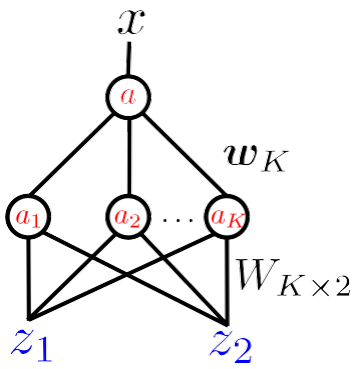
Training a Neural Network with a Single Hidden Layer



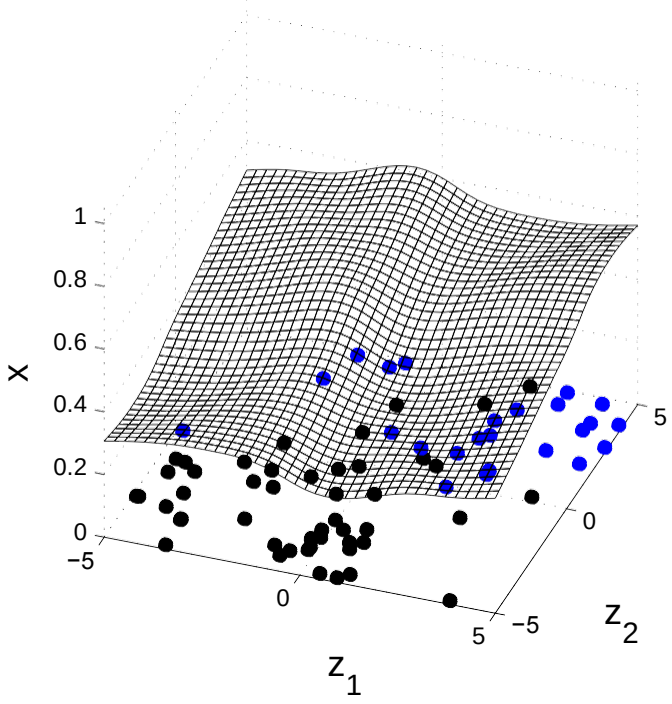
Training a Neural Network with a Single Hidden Layer

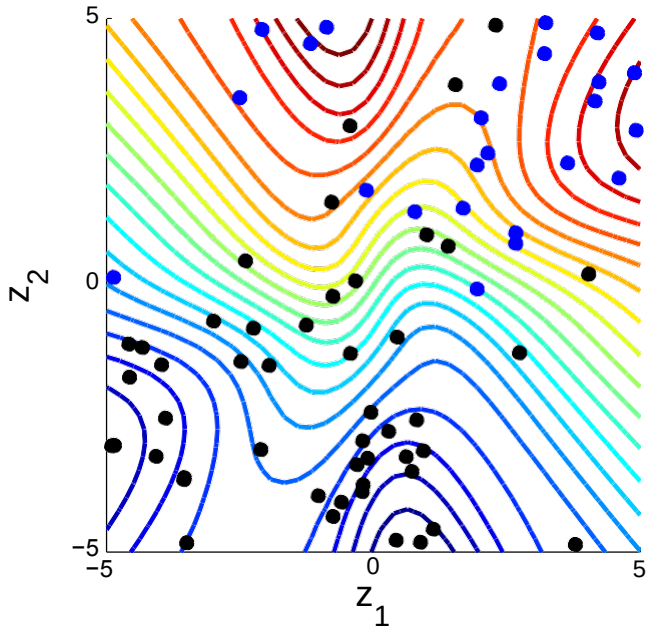


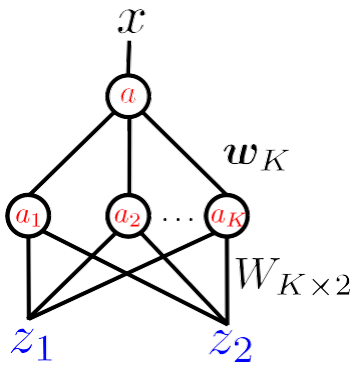




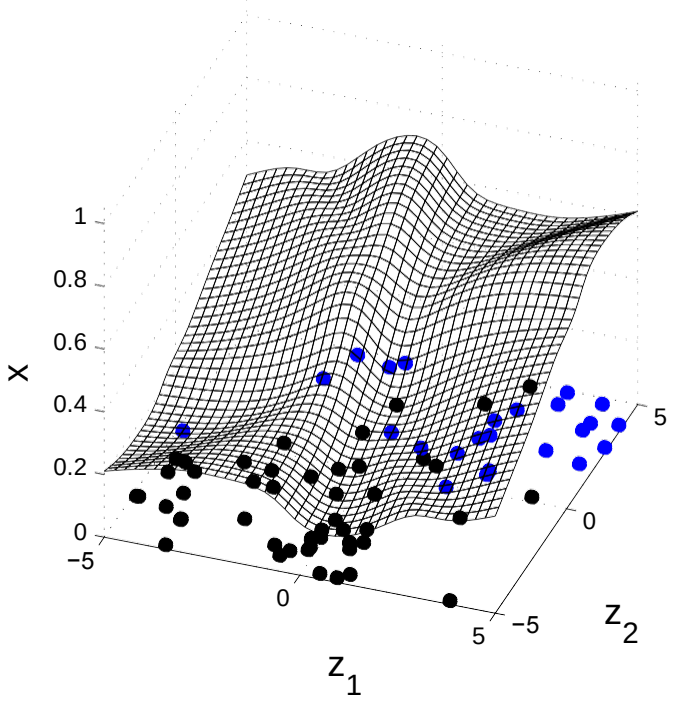
Training a Neural Network with a Single Hidden Layer

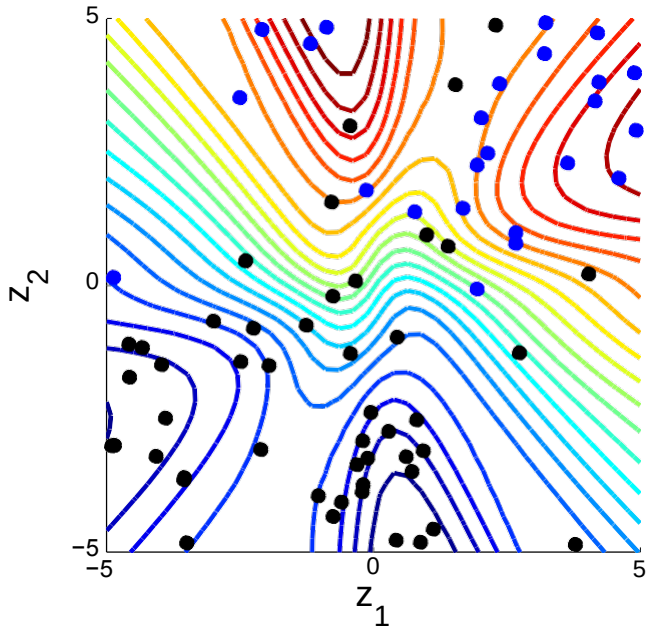


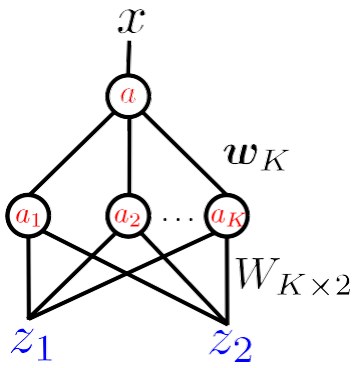




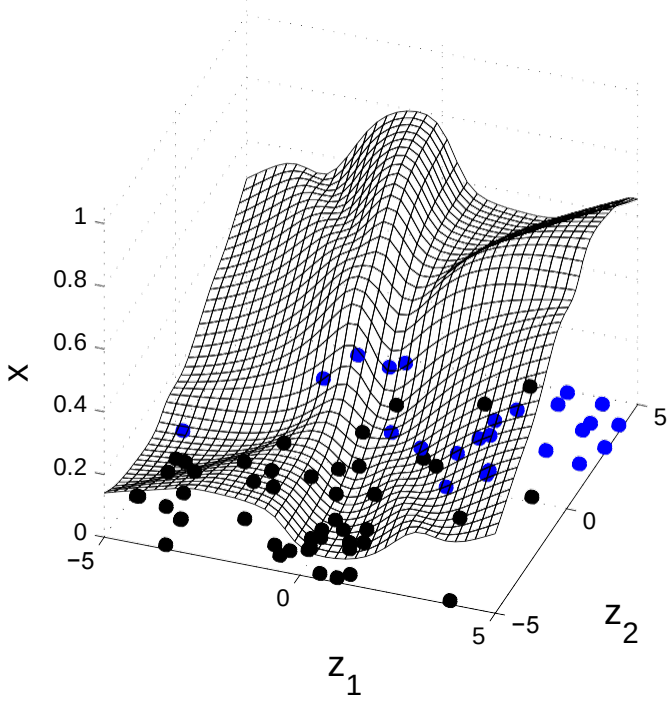
Training a Neural Network with a Single Hidden Layer

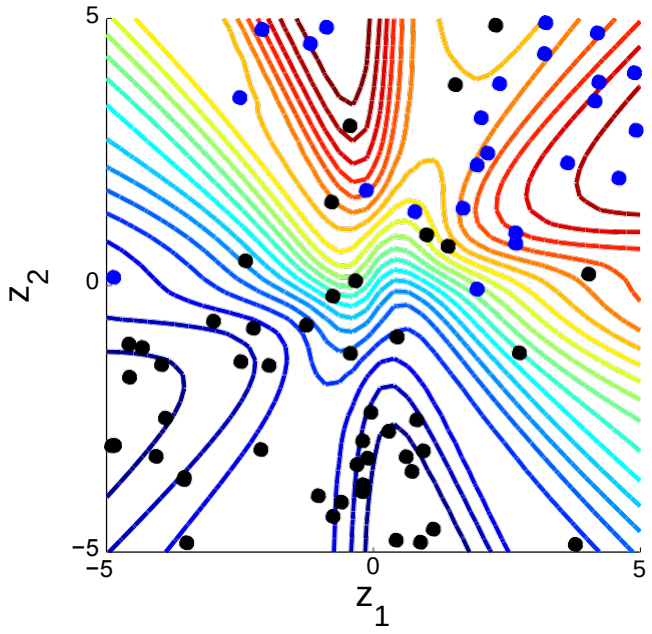


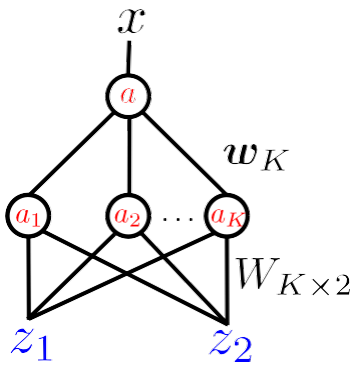




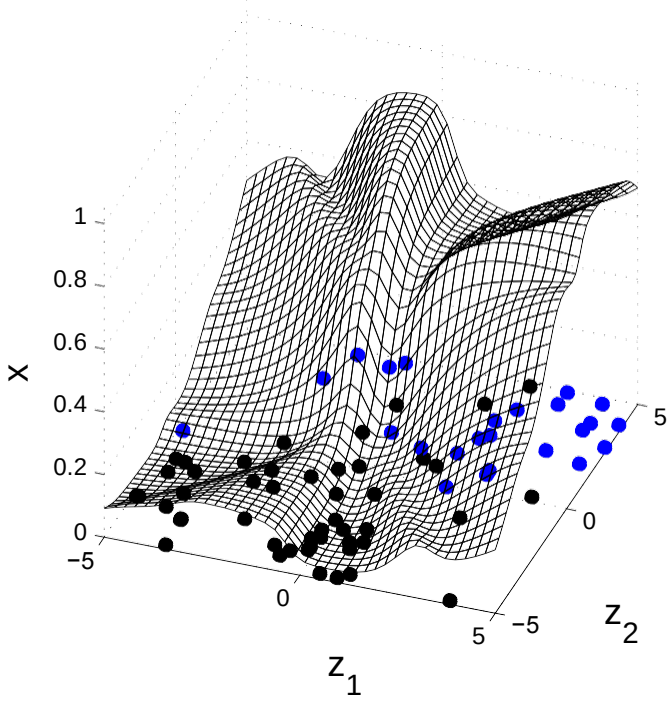
Training a Neural Network with a Single Hidden Layer

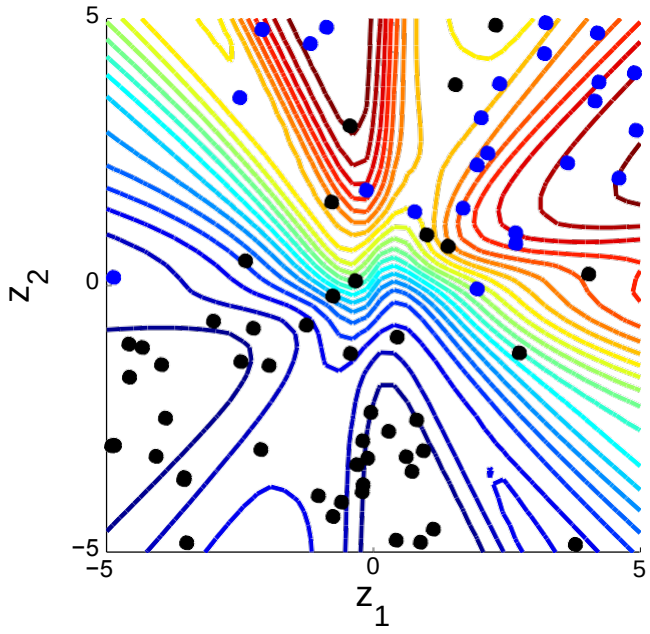


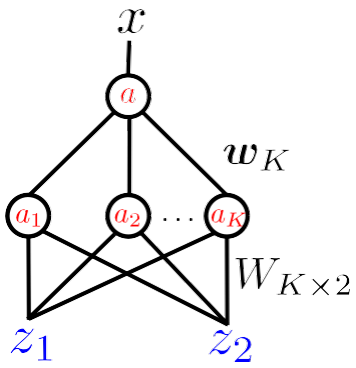




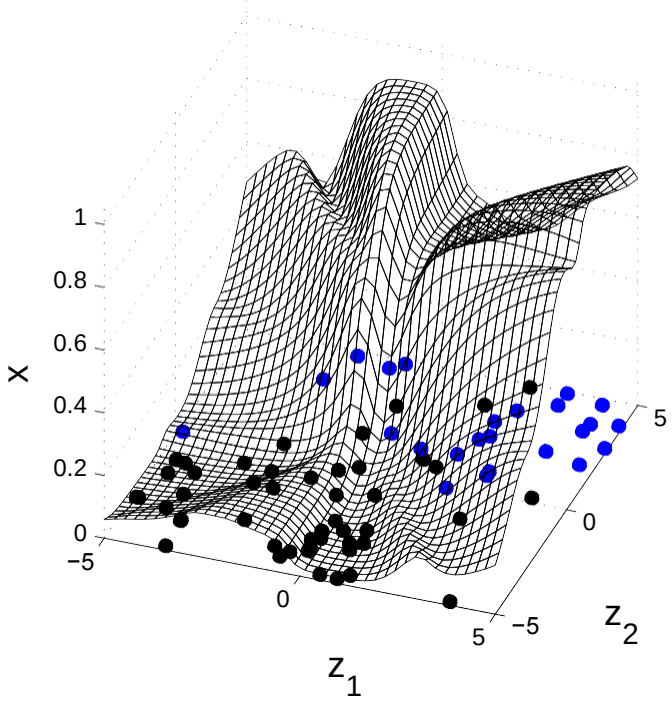
Training a Neural Network with a Single Hidden Layer

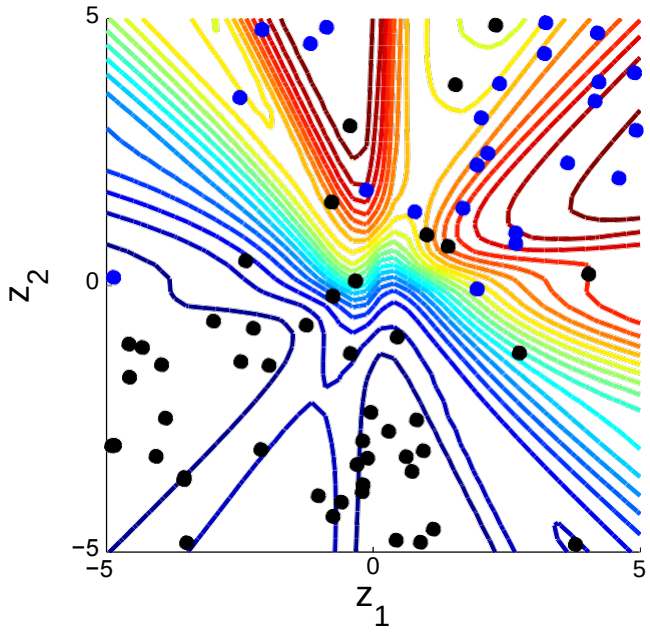


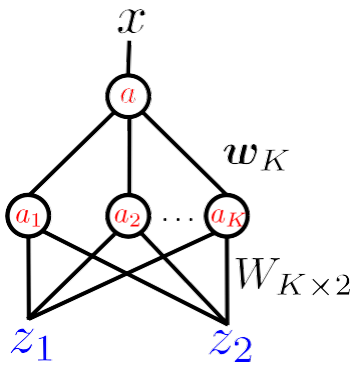




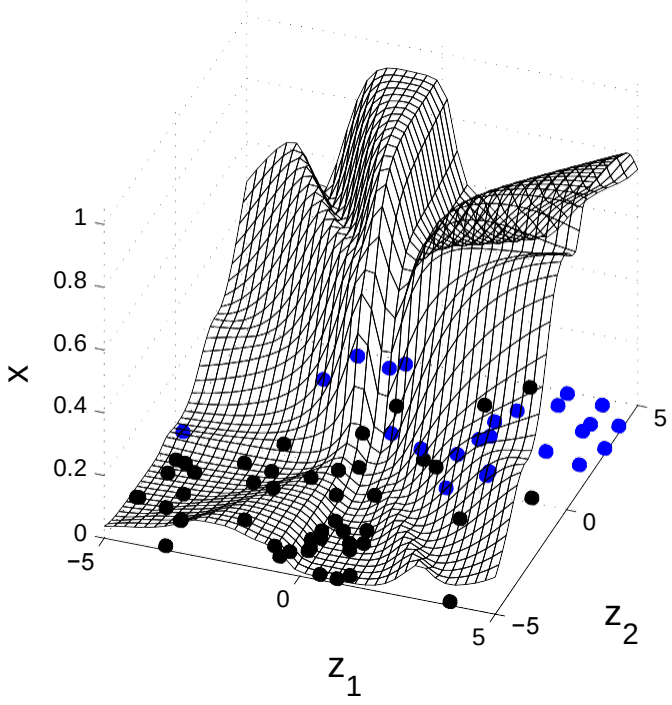
Training a Neural Network with a Single Hidden Layer

