



mp

max planck institut
informatik

SIC Saarland Informatics
Campus

High Level Computer Vision

Image Classification, Linear Classifier & Losses
@ April 19, 2023

Bernt Schiele

<https://cms.sic.saarland/hlcvss23/>

**Max Planck Institute for Informatics & Saarland University,
Saarland Informatics Campus Saarbrücken**

Overview Lecture (preliminary)

- CNNs for High-Level Computer Vision Tasks:
 - Image Classification, Object Detection (Recognition, 2D Localization), Semantic Segmentation
 - Data preprocessing & Batch normalization
 - CNN architectures
- Transformer Architectures for High-Level Vision Tasks
 - Vision Transformers for image classification, DETR for object detection, etc.
- Generative Models
 - Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAEs), (Stable) Diffusion, ...
- Self-Supervised Learning
 - How to learn with less or even no supervision
- Foundational Models
 - Vision-Language Models, Segment Anything Model (SAM), ...
- Recent Trends and Works
 - B-Cos Networks (inherently interpretable), Flamingo, ConvNeXt, ...

Overview Today's Lecture

- Image Classification
 - ▶ using data-driven approaches (machine learning)
 - ▶ K-nearest neighbor classifier
 - ▶ linear classification (parametric approach)
 - ▶ loss functions and regularization
 - ▶ softmax classifier
 - ▶ optimization via gradient descent
- What is Deep Learning
 - ▶ intuition why deep learning can help
 - ▶ integrated learning of features and classifier

Image Classification: a core task in computer vision



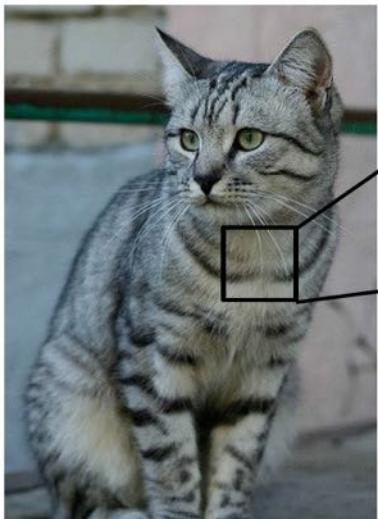
This image by Nikita is
licensed under CC-BY 2.0

(assume given set of discrete labels)
 $\{\text{dog, cat, truck, plane, ...}\}$



Image Classification

The Problem: Semantic Gap



This image by [Nikita](#) is
licensed under [CC-BY 2.0](#).

```
[[105 112 108 111 104 99 106 99 96 103 112 119 104 67 93 87]
 [ 91 98 102 106 104 79 98 103 99 105 123 136 110 105 94 85]
 [ 76 85 90 105 128 105 87 96 95 99 115 112 106 103 99 85]
 [ 99 81 81 93 128 131 127 108 95 98 102 99 96 93 101 94]
 [106 91 61 64 69 91 88 85 101 107 109 98 75 84 96 95]
 [114 108 85 55 55 69 64 54 64 87 112 129 98 74 84 91]
 [133 137 147 103 65 81 88 65 52 54 74 84 102 93 85 62]
 [128 137 144 148 109 95 86 70 62 65 63 63 60 73 86 101]
 [125 129 140 137 119 121 117 94 65 79 80 65 54 64 72 88]
 [127 125 131 147 133 127 126 131 111 56 89 75 61 64 72 84]
 [115 114 109 123 158 148 131 118 113 109 100 52 74 65 72 78]
 [ 89 93 90 97 108 147 131 118 113 114 113 109 106 65 77 88]
 [ 63 77 86 81 77 79 102 123 117 115 117 125 125 130 115 87]
 [ 62 65 82 89 78 71 88 101 124 126 119 101 107 114 131 119]
 [ 63 65 75 88 89 71 62 81 120 138 135 105 81 68 110 118]
 [ 87 65 71 87 106 95 69 45 76 130 126 107 92 94 105 112]
 [118 97 82 86 117 123 116 66 41 51 95 53 69 65 102 107]
 [164 146 112 88 62 128 124 104 76 48 45 66 68 101 102 109]
 [157 170 157 120 93 86 114 132 112 57 69 55 70 82 99 94]
 [130 128 134 161 139 108 109 118 121 134 114 87 65 53 69 86]
 [128 112 96 86 83 112 153 149 122 109 104 75 88 107 112 99]
 [123 107 96 86 88 82 86 94 117 145 148 153 102 58 78 92 107]
 [122 121 102 88 82 86 94 117 145 148 153 102 58 78 92 107]
 [122 164 148 103 71 56 78 83 93 103 119 139 102 61 69 84]]
```

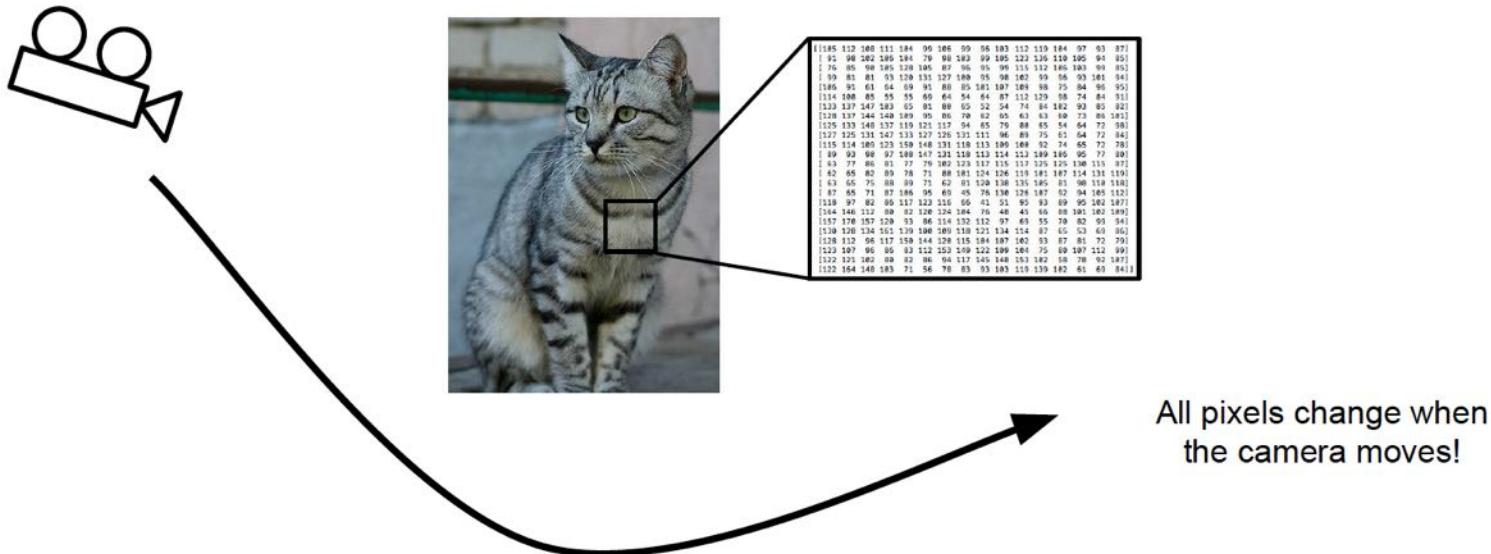
What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Image Classification

Challenges: Viewpoint variation



This image by Nikita is
licensed under CC-BY 2.0

Image Classification

Challenges: Illumination



[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)



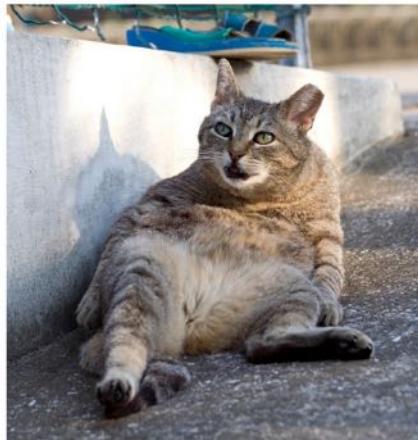
[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)

Image Classification

Challenges: Deformation



This image by [Umberto Salvagnin](#)
is licensed under [CC-BY 2.0](#)



This image by [Umberto Salvagnin](#)
is licensed under [CC-BY 2.0](#)



This image by [sare bear](#) is
licensed under [CC-BY 2.0](#)



This image by [Tom ThaIs](#)
is licensed under [CC-BY 2.0](#)

Image Classification

Challenges: Occlusion



[This image](#) is CC0 1.0 public domain



[This image](#) is CC0 1.0 public domain



[This image](#) by [jonsson](#) is licensed under CC-BY 2.0

Image Classification

Challenges: Background Clutter



[This image](#) is CC0 1.0 public domain



[This image](#) is CC0 1.0 public domain

Image Classification

Challenges: Intraclass variation



This image is CC0 1.0 public domain

Image Classification

An image classifier

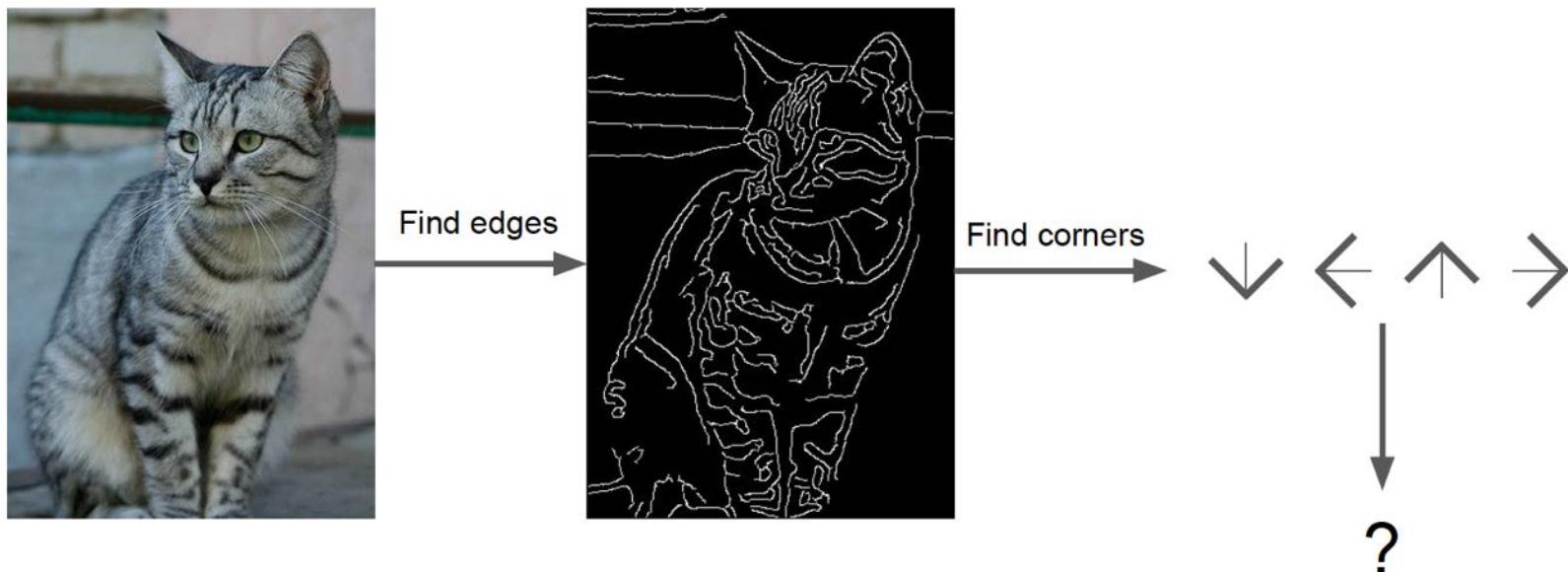
```
def classify_image(image):  
    # Some magic here?  
    return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for
recognizing a cat, or other classes.

Image Classification

Attempts have been made



John Canny, "A Computational Approach to Edge Detection", IEEE TPAMI 1986

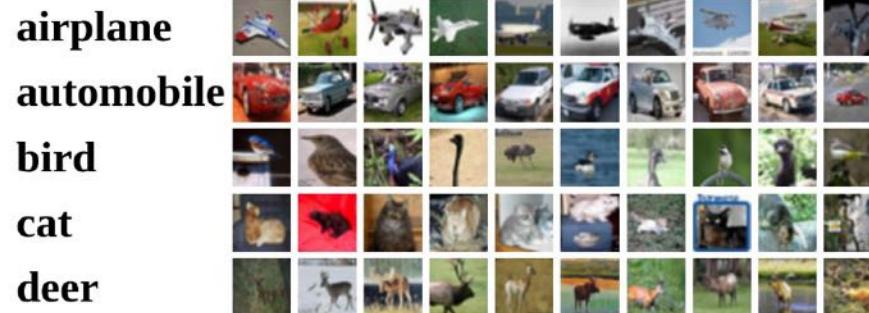
Machine Learning: Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

Example training set

```
def train(images, labels):
    # Machine learning!
    return model

def predict(model, test_images):
    # Use model to predict labels
    return test_labels
```



First Classifier: Nearest Neighbor