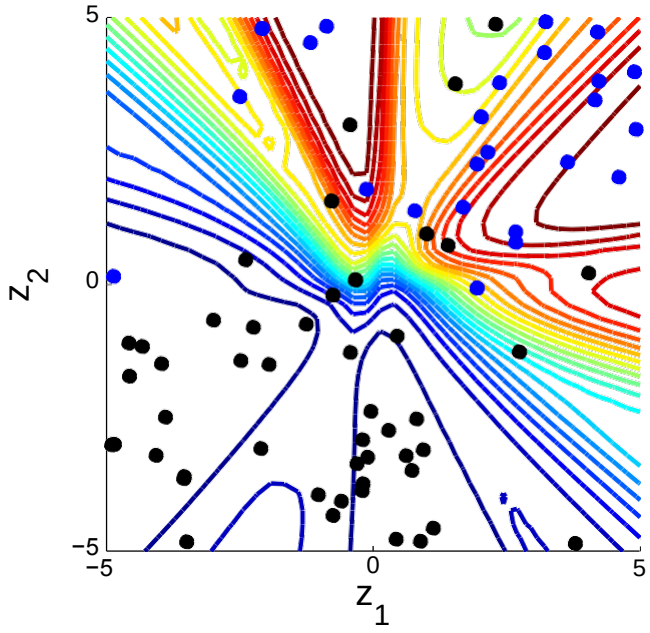


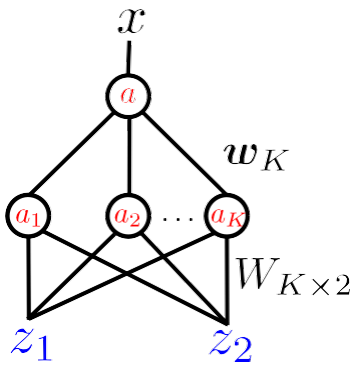




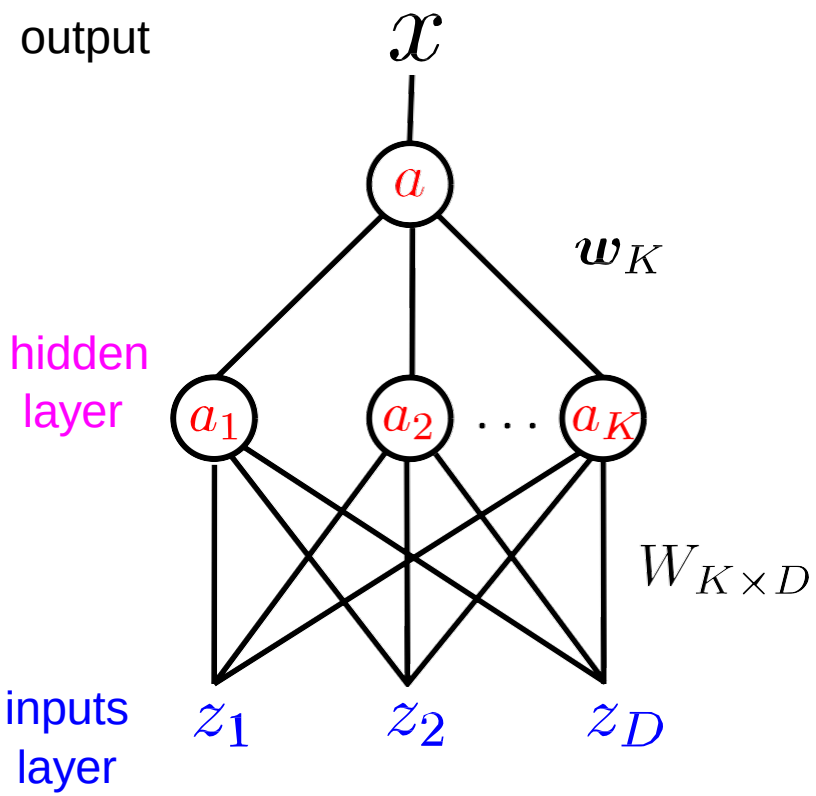
Training a Neural Network with a Single Hidden Layer

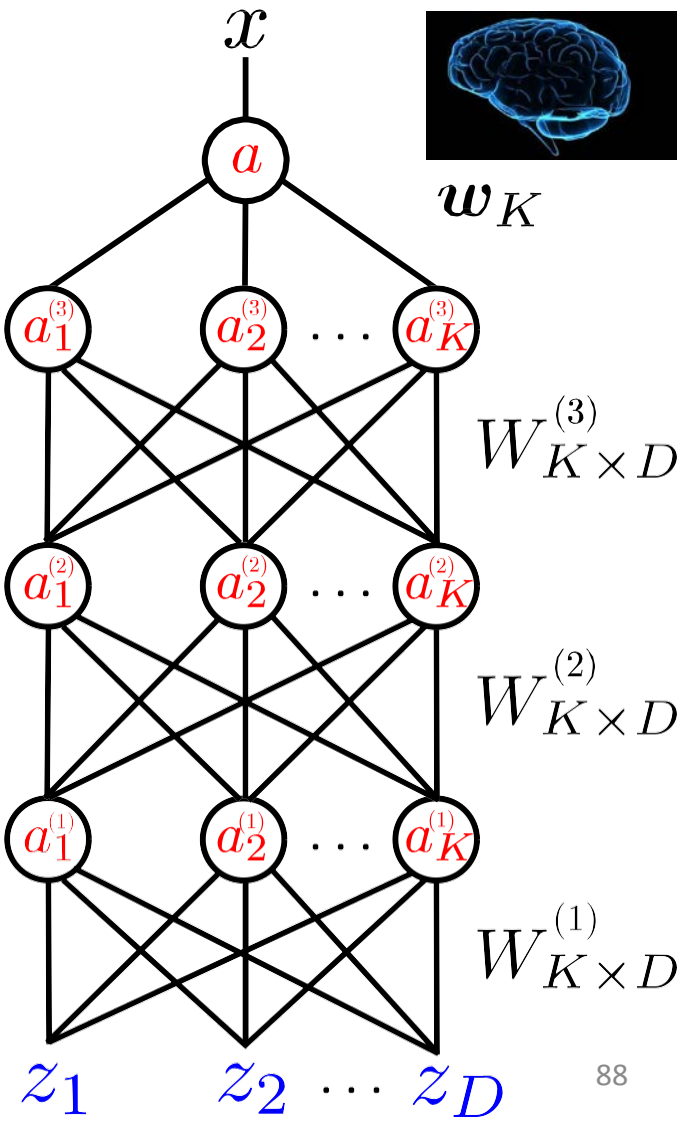






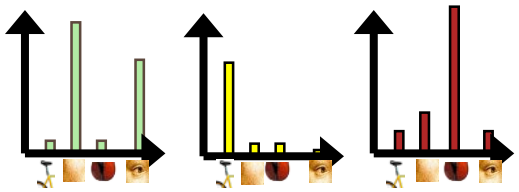
Hierarchical Models with Many Layers

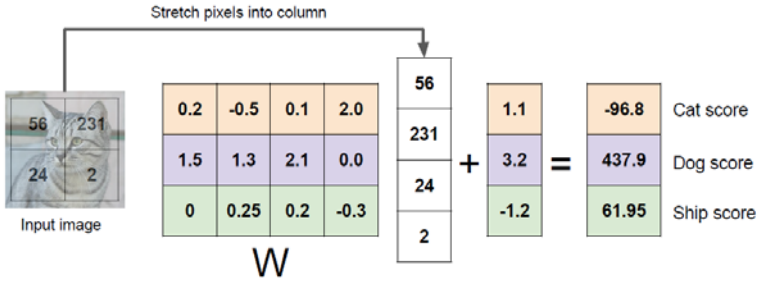


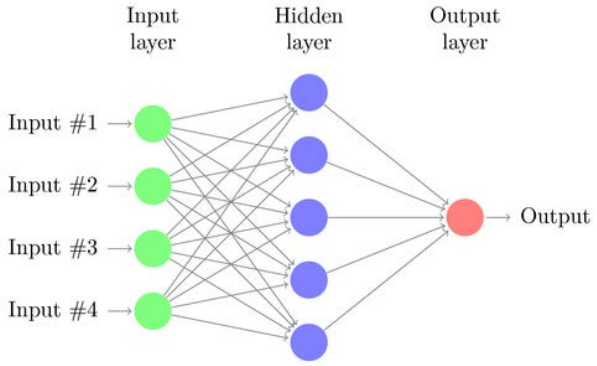


What's to Be Covered Today…

- General Framework for Visual Classification
  - Bag-of-Words Models as Image Features
- Intro to Neural Networks & CNN
  - Linear Classification
  - Neural Network for Machine Vision
  - Multi-Layer Perceptron
  - Convolutional Neural Networks



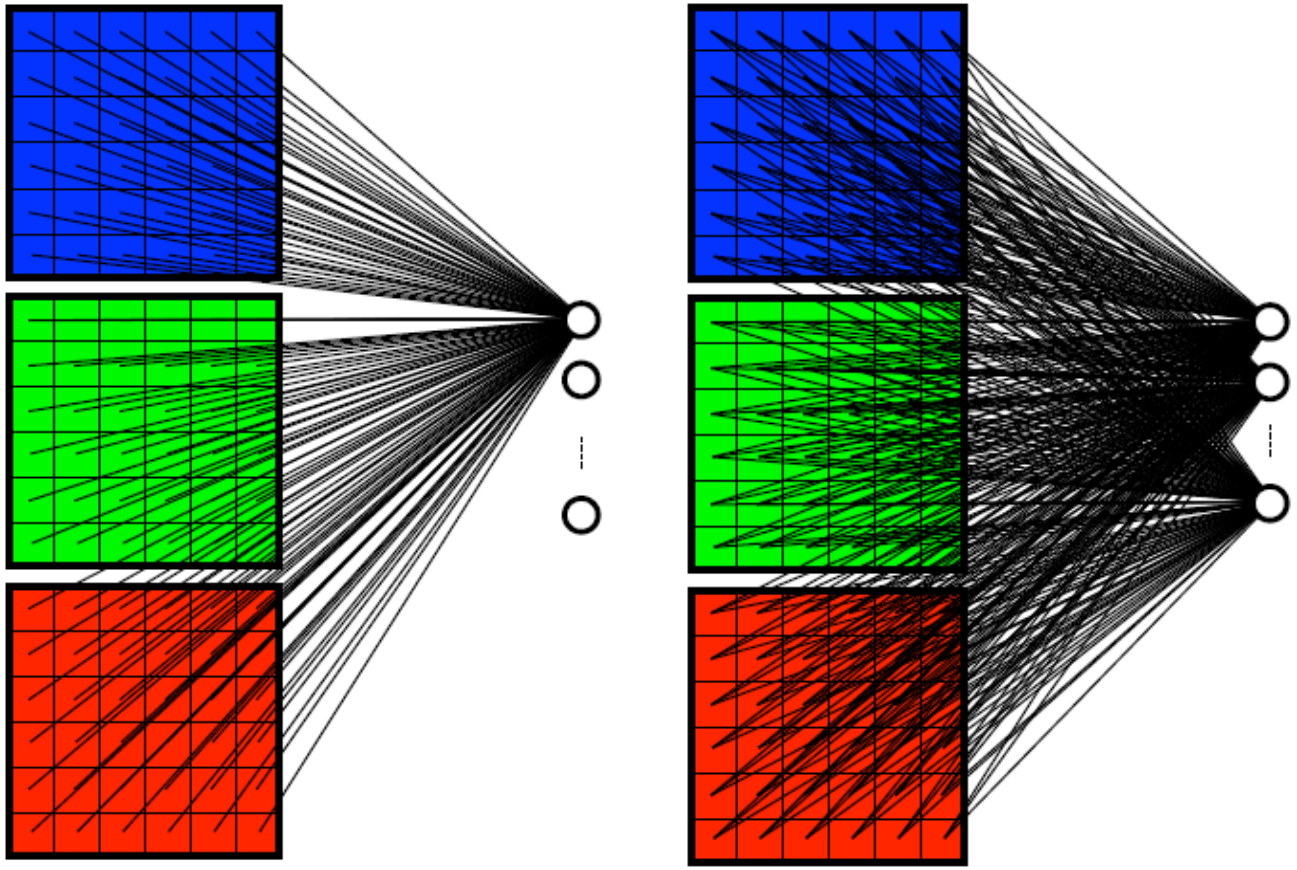




Many slides from Richard Turner, Fei-Fei Li, Yaser Sheikh, Simon Lucey, Kaiming He, and J.-B. Huang

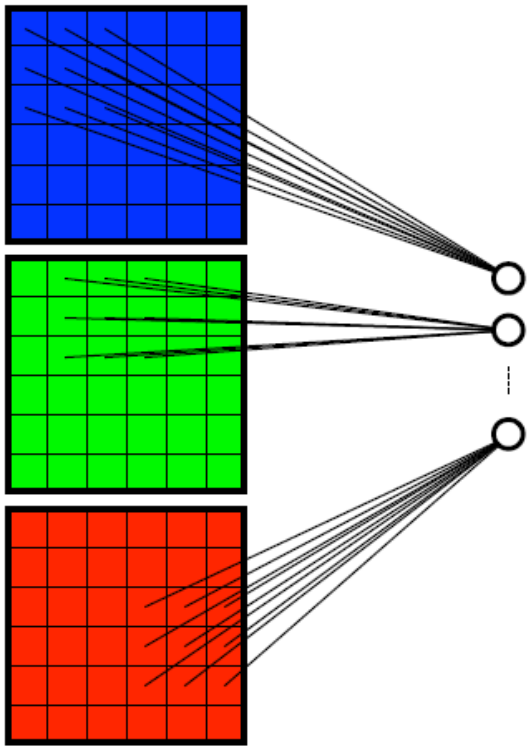
Convolutional Neural Networks

- How many weights for MLPs for images?



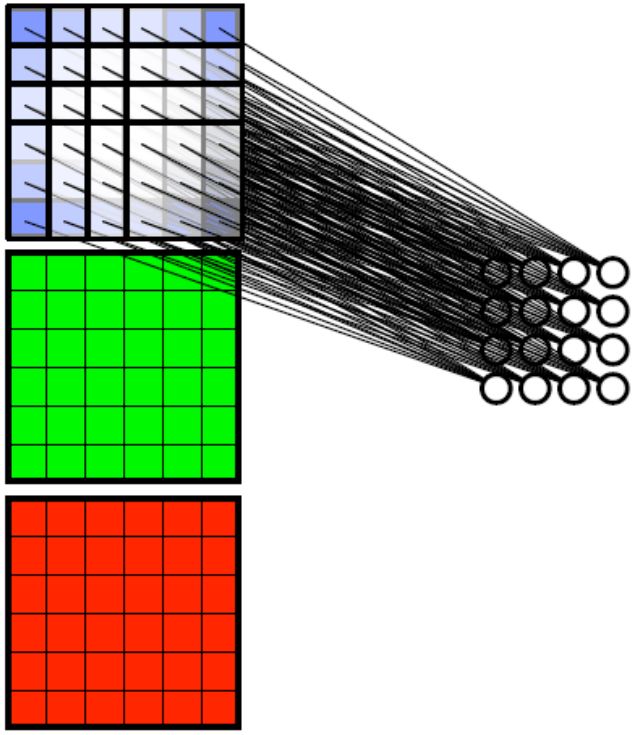
Convolutional Neural Networks

- Property I of CNN: Local Connectivity
  - Each neuron takes info only from a neighborhood of pixels.