```
def train(images, labels):  
    # Machine learning!  
    return model
```

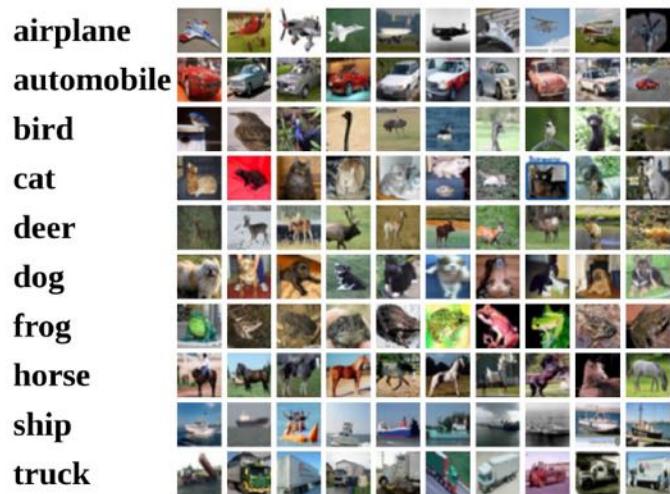
→ Memorize all
data and labels

```
def predict(model, test_images):  
    # Use model to predict labels  
    return test_labels
```

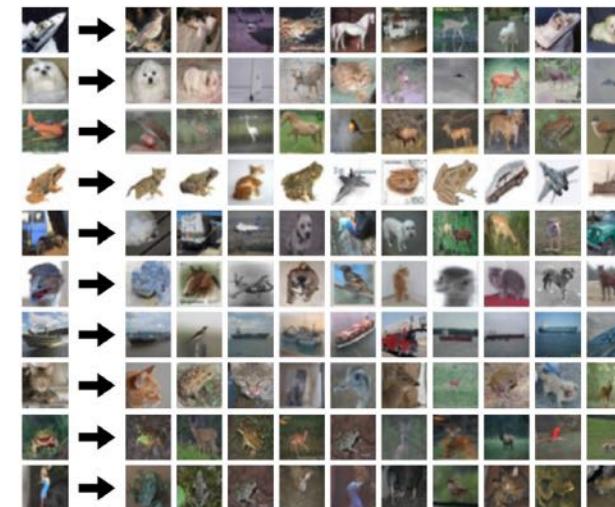
→ Predict the label
of the most similar
training image

Example Dataset: CIFAR10

10 classes
50,000 training images
10,000 testing images



Test images and nearest neighbors



Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

First Classifier: Nearest Neighbor Classifier

Distance Metric to compare images

L1 distance:

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image				training image				pixel-wise absolute value differences			
56	32	10	18	10	20	24	17	46	12	14	1
90	23	128	133	8	10	89	100	82	13	39	33
24	26	178	200	12	16	178	170	12	10	0	30
2	0	255	220	4	32	233	112	2	32	22	108

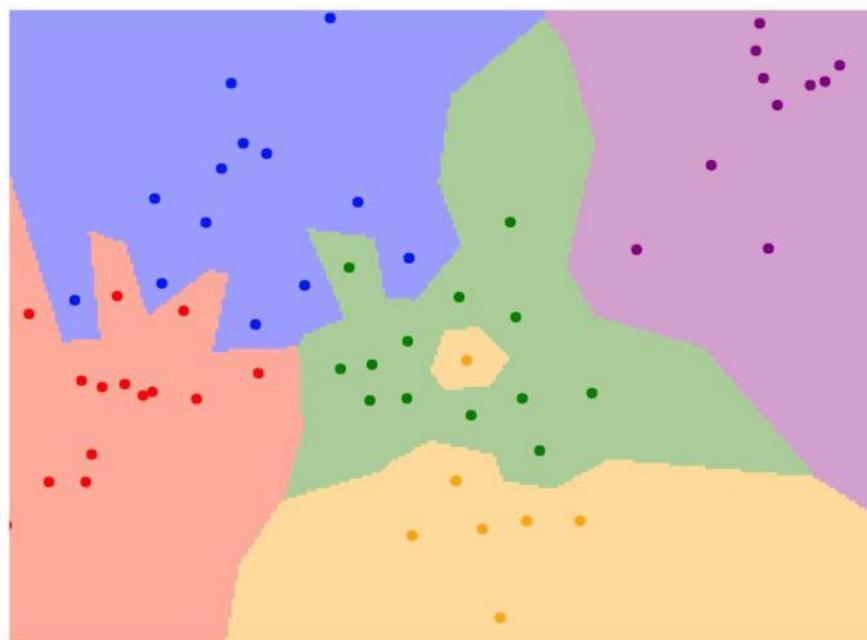
-

=

add → 456

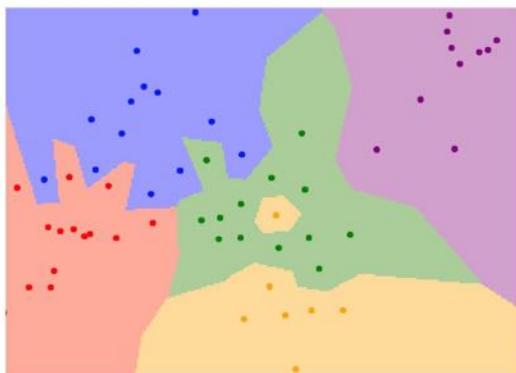
First Classifier: Nearest Neighbor Classifier

What does this look like?

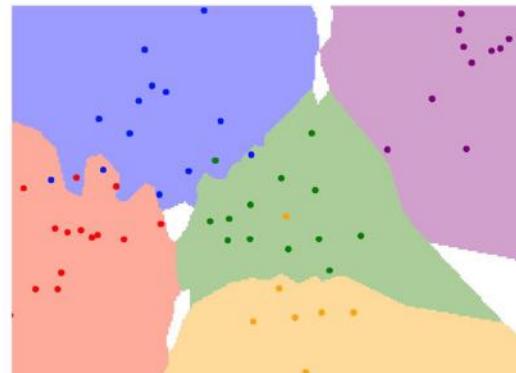


K-Nearest Neighbor Classifier

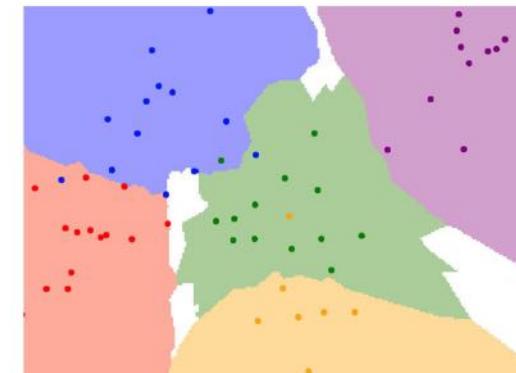
Instead of copying label from nearest neighbor,
take **majority vote** from K closest points



$K = 1$



$K = 3$



$K = 5$

K-Nearest Neighbor Classifier

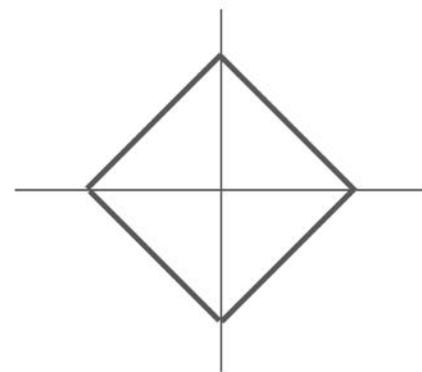
What does this look like?



K-Nearest Neighbor Classifier: Distance Metric

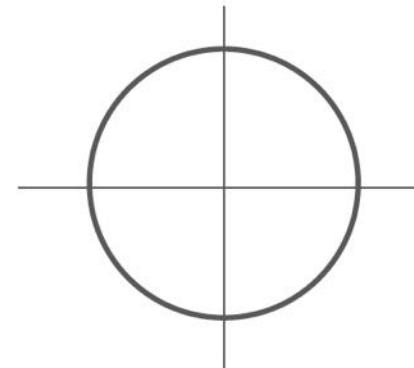
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



K-Nearest Neighbor Classifier: Distance Metric

L1 (Manhattan) distance

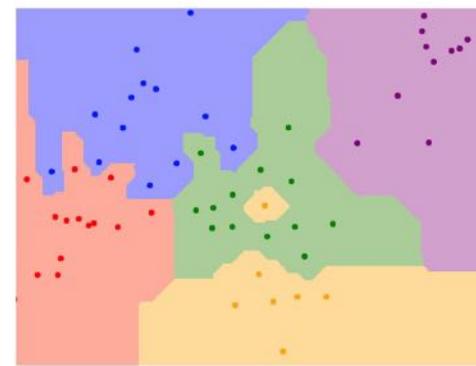
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



$K = 1$

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



$K = 1$

Hyperparameters

What is the best value of **k** to use?

What is the best **distance** to use?

These are **hyperparameters**: choices about the algorithm that we set rather than learn

Very problem-dependent.

Must try them all out and see what works best.

Setting Hyperparameters

Idea #1: Choose hyperparameters
that work best on the data

BAD: K = 1 always works
perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose
hyperparameters that work best on test data

BAD: No idea how algorithm
will perform on new data

train

test

Idea #3: Split data into **train**, **val**, and **test**; choose
hyperparameters on val and evaluate on test

Better!

train

validation

test

Setting Hyperparameters

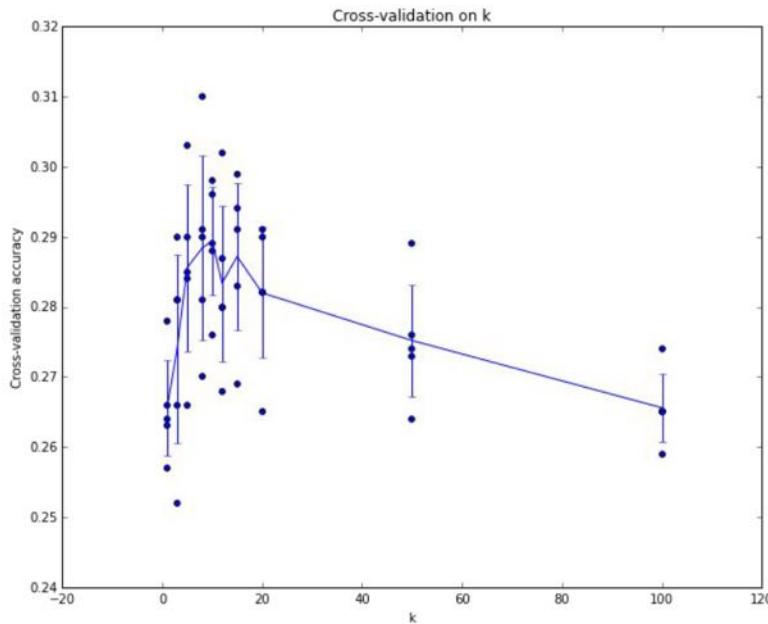
Your Dataset

Idea #4: Cross-Validation: Split data into **folds**,