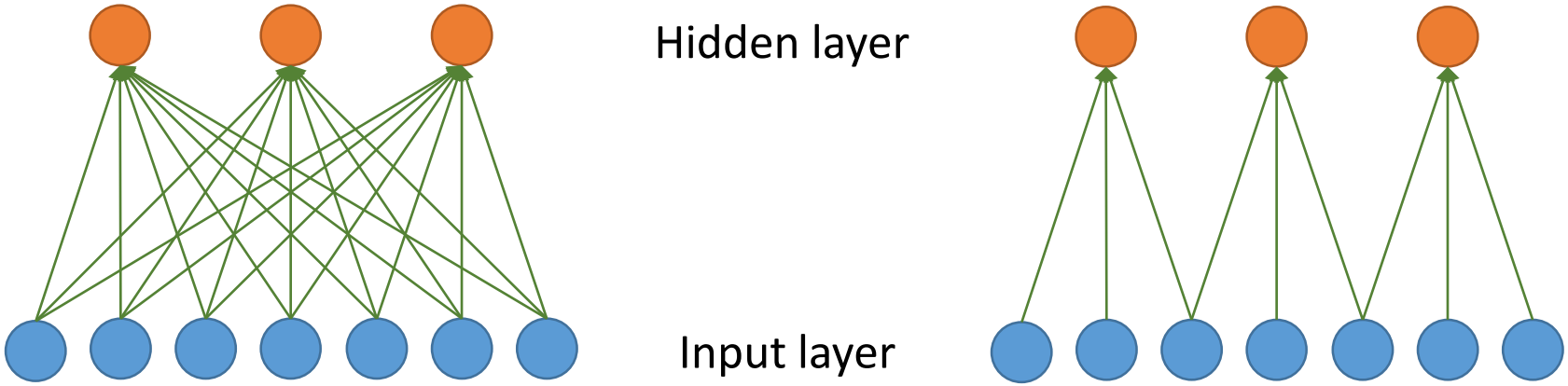


Convolutional Neural Networks

- Property II of CNN: Weight Sharing
  - Neurons connecting all neighborhoods have identical weights.



CNN: Local Connectivity

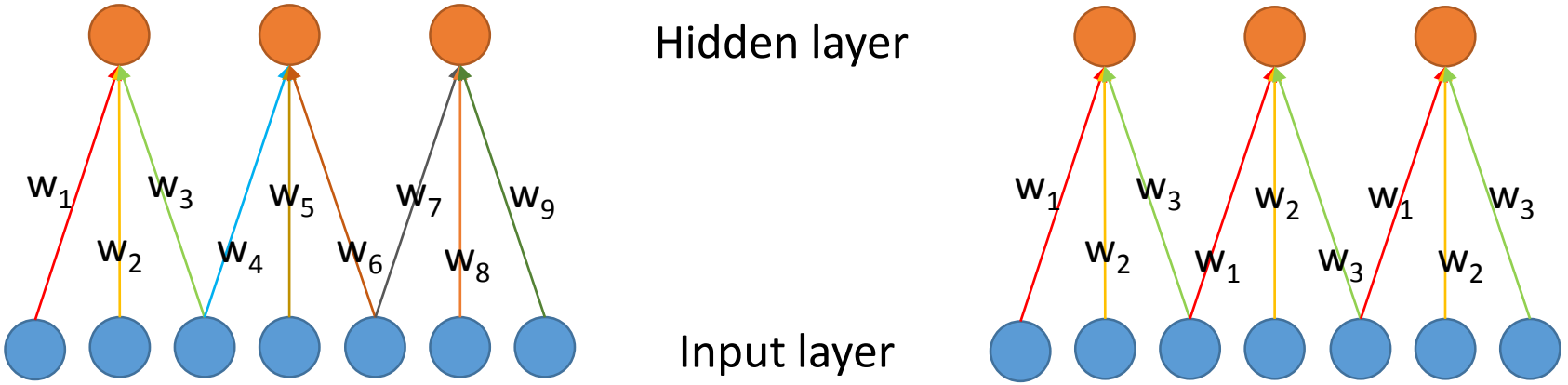


Global connectivity

Local connectivity

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Global connectivity:
  - Local connectivity:

CNN: Weight Sharing

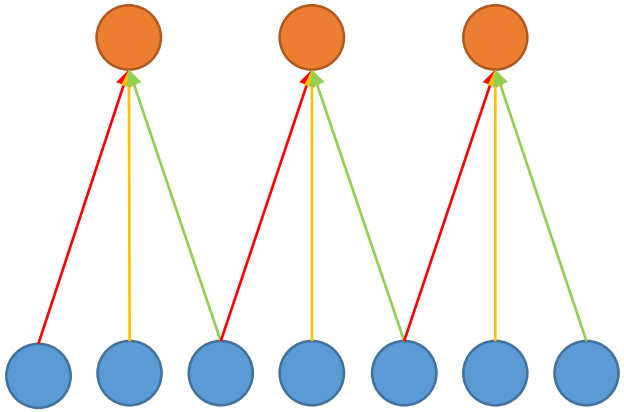


Without weight sharing

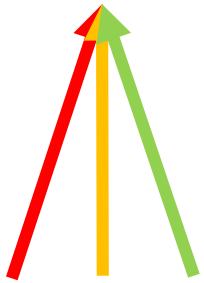
With weight sharing

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Without weight sharing:
  - With weight sharing :

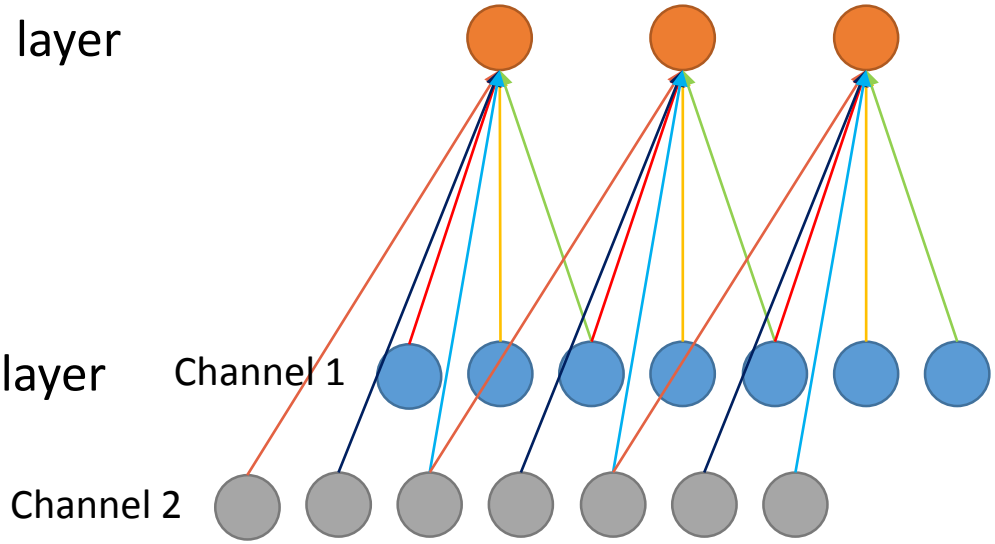
CNN with Multiple Input Channels



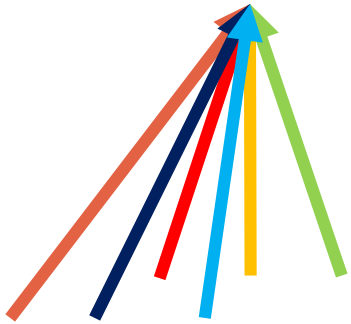
Single input channel



Filter weights

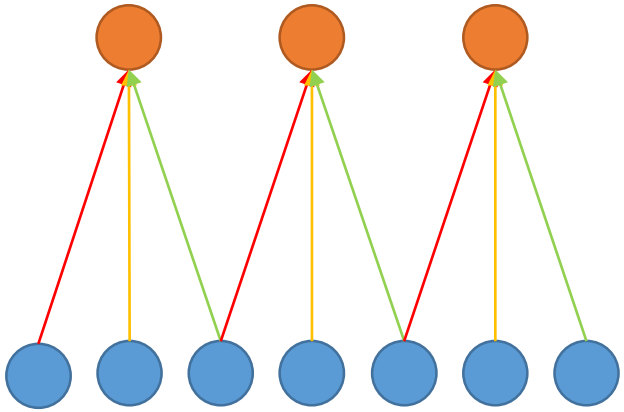


Multiple input channels



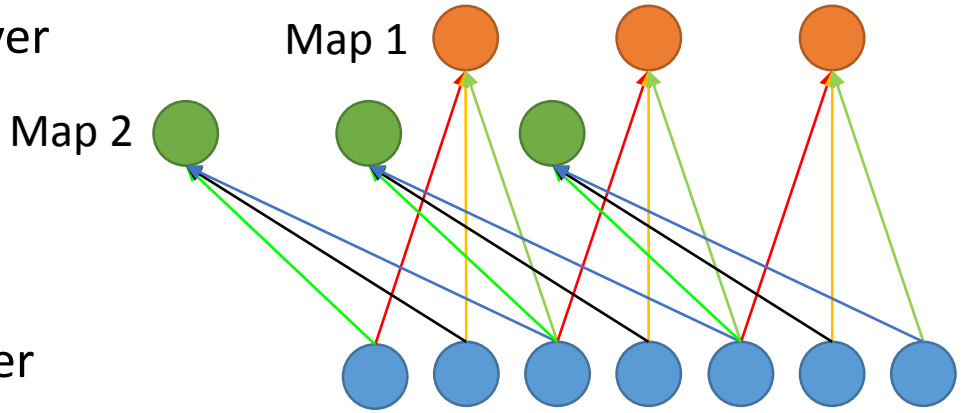
Filter weights

CNN with Multiple Output Maps

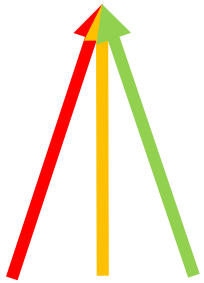


Hidden layer

Input layer

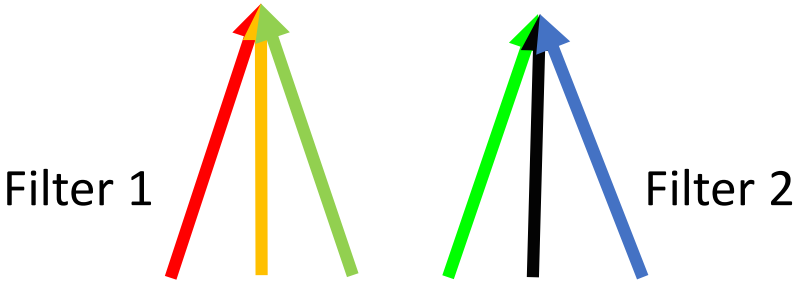


Single output map



Filter weights

Multiple output maps



Filter weights

Putting them together

- Local connectivity
- Weight sharing
- Handling multiple input channels
- Handling multiple output maps

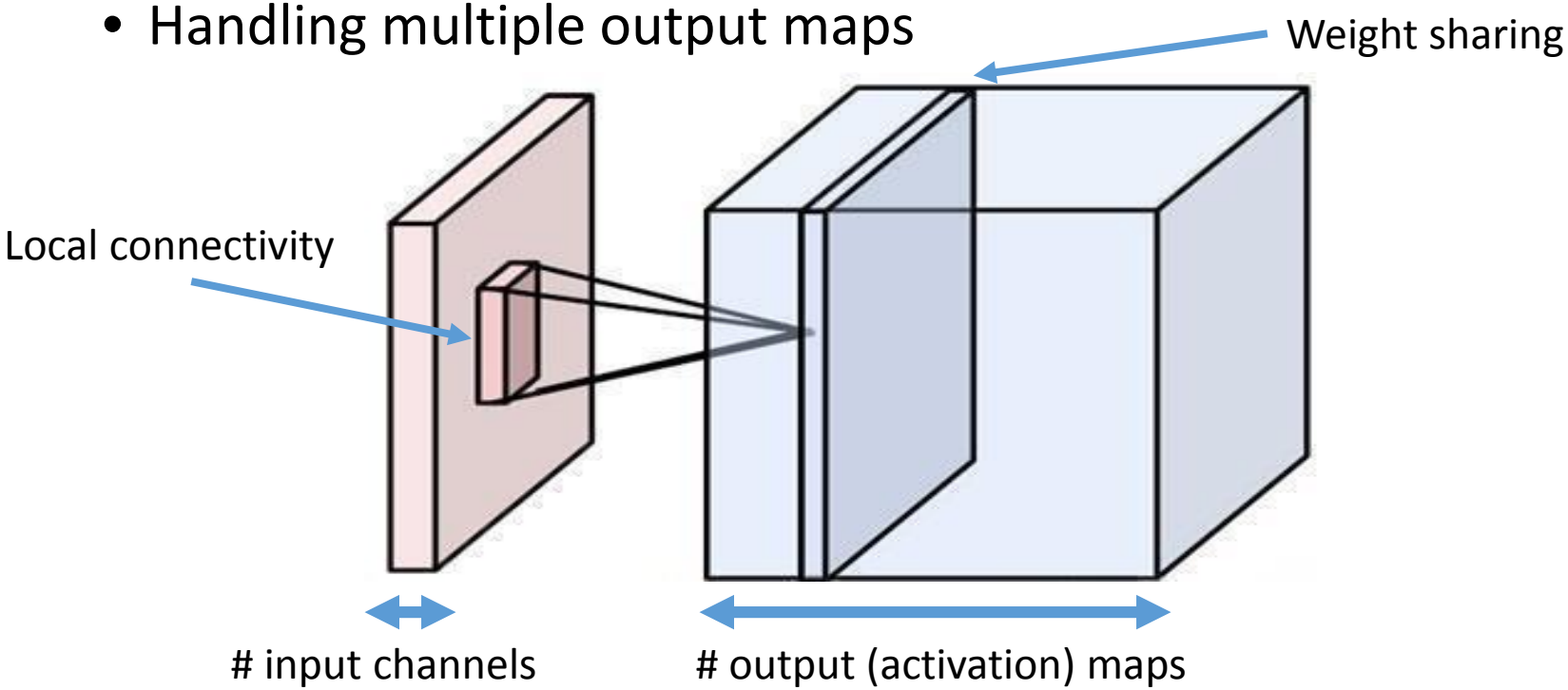
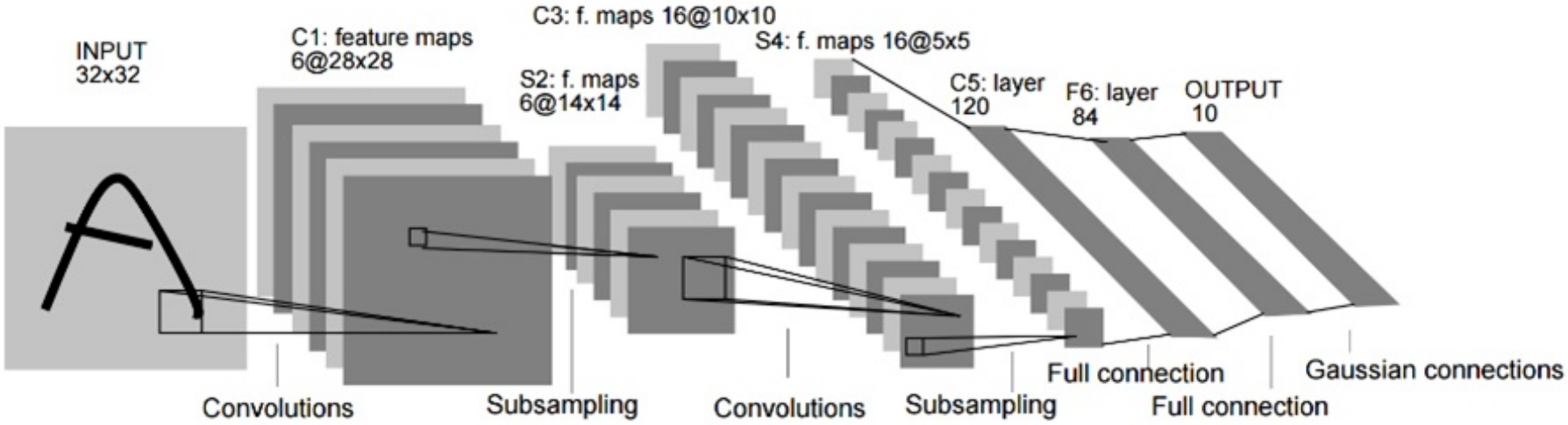
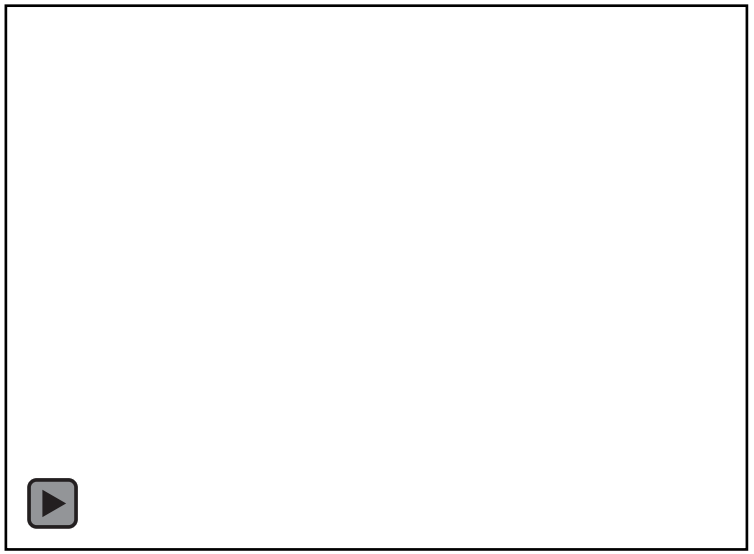


Image credit: A. Karpathy

LeNet [LeCun et al. 1998]

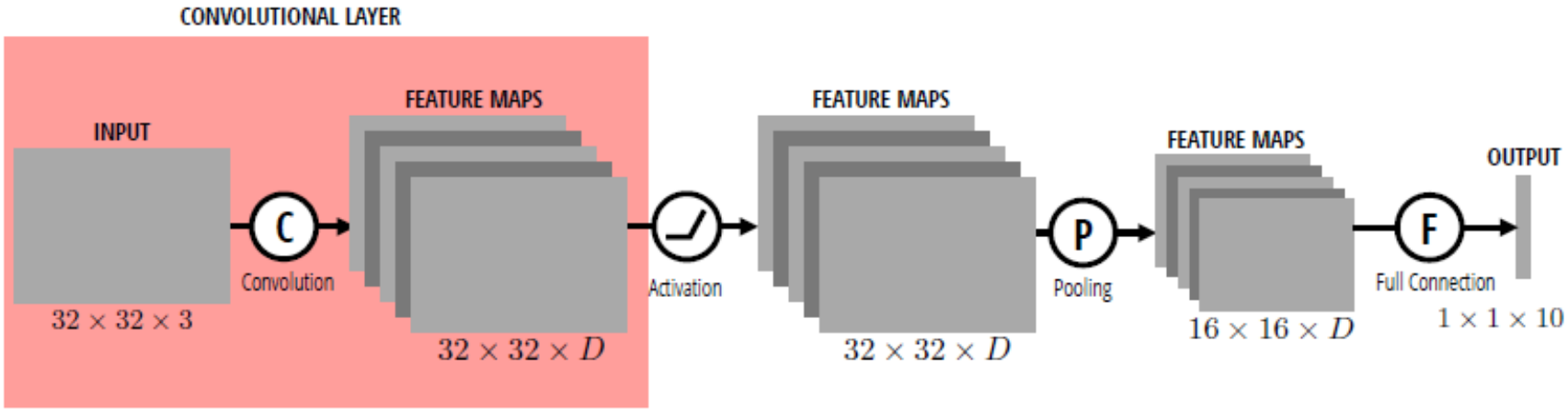


Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]



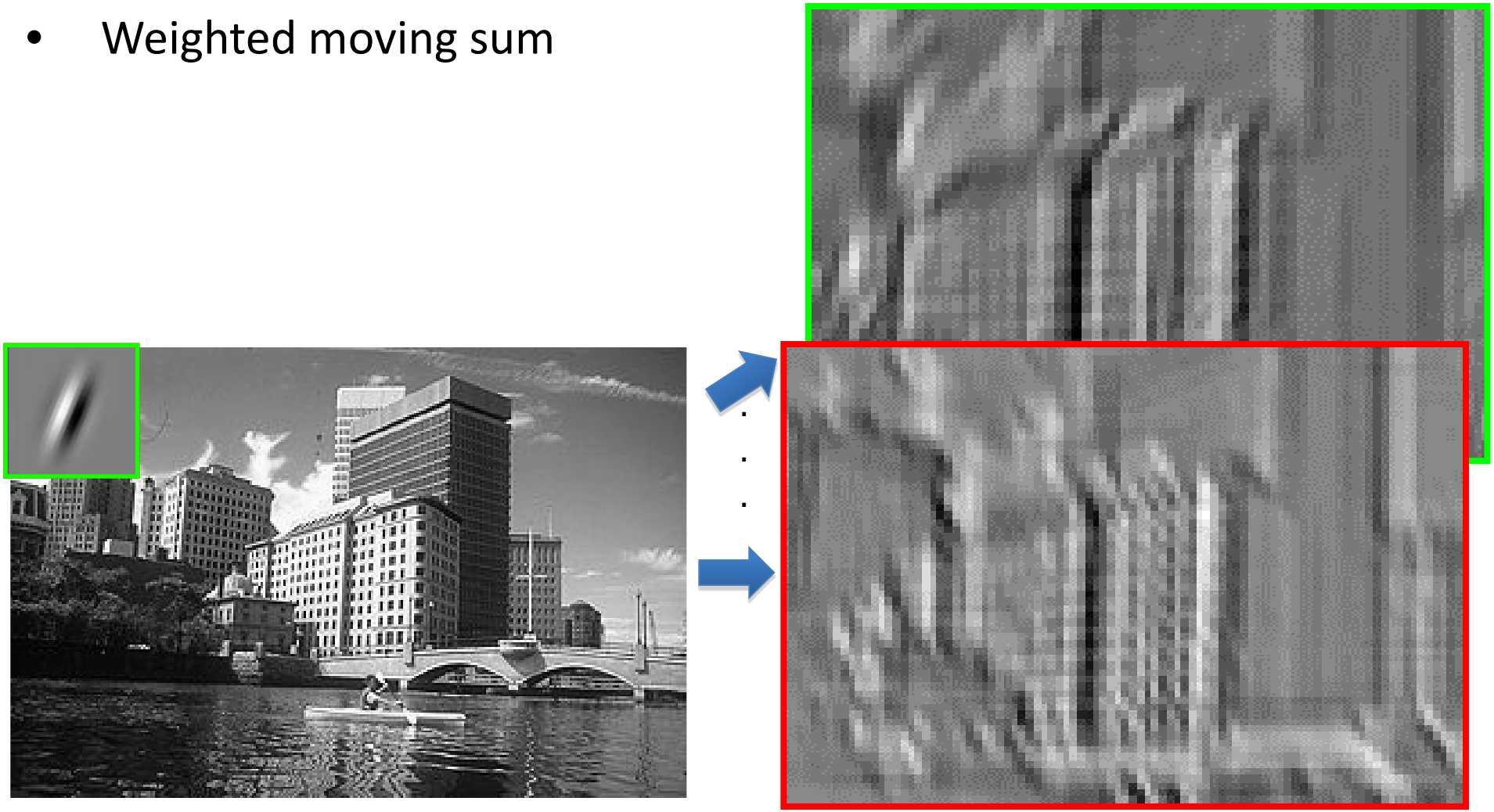
LeNet-1 from 1993

Convolution Layer in CNN



What is a Convolution?

- Weighted moving sum

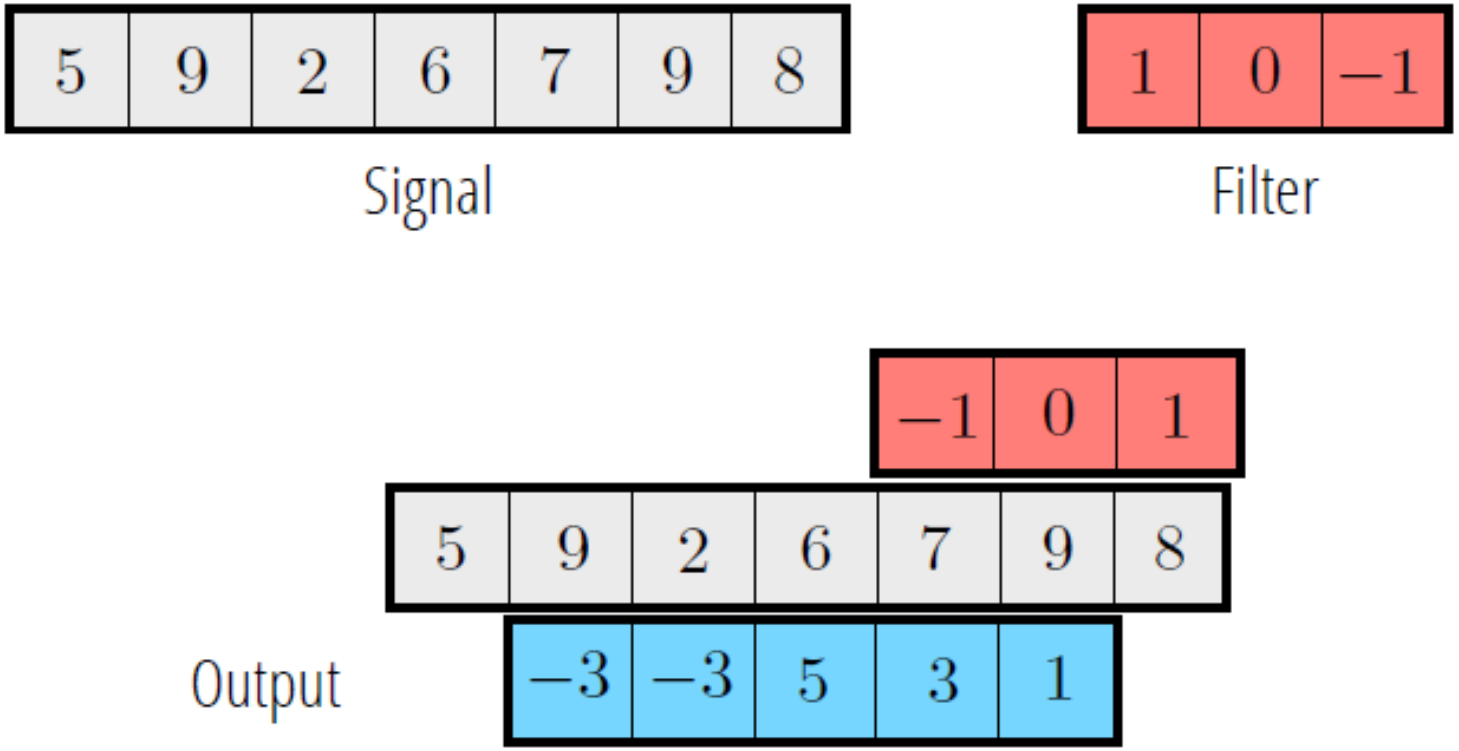


Input

Feature Activation Map

slide credit: S. Lazebnik

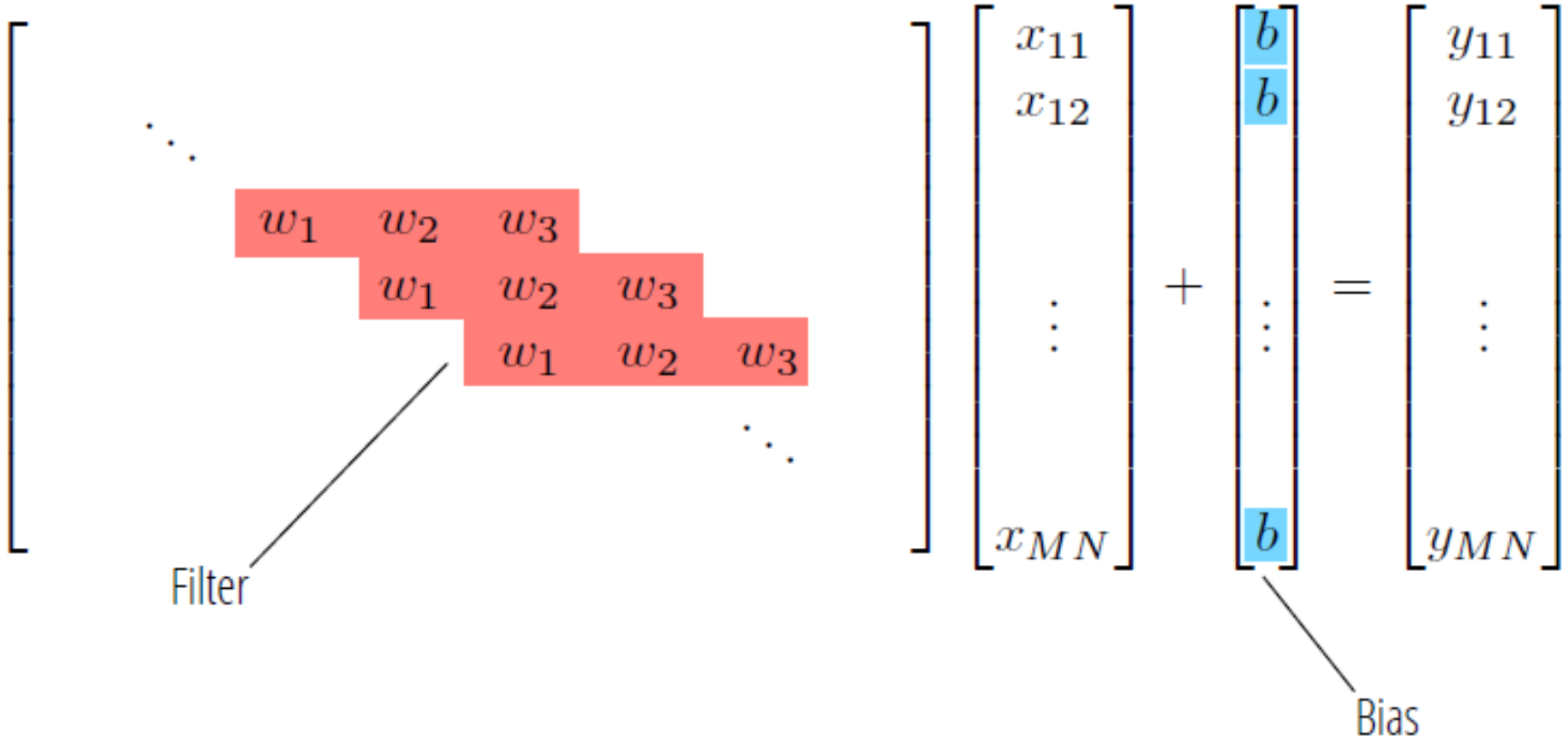
What is a Convolution?



Convolution is a local linear operator

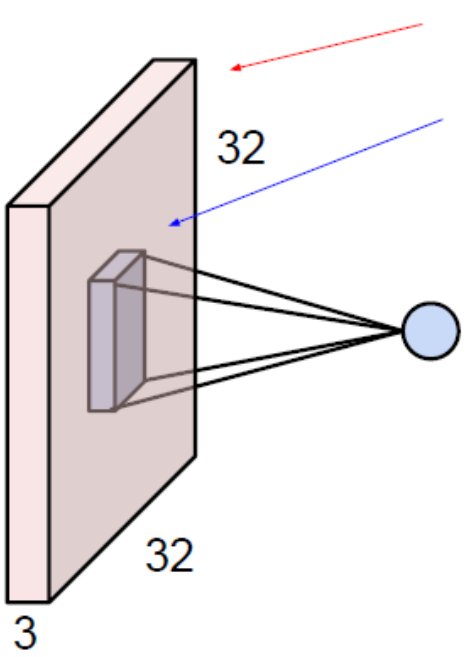
What is a Convolution?

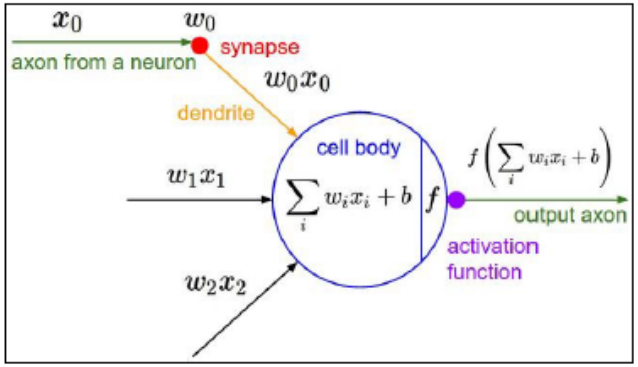
- Toeplitz Matrix Form



Putting them together (cont'd)

- The brain/neuron view of CONV layer



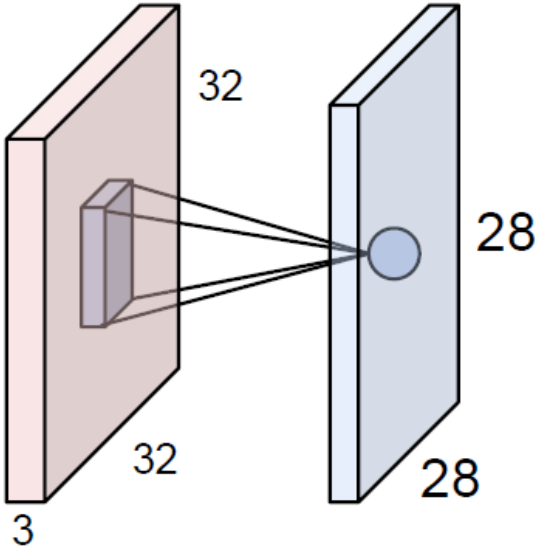


It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

Putting them together (cont'd)

- The brain/neuron view of CONV layer



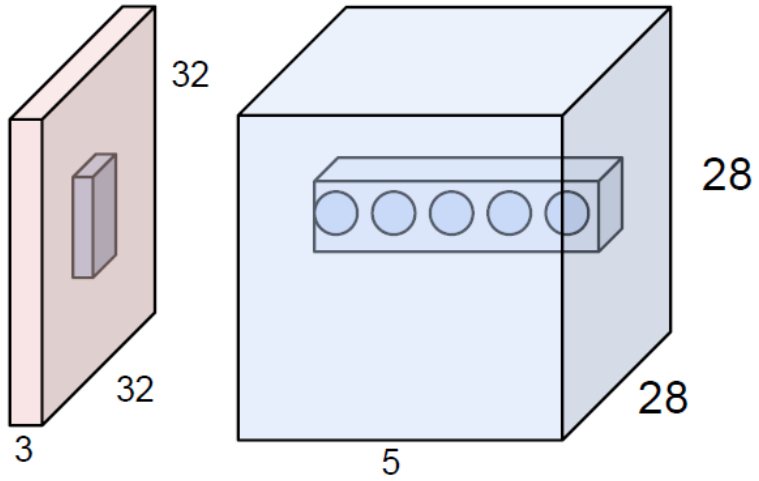
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

Putting them together (cont'd)