try each fold as validation and average the results

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Useful for small datasets, but not used too frequently in deep learning

Setting Hyperparameters



Example of
5-fold cross-validation
for the value of **k**.

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that $k \approx 7$ works best
for this data)

Nearest Neighbor - not used for images :-)

- Very slow at test time
- Distance metrics on pixels are not informative



[Original image is
CC0 public domain](#)

(all 3 images have same L2 distance to the one on the left)

K-Nearest Neighbors: Summary

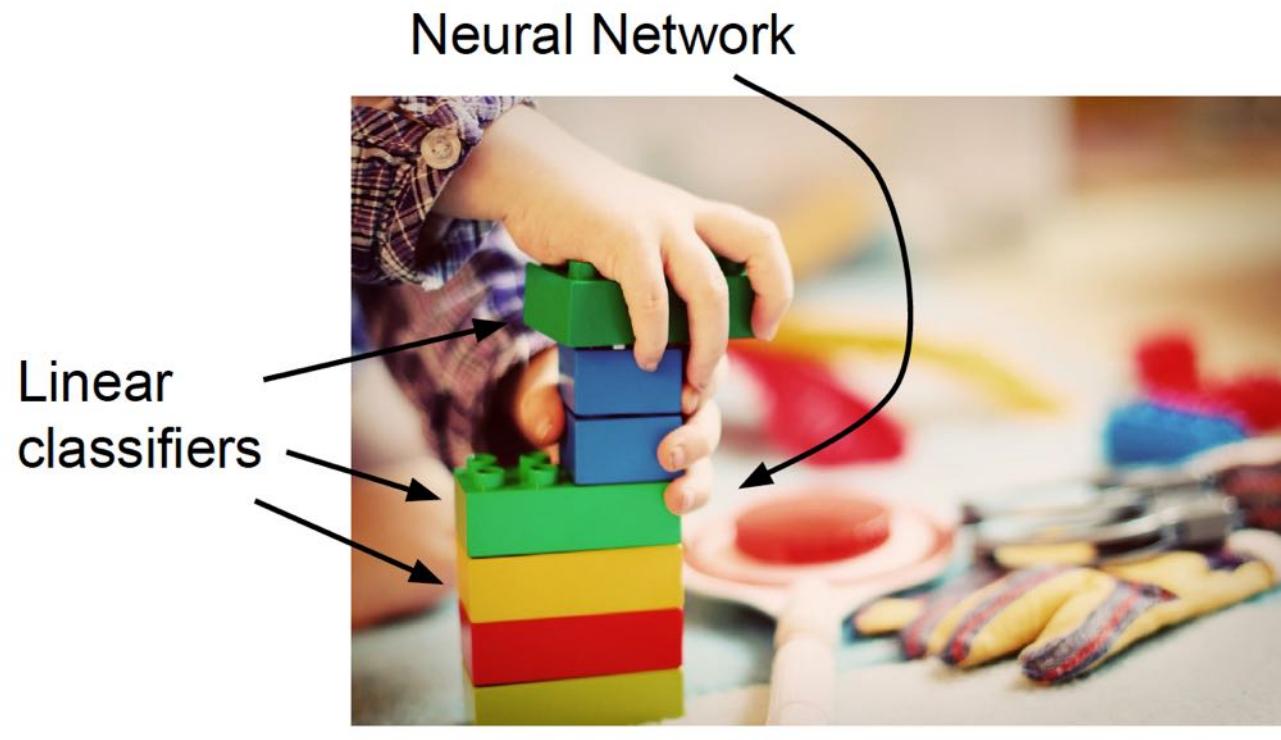
In **Image classification** we start with a **training set** of images and labels, and must predict labels on the **test set**

The **K-Nearest Neighbors** classifier predicts labels based on nearest training examples

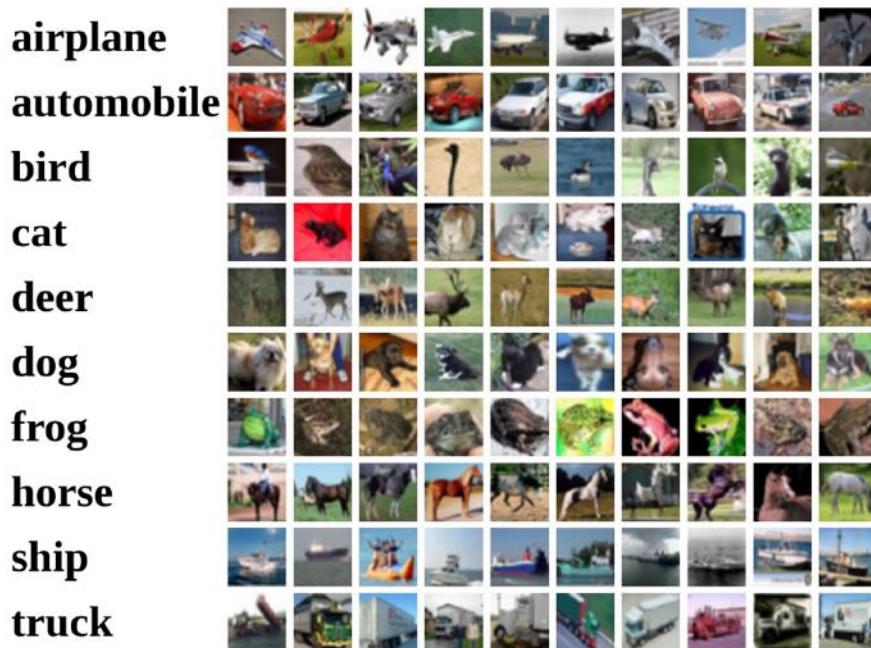
Distance metric and K are **hyperparameters**

Choose hyperparameters using the **validation set**; only run on the test set once at the very end!

Linear Classification



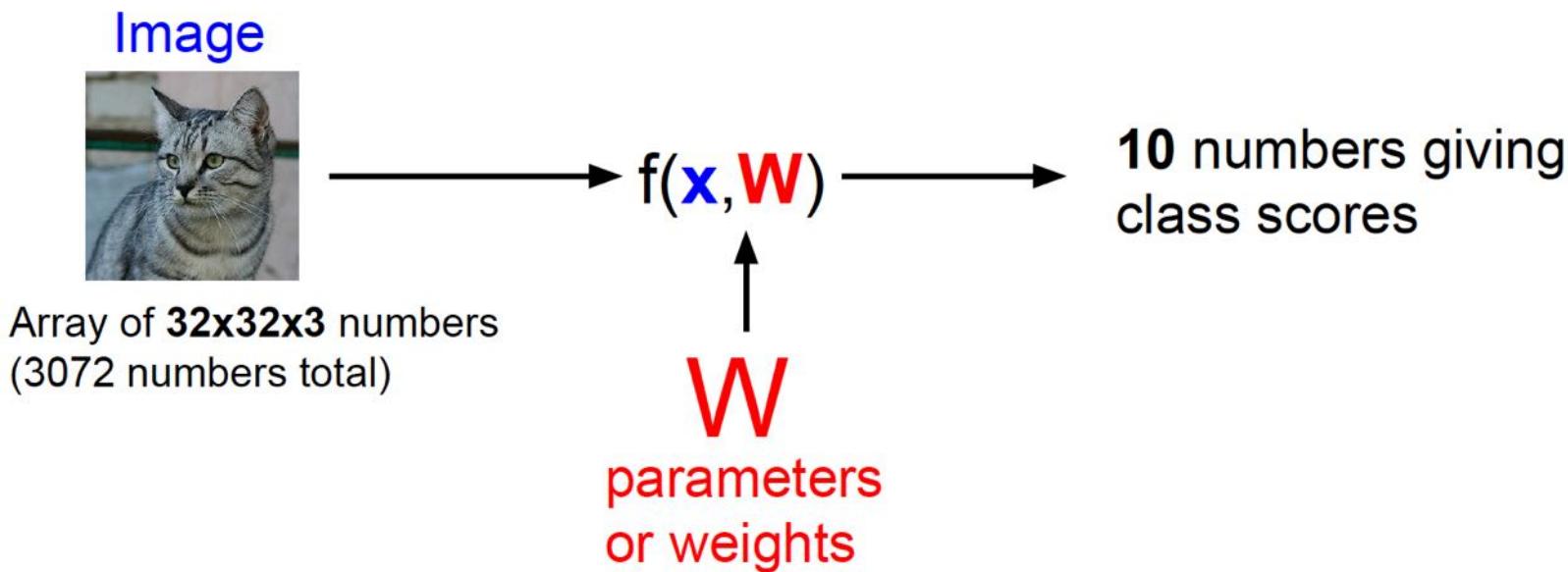
Recall CIFAR10



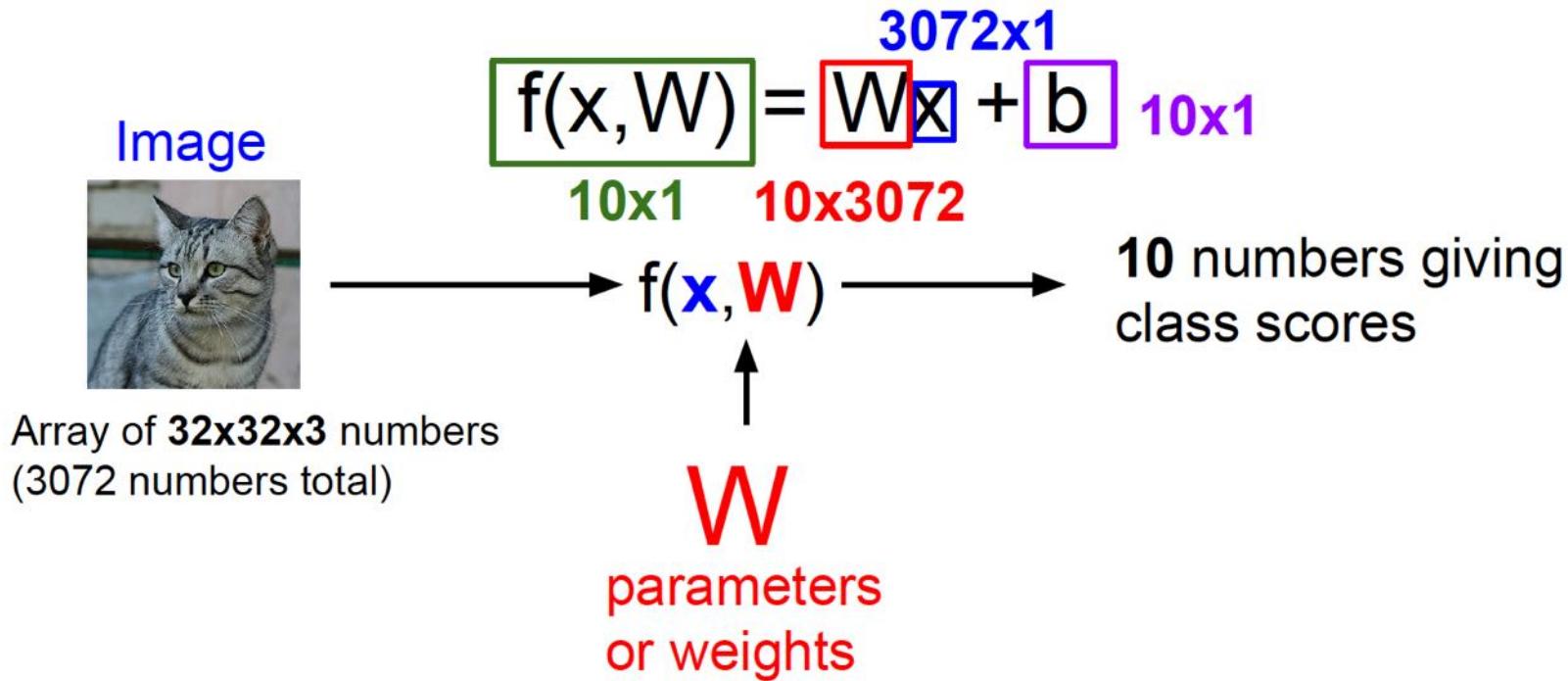
50,000 training images
each image is **32x32x3**

10,000 test images.

Parametric Approach