- The brain/neuron view of CONV layer

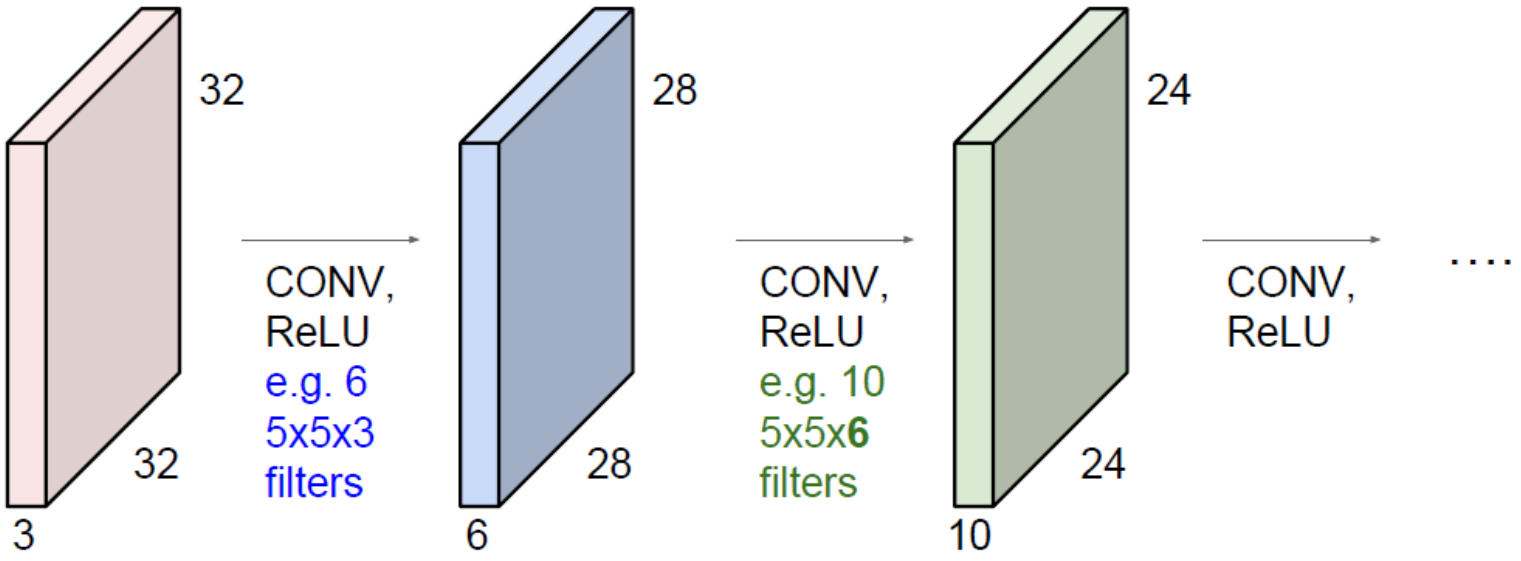


E.g. with 5 filters,
CONV layer consists of
neurons arranged in a 3D grid
(28x28x5)

There will be 5 different
neurons all looking at the same
region in the input volume

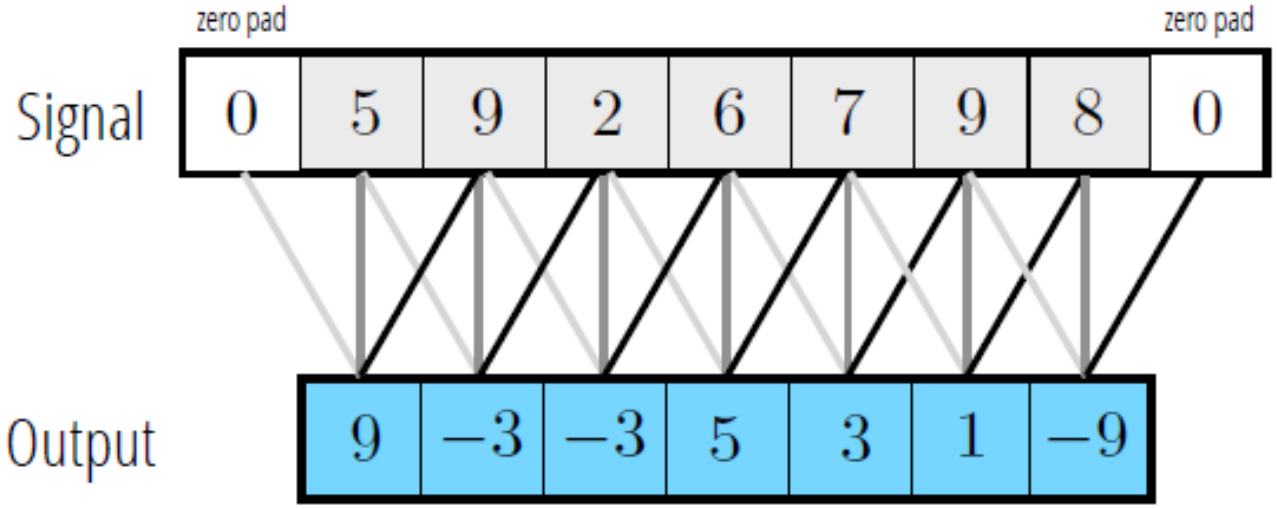
Putting them together (cont'd)

- Image input with 32 x 32 pixels convolved repeatedly with 5 x 5 x 3 filters shrinks volumes spatially (32 -> 28 -> 24 -> …).



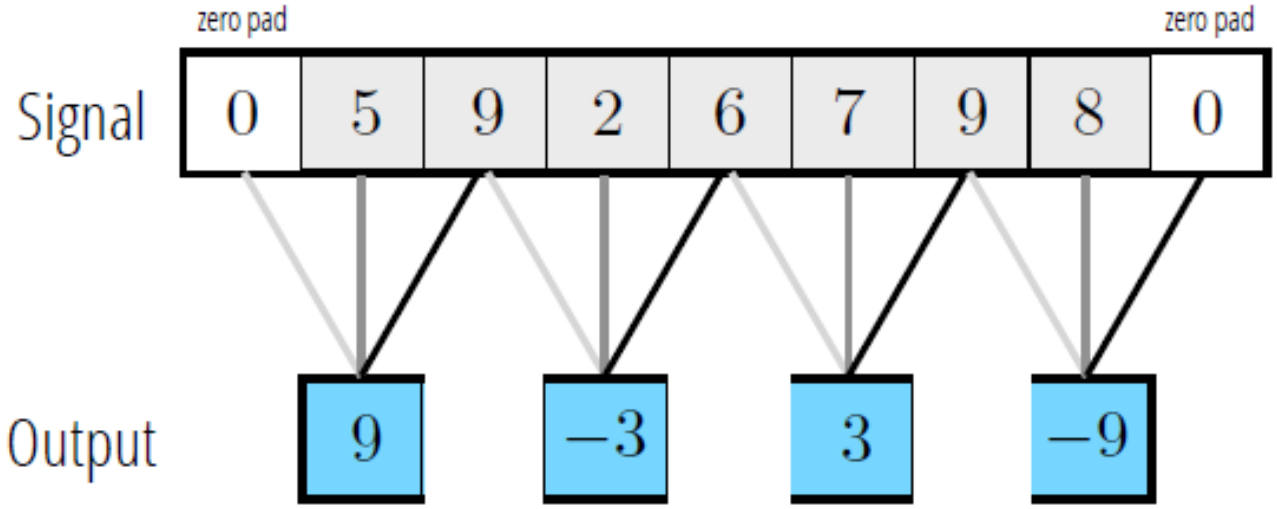
What is a Convolution?

- Zero Padding
  - Output is the same size as input (doesn't shrink as the network gets deeper).



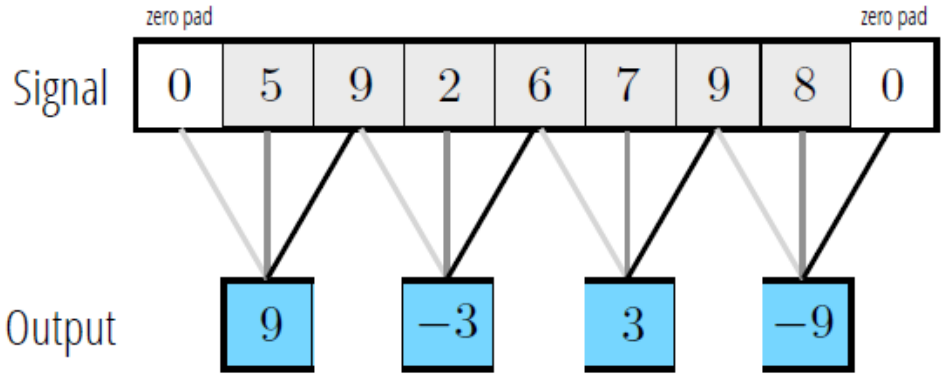
What is a Convolution?

- Stride
  - Step size across signals



What is a Convolution?

- Stride
  - Step size across signals



Input Size

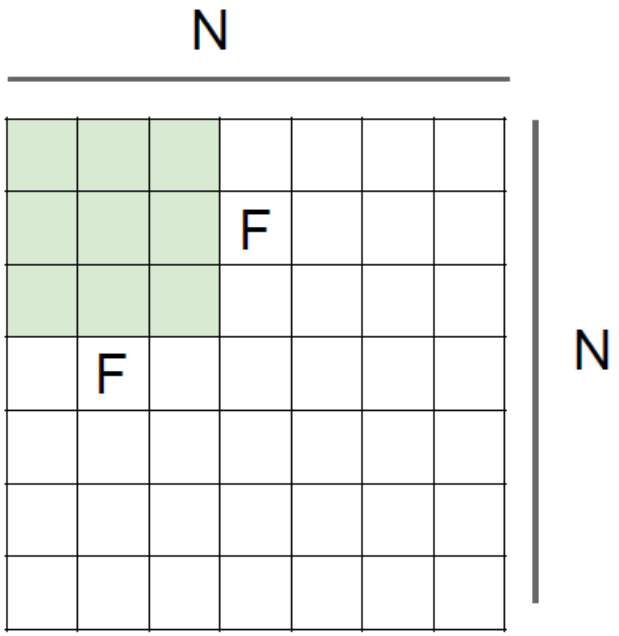
Filter Size

$$\text{Output Size} = \frac{N - c}{s} + 1$$

Stride: step size across the signal

What is a Convolution?

- Stride
  - Step size across signals

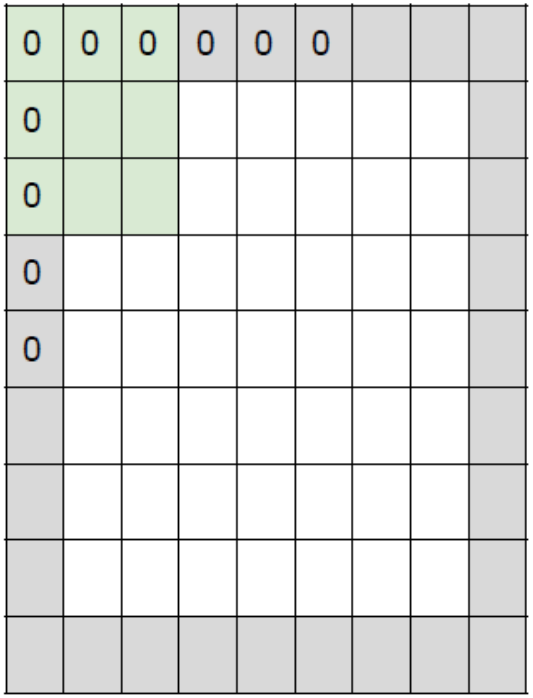


Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

What is a Convolution?

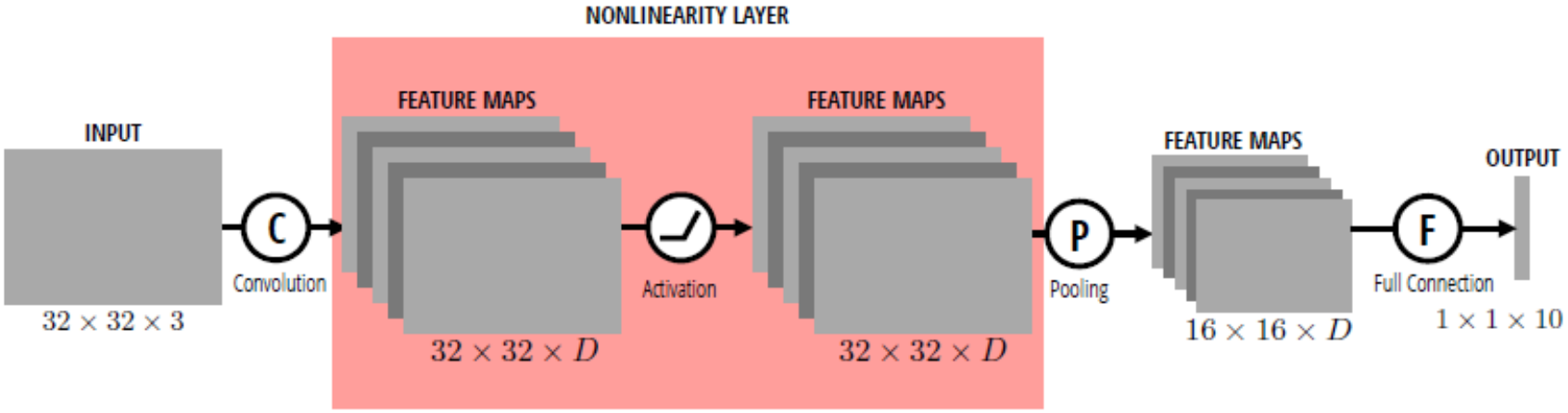
- Zero Padding + Stride



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

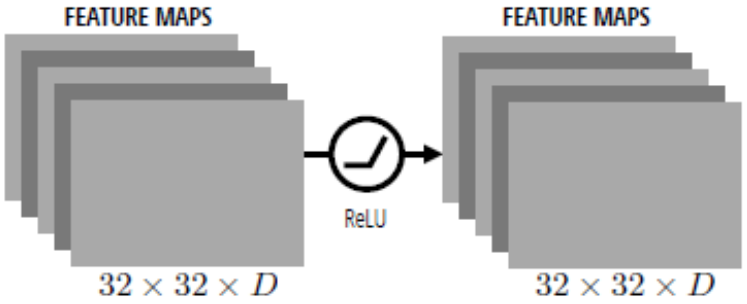
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

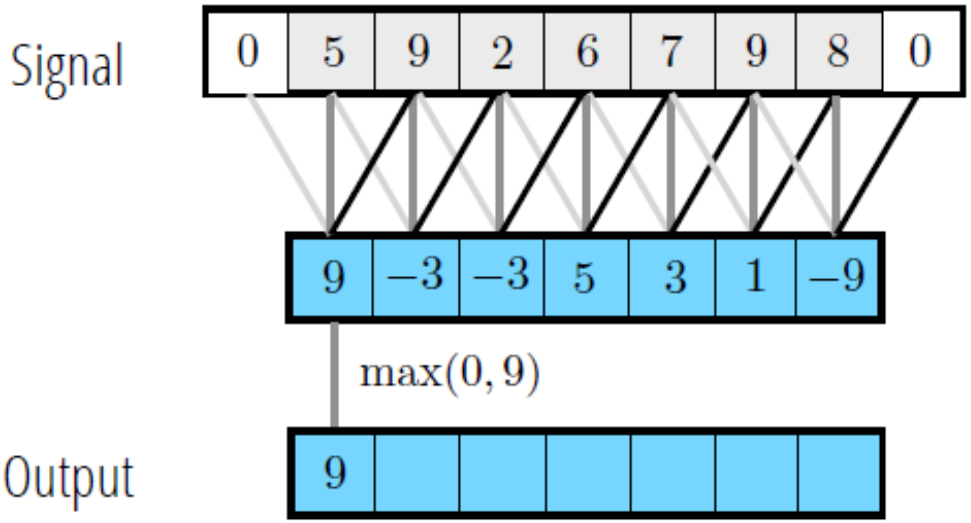
Nonlinearity Layer in CNN



Nonlinearity Layer

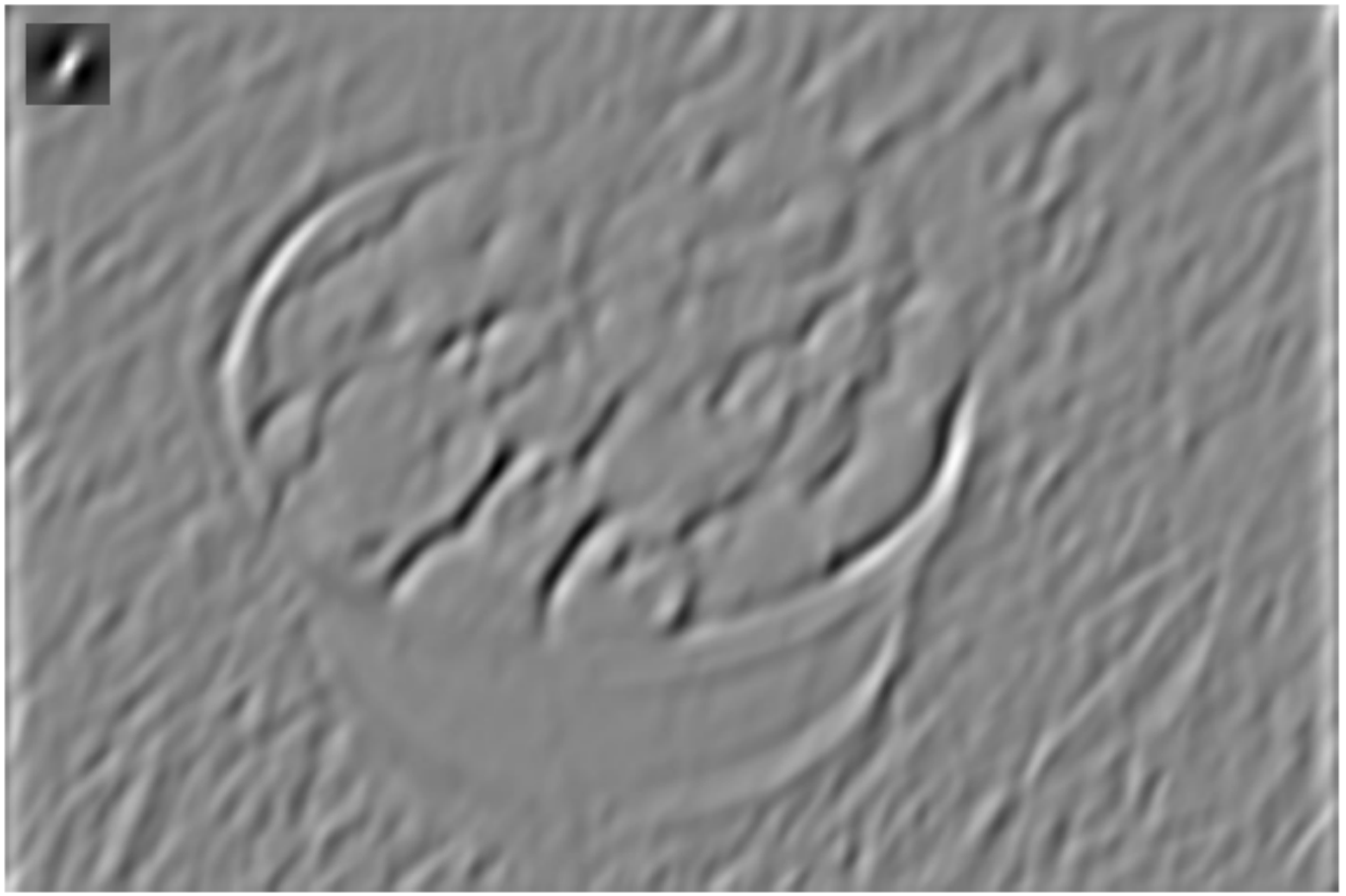
- E.g., ReLU (Rectified Linear Unit)
  - Pixel by pixel computation of max(0, x)





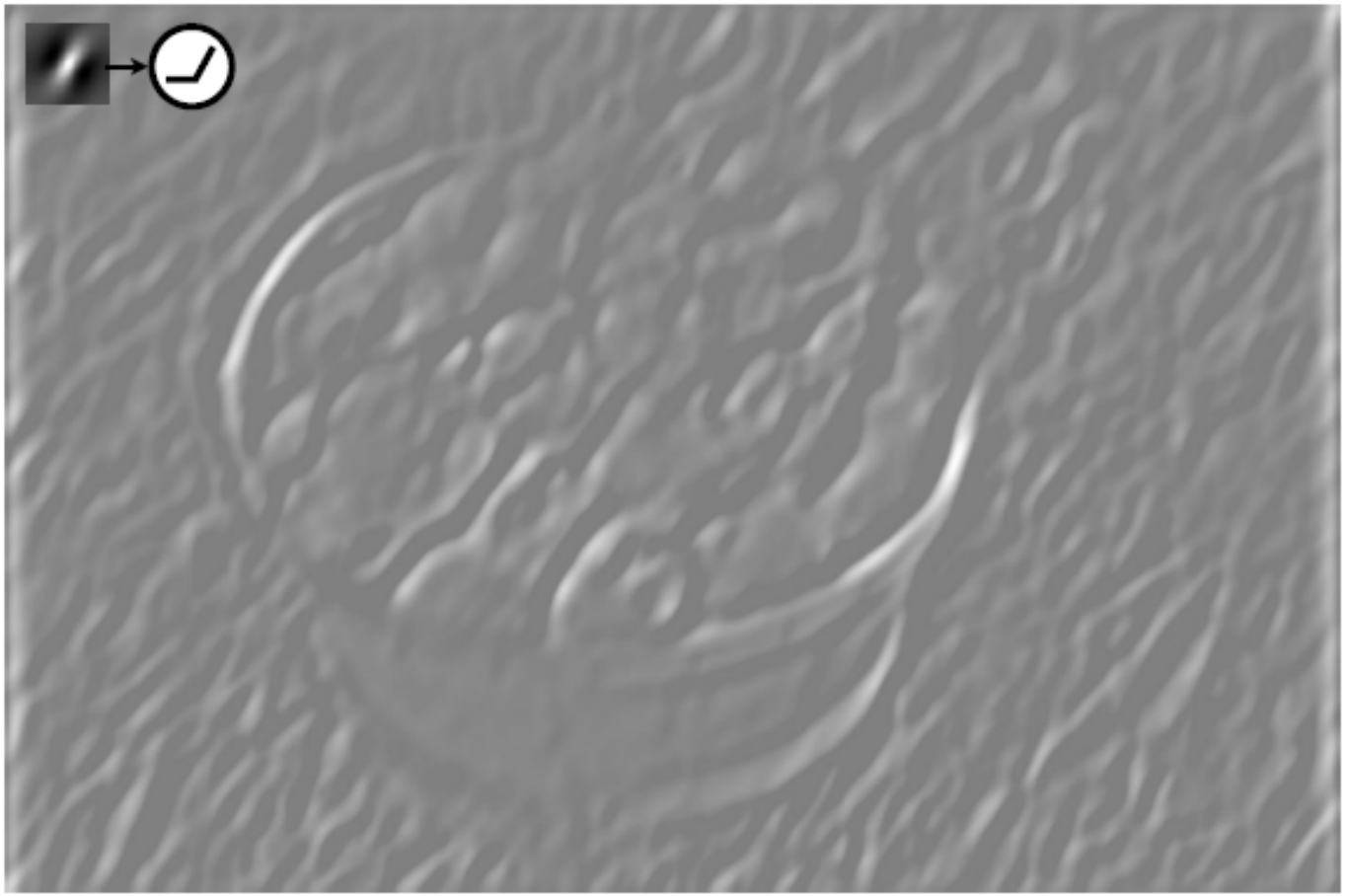
Nonlinearity Layer

- E.g., ReLU (Rectified Linear Unit)
  - Pixel by pixel computation of max(0, x)

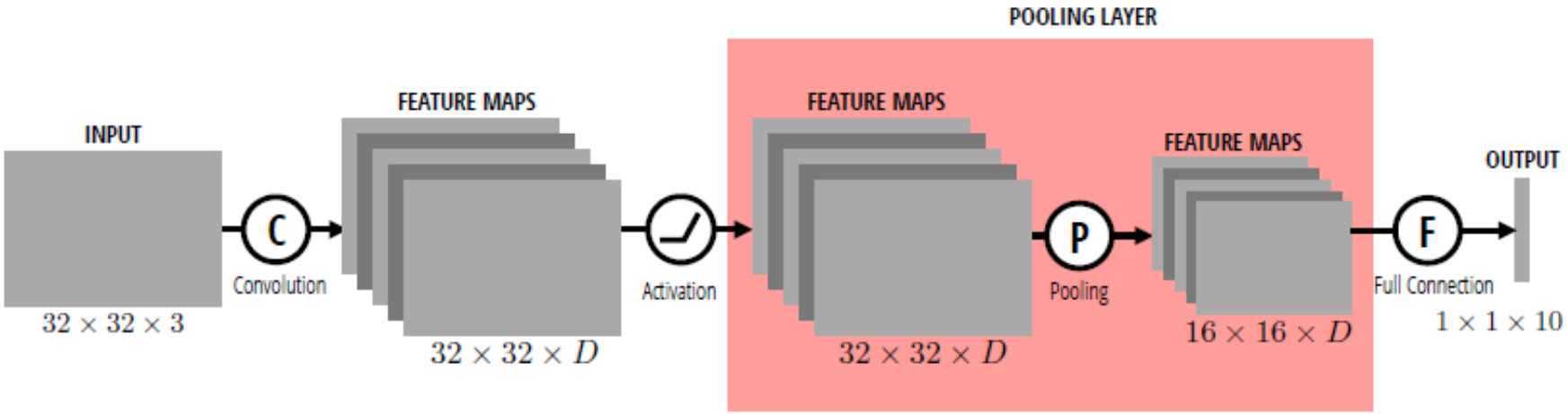


Nonlinearity Layer

- E.g., ReLU (Rectified Linear Unit)
  - Pixel by pixel computation of max(0, x)

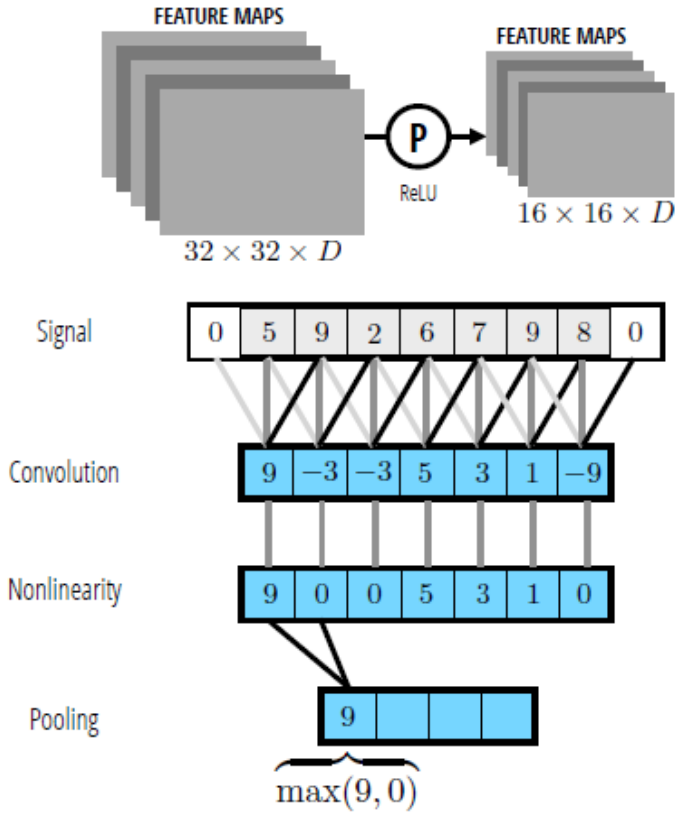


Pooling Layer in CNN



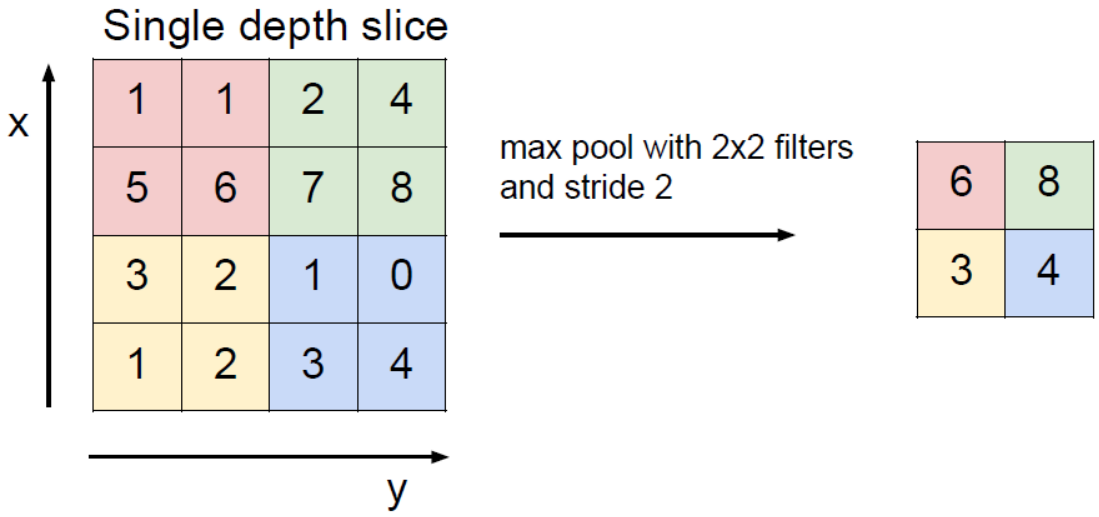
Pooling Layer

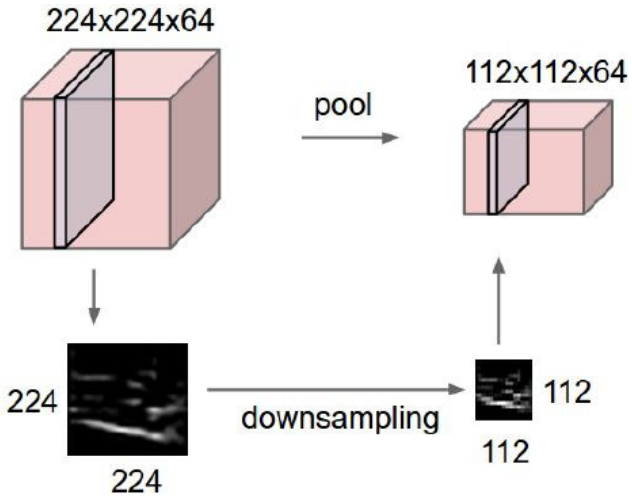
- Makes the representations smaller and more manageable
- Operates over each activation map independently
- E.g., Max Pooling