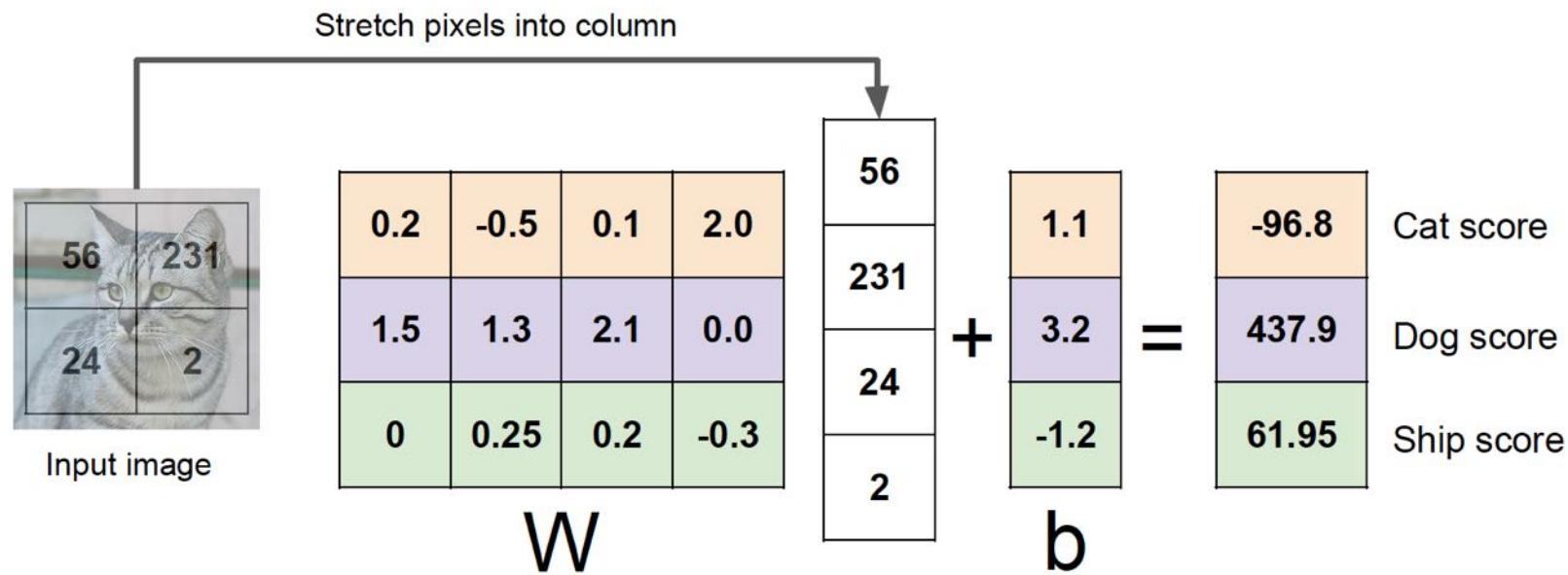


Parametric Approach: Linear Classifier



Parametric Approach: Linear Classifier

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

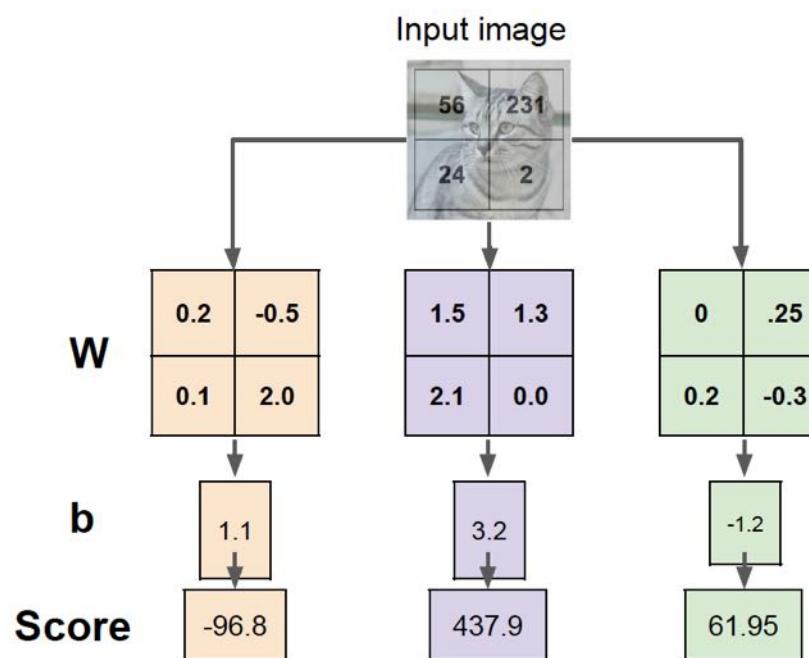
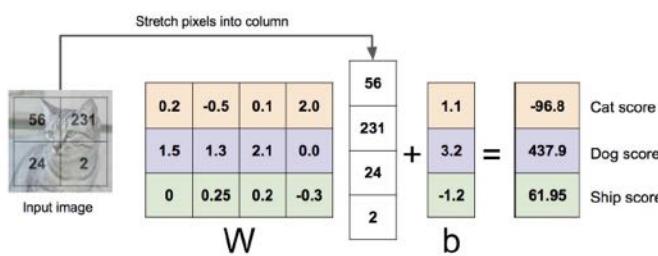


Parametric Approach: Linear Classifier

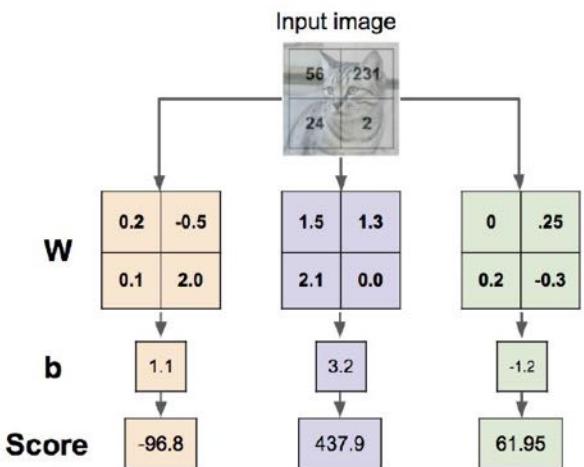
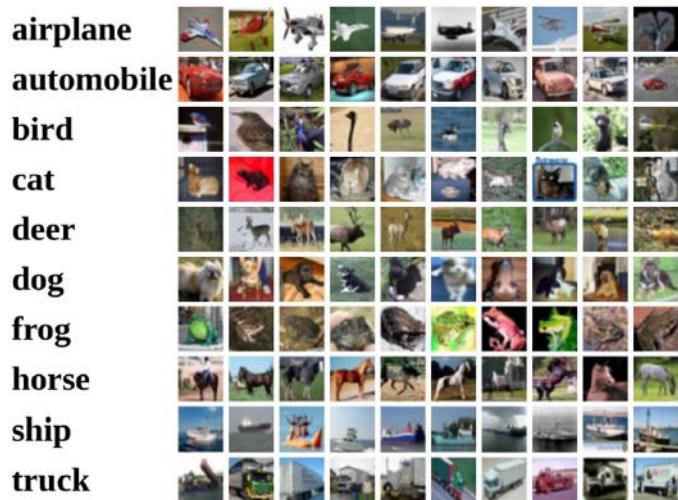
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Algebraic Viewpoint

$$f(x, W) = Wx$$



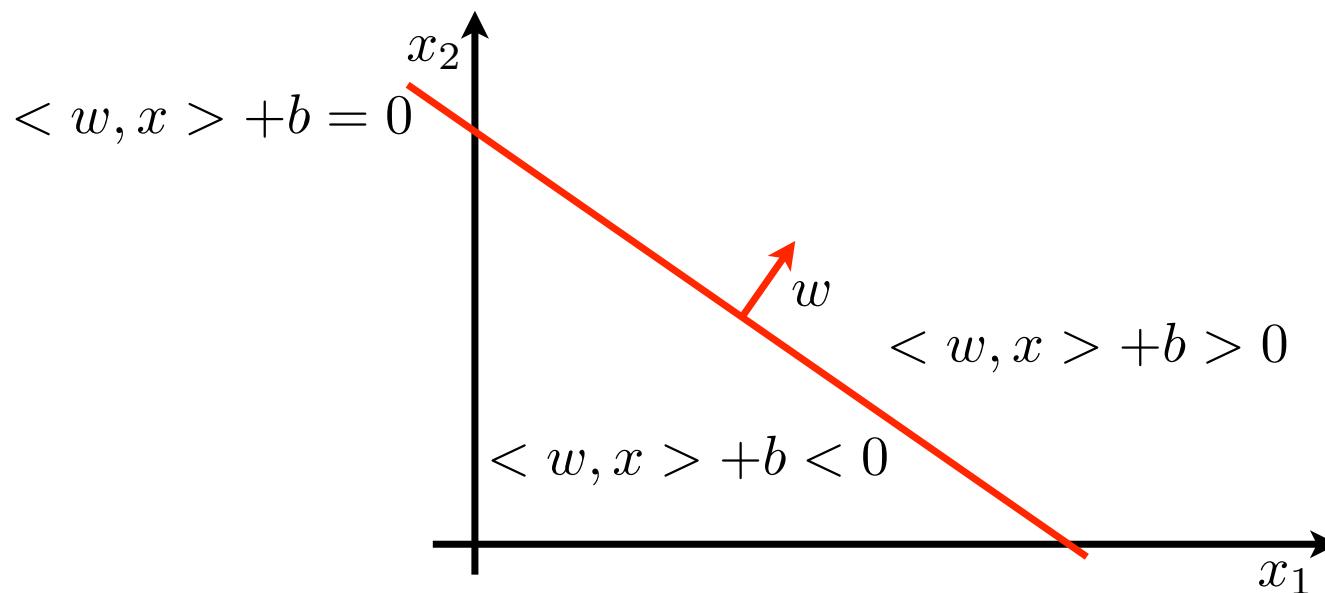
Interpreting a Linear Classifier: Visual Viewpoint



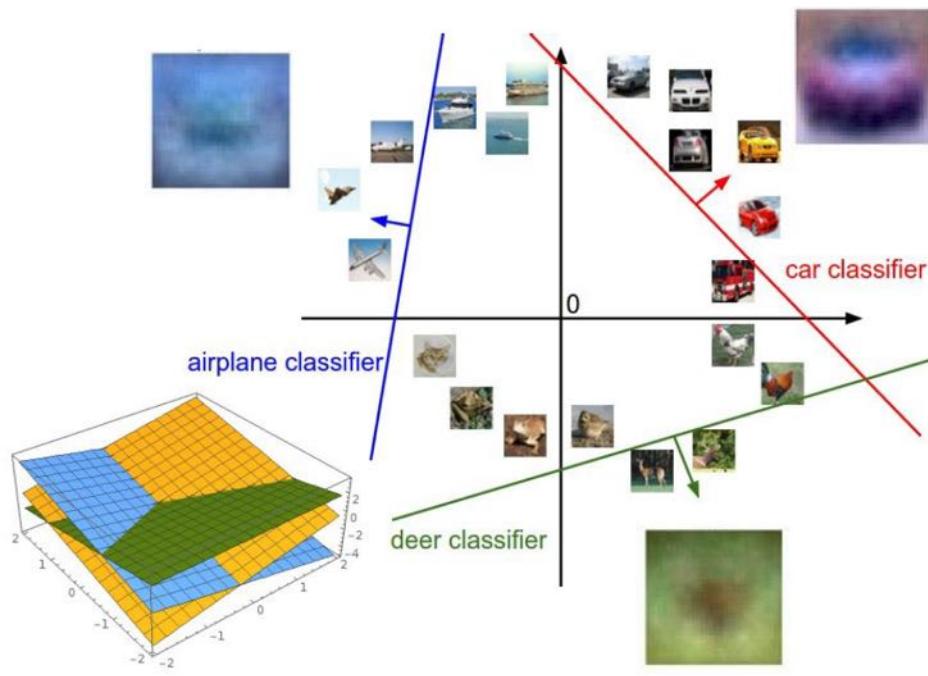
Interpreting a Linear Classifier: Geometric Viewpoint

- 2-class linear classifier:

$$f(x) = \text{sign}(\langle w, x \rangle + b) = \begin{cases} 1 & \text{if } \langle w, x \rangle + b > 0 \\ -1 & \text{if } \langle w, x \rangle + b \leq 0 \end{cases}$$



Interpreting a Linear Classifier: Geometric Viewpoint



Plot created using [Wolfram Cloud](#)

$$f(x, W) = Wx + b$$



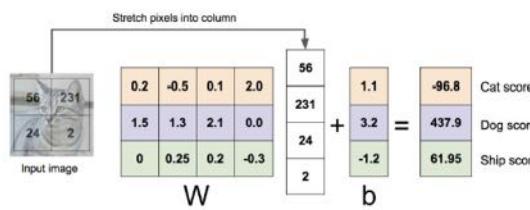
Array of **32x32x3** numbers
(3072 numbers total)

Cat image by [Nikita](#) is licensed under [CC-BY 2.0](#)

Linear Classifier: Three Viewpoints

Algebraic Viewpoint

$$f(x, W) = Wx$$



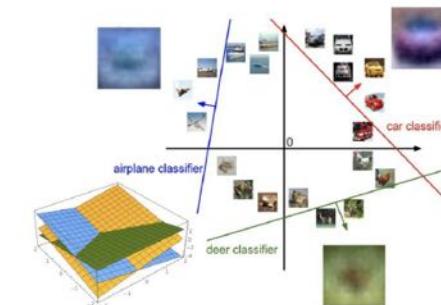
Visual Viewpoint

One template per class



Geometric Viewpoint

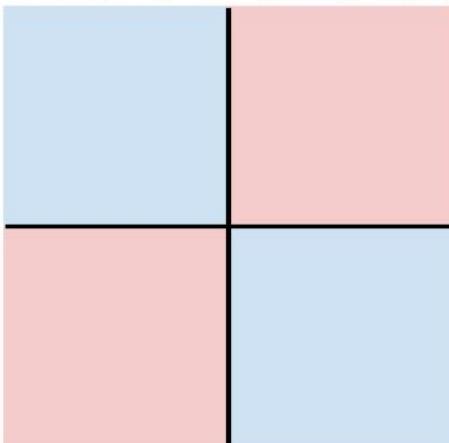
Hyperplanes cutting up space



Hard Cases for a Linear Classifier

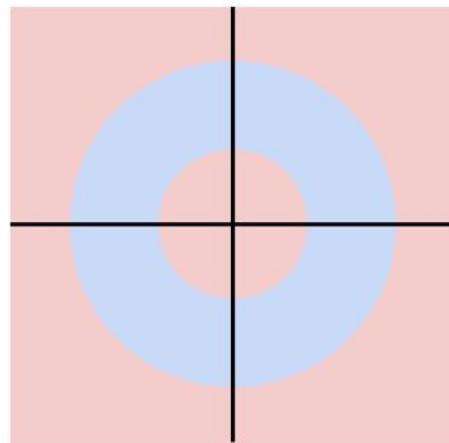
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



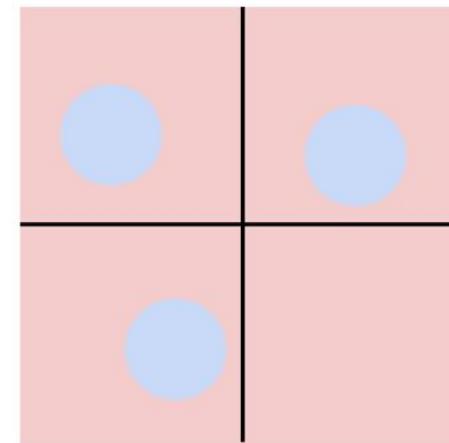
Class 1:
 $1 \leq L_2 \text{ norm} \leq 2$

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



Linear Classifier...

So far: Defined a (linear) score function $f(x,W) = Wx + b$

Example class scores for 3 images for some W :

How can we tell whether this W is good or bad?

Cat image by Nikita is licensed under CC-BY 2.0
Car image is CC0 1.0 public domain
Frog image is in the public domain



airplane	-3.45	-0.51	3.42
automobile	-8.87	6.04	4.64
bird	0.09	5.31	2.65
cat	2.9	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	-4.34
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

Linear Classifier...



airplane	-3.45	-0.51	3.42
automobile	-8.87	6.04	4.64
bird	0.09	5.31	2.65
cat	2.9	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	-4.34
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

Cat image by Nikita is licensed under CC-BY 2.0; Car image is CC0 1.0 public domain; Frog image is in the public domain

TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.
2. Come up with a way of efficiently finding the parameters that minimize the loss function. **(optimization)**

Example

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

A **loss function** tells how good our current classifier is

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^N$$

Where x_i is image and
 y_i is (integer) label

Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

Multiclass Support Vector Machine (SVM) Loss

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$\begin{aligned} L_i &= \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases} \\ &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \end{aligned}$$

Multiclass Support Vector Machine (SVM) Loss

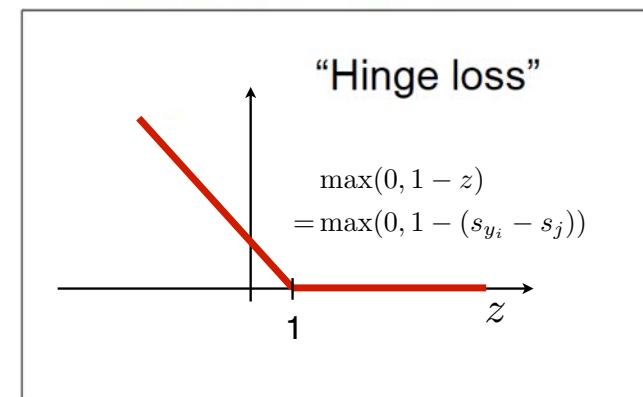
Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

Multiclass SVM loss:



$$\begin{aligned}L_i &= \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases} \\&= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)\end{aligned}$$

Multiclass Support Vector Machine (SVM) Loss

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9		

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$\begin{aligned} L_i &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \\ &= \max(0, 5.1 - 3.2 + 1) \\ &\quad + \max(0, -1.7 - 3.2 + 1) \\ &= \max(0, 2.9) + \max(0, -3.9) \\ &= 2.9 + 