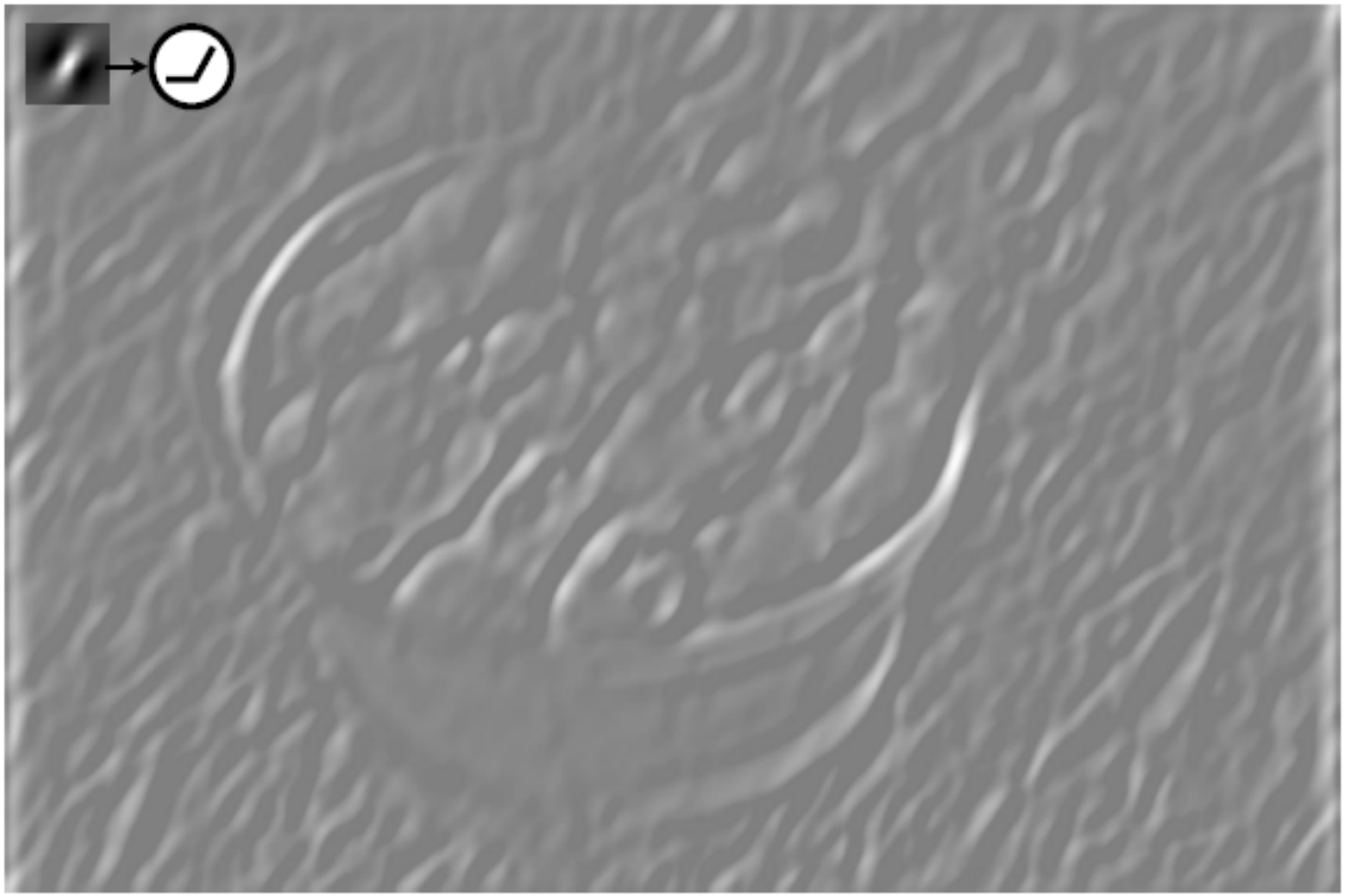
Pooling Layer

- Reduces the spatial size and provides spatial invariance

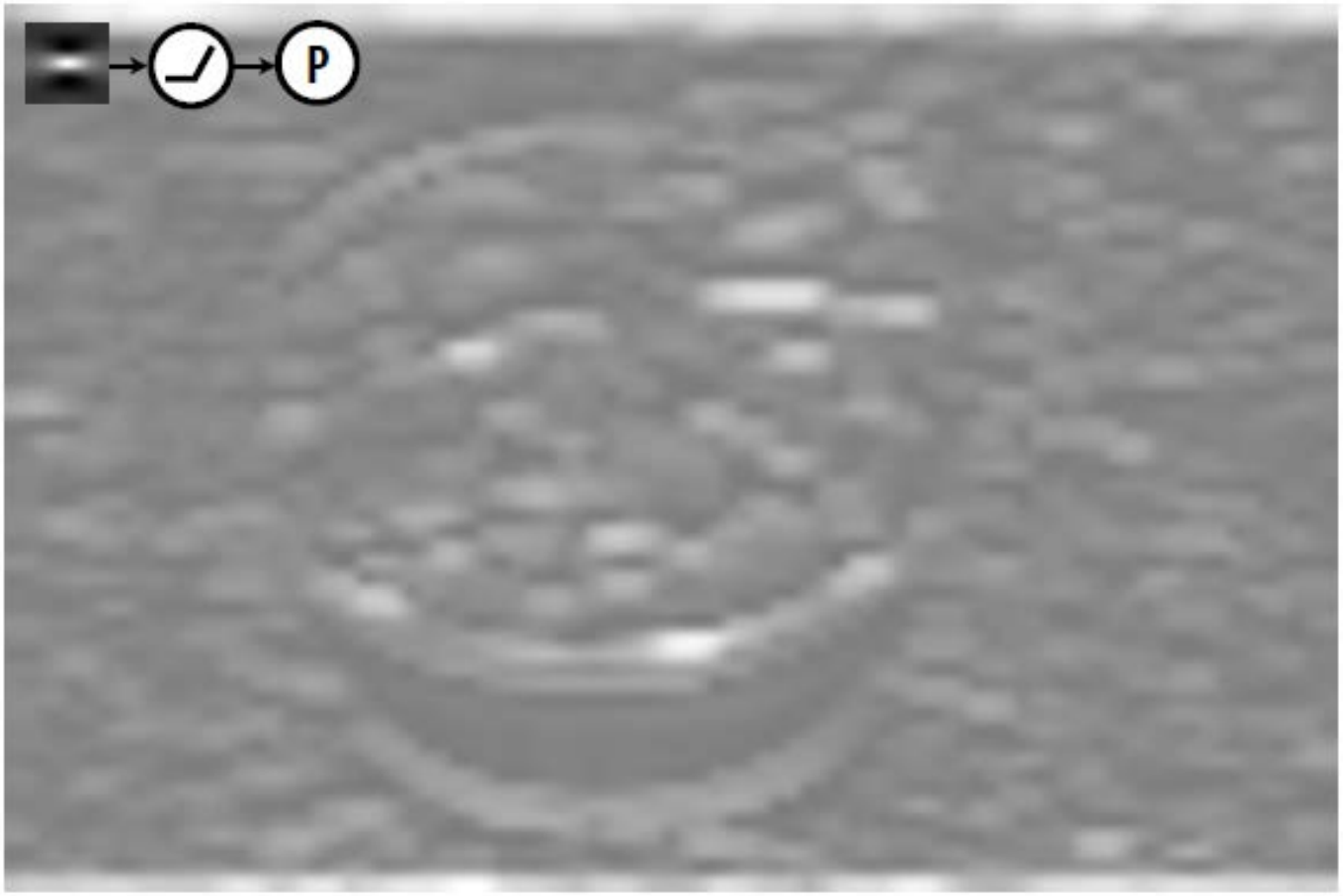




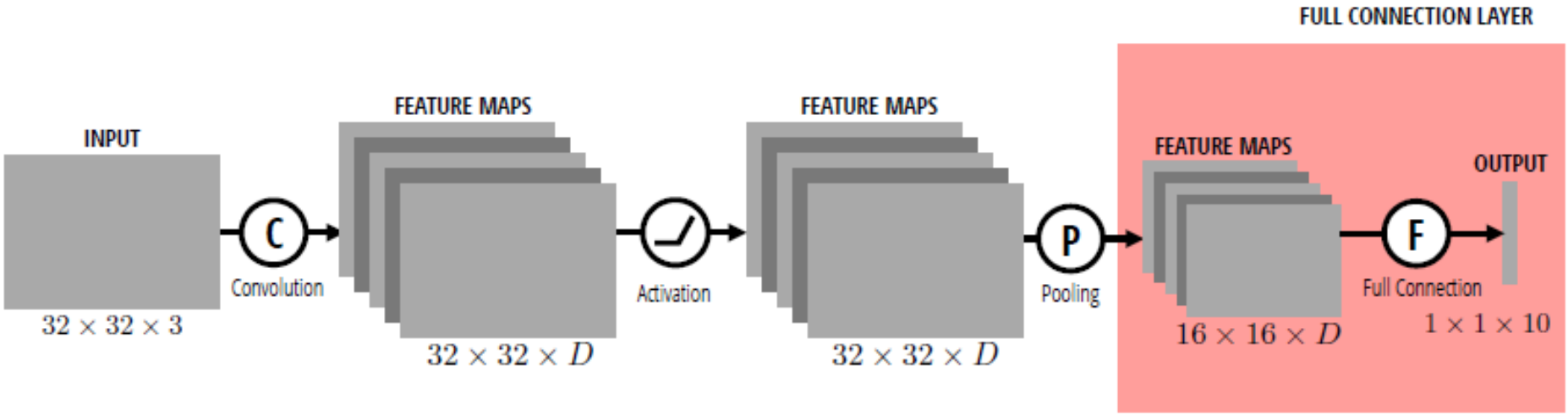
- Example
  - Nonlinearity by ReLU



- Example
  - Max pooling

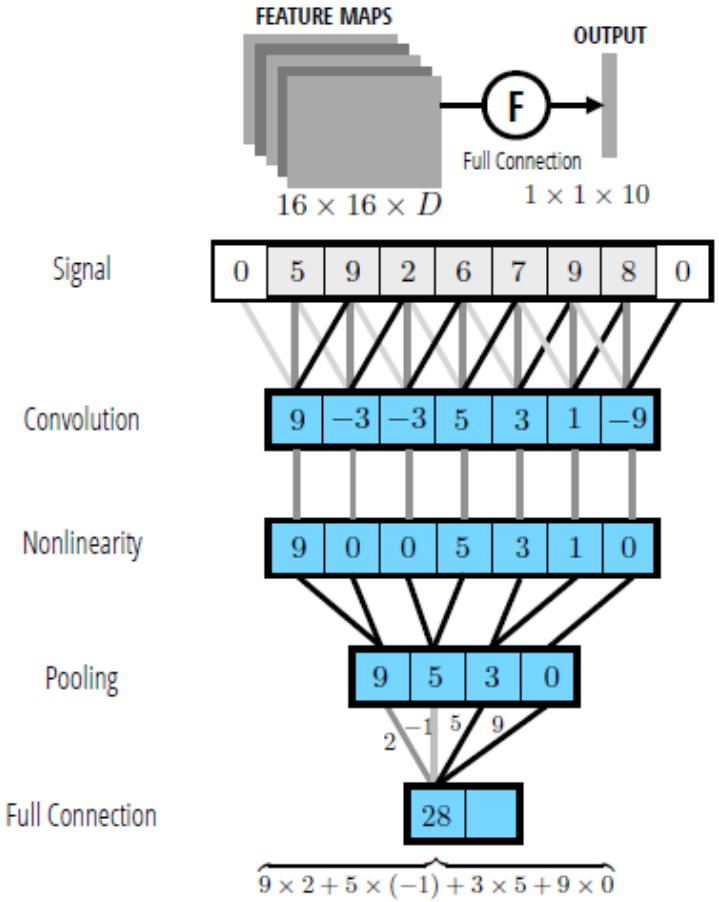


Fully Connected (FC) Layer in CNN



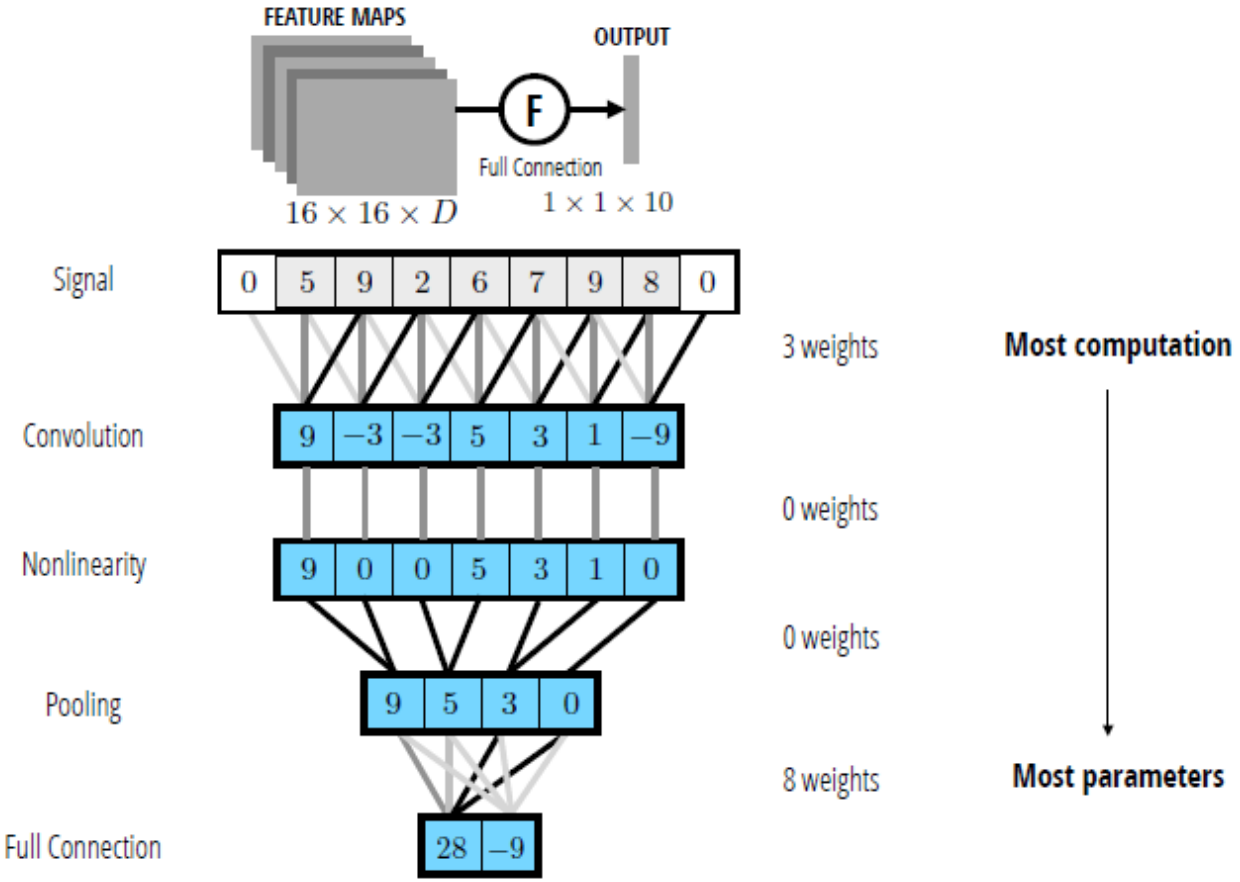
FC Layer

- Contains neurons that connect to the entire input volume, as in ordinary neural networks

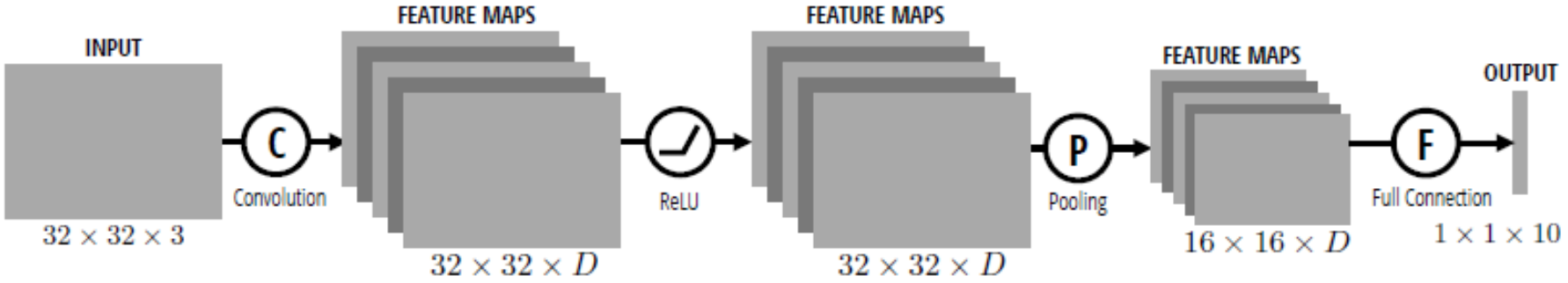


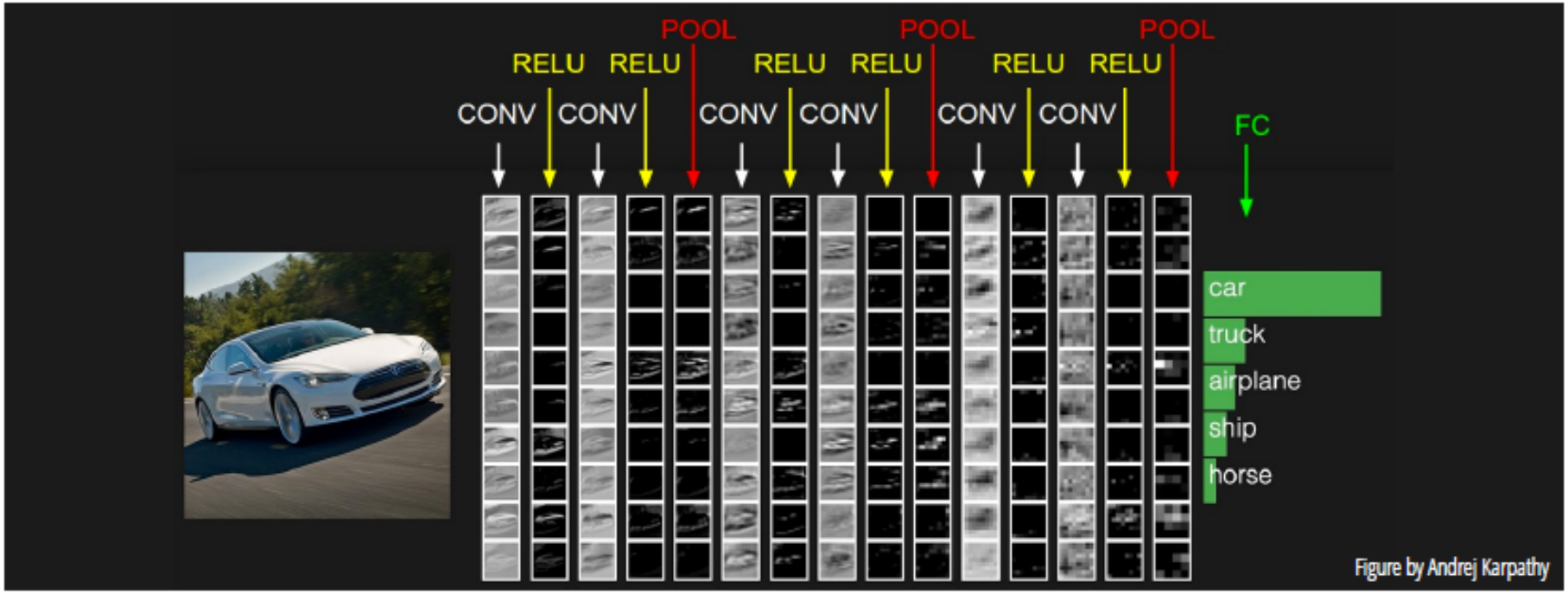
FC Layer

- Contains neurons that connect to the entire input volume, as in ordinary neural networks



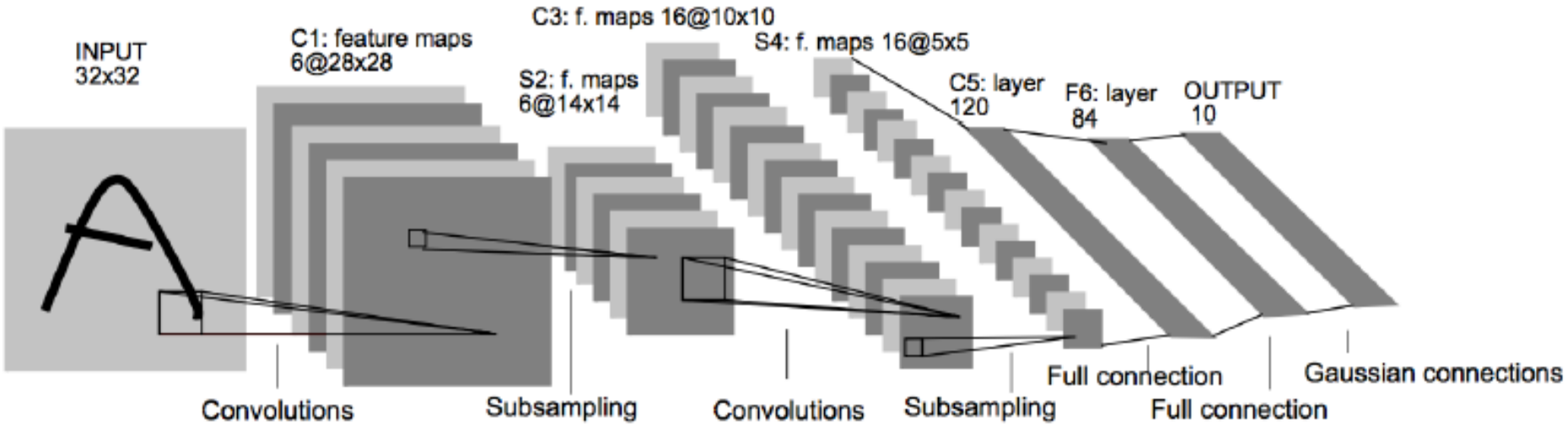
CNN


INPUT
$32 \times 32 \times 3$
C
Convolution
FEATURE MAPS
$32 \times 32 \times D$
ReLU
FEATURE MAPS
$32 \times 32 \times D$
P
Pooling
FEATURE MAPS
$16 \times 16 \times D$
F
Full Connection
OUTPUT
$1 \times 1 \times 10$


CONV
RELU
CONV
POOL
CONV
RELU
CONV
POOL
CONV
RELU
CONV
POOL
FC
car
truck
airplane
ship
horse
Figure by Andrej Karpathy

LeNet

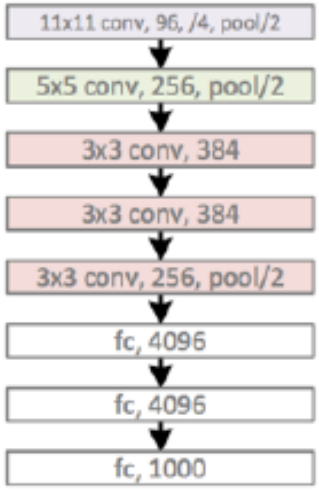
- Presented by Yann LeCun during the 1990s for reading digits
- Has the elements of modern architectures

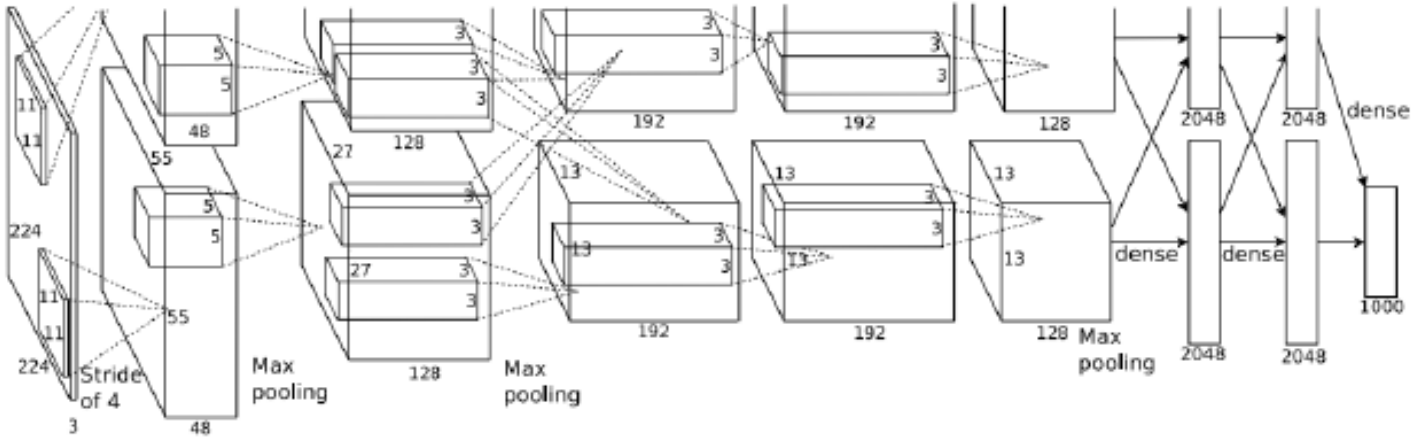


AlexNet [Krizhevsky et al., 2012]

- Repopularized CNN
  by winning the ImageNet Challenge 2012
- 7 hidden layers, 650,000 neurons,
  60M parameters
- Error rate of 16% vs. 26% for 2nd place.

Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)





Deep or Not?

- Depth of the network is critical for performance.

AlexNet: 8 Layers with 18.2% top-5 error

Removing Layer 7 reduces 16 million parameters, but only 1.1% drop in performance!

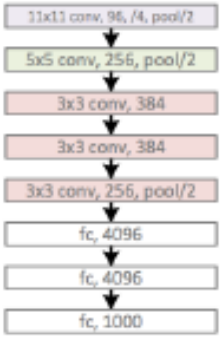
Removing Layer 6 and 7 reduces 50 million parameters, but only 5.7% drop in performance

Removing middle conv layers reduces 1 million parameters, but only 3% drop in performance

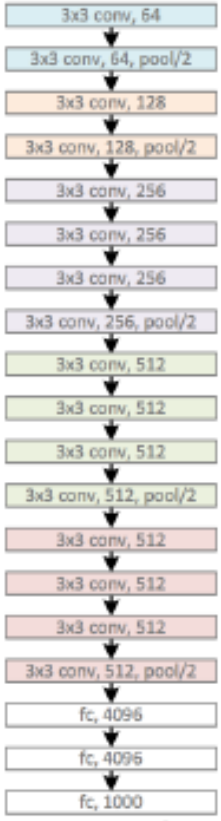
Removing feature & conv layers produces a **33% drop** in performance

CNN: A Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



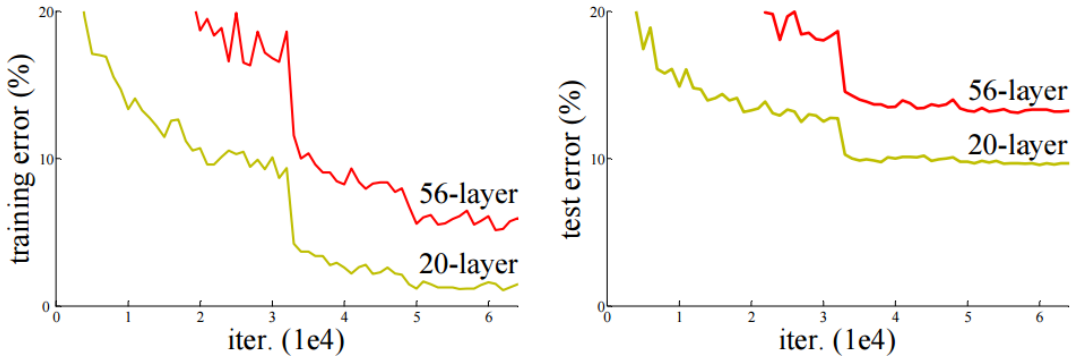
VGG, 19 layers
(ILSVRC 2014)



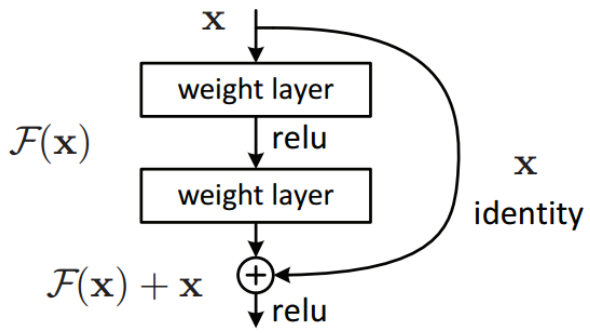
GoogleNet, 22 layers
(ILSVRC 2014)

ResNet

- Can we just increase the #layer?



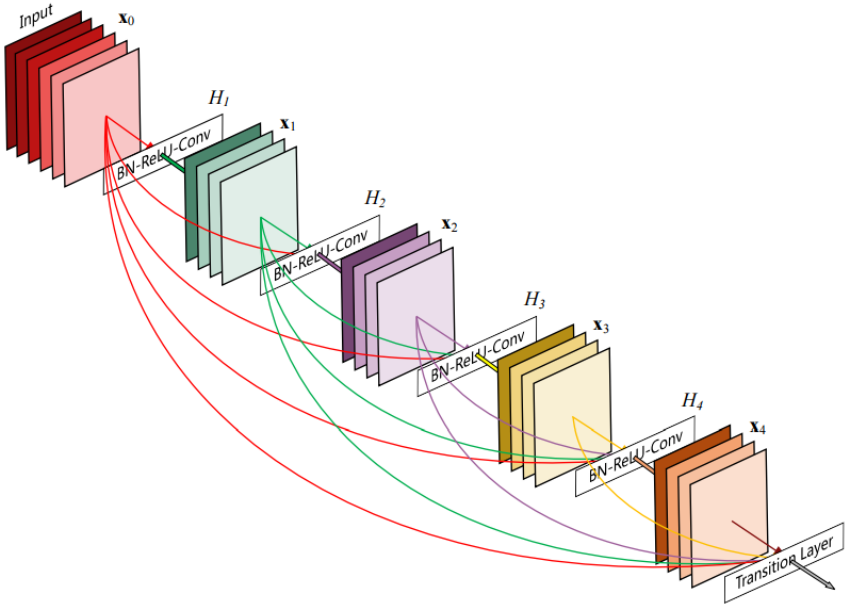
- How can we train very deep network?
  - Residual learning

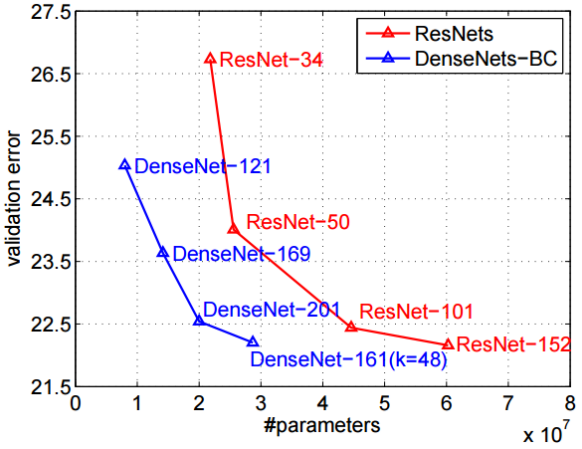


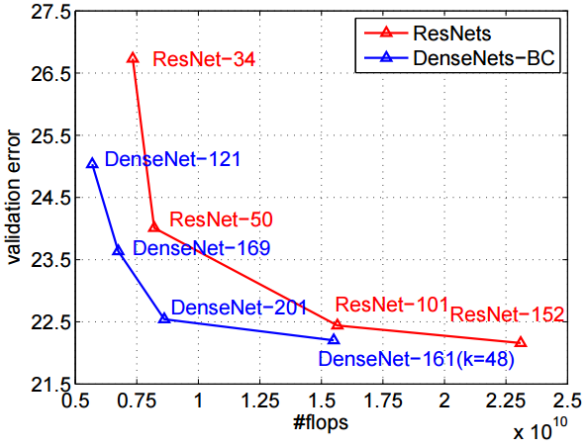
method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

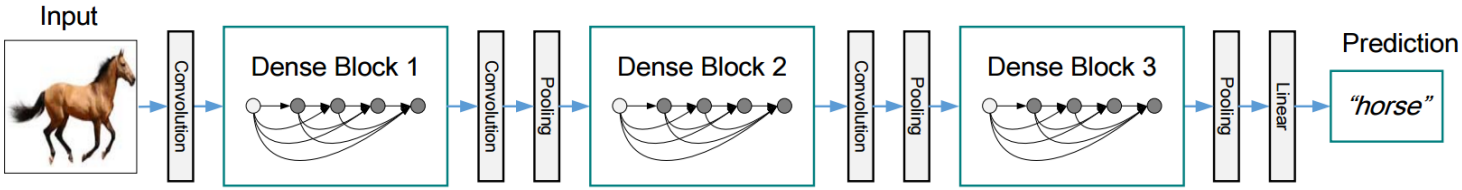
DenseNet

- Shorter connections (like ResNet) help
- Why not just connect them all?



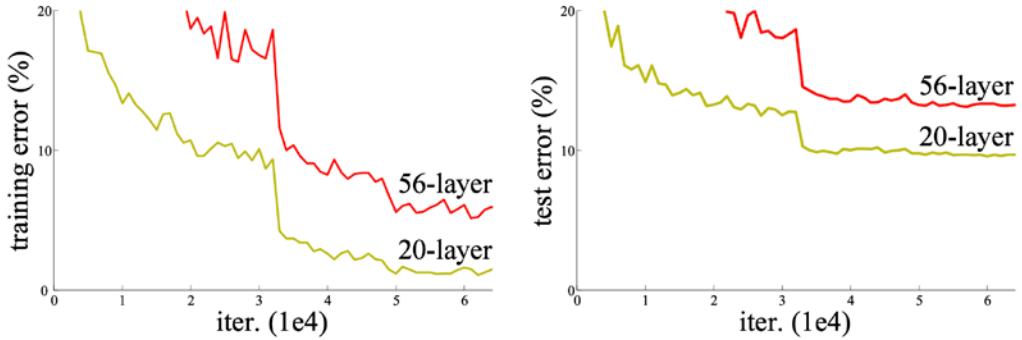




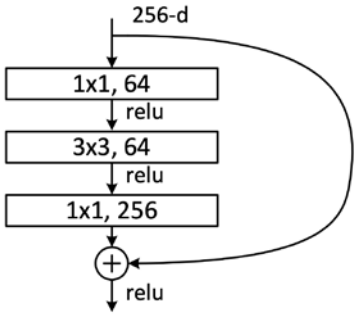


ResNet (cont'd)

- Can we just increase # of layers?



- How to train very deep networks?
  - Residual learning



Non-Bottleneck (ResNet-18, 34)

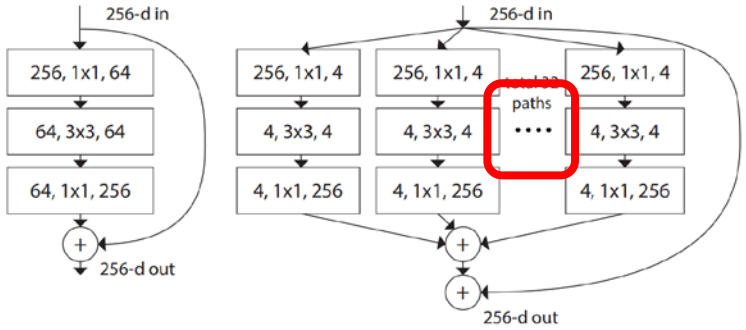
Bottleneck (ResNet-50, 101, 152)

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Ref: He, Kaiming, et al. "Deep residual learning for image recognition." *CVPR,* 2016.

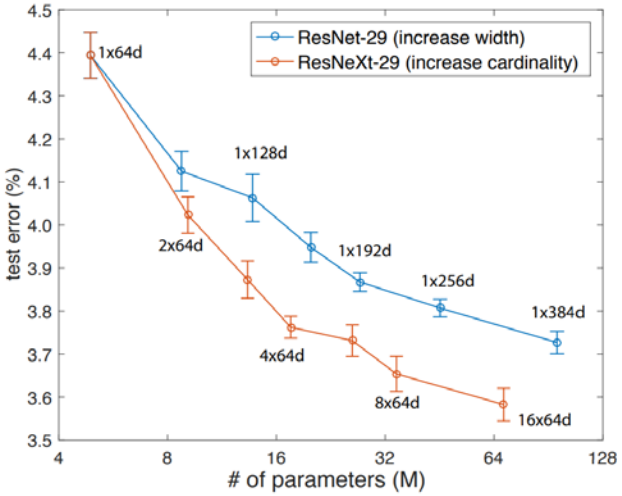
ResNeXT

- Deeper and wider → better…what else?
  - Increase cardinality



ResNet block ResNeXt block

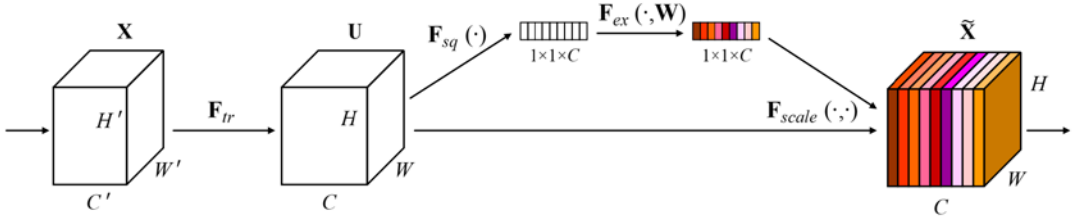
	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

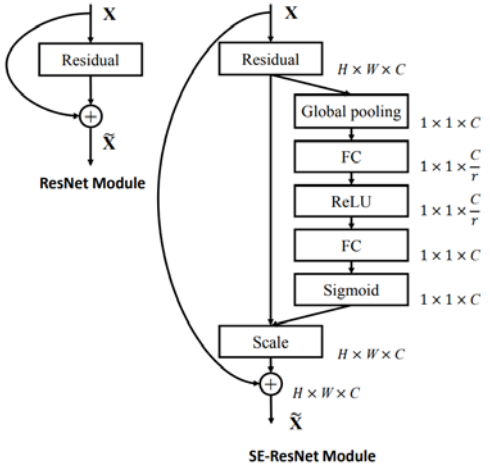


Xie, Saining, et al. "Aggregated residual transformations for deep neural networks." *CVPR,* 2017.

Squeeze-and-Excitation Net (SENet)

- How to improve acc. without much overhead?
  - Feature recalibration (channel attention)





	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [13]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [13]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [13]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [19]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [19]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16 [11]	-	-	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception [6]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [21]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." *CVPR,* 2018.

Remarks

- CNN:
  - Reduce the number of parameters
  - Reduce the memory requirements
  - Make computation independent of the size of the image
- Neuroscience provides strong inspiration on the NN design, but little guidance on how to train CNNs.
- Few structures discussed: convolution, nonlinearity, pooling

Training Convolutional Neural Networks

- Backpropagation +
  stochastic gradient descent with momentum
  - Neural Networks: Tricks of the Trade
- Dropout
- Data augmentation
- Batch normalization

What Will We Cover Next Week?

- Pytorch Framework Tutorial
  - Introduction to Pytorch
  - Basis: Simple Classification Example
  - Advance : nn Module Class
  - Advance : DataSet & DataLoader Class
  - Advance : Finetuning with pretrained model

- HW #1 is due 3/23 Sat 3AM & no late submission!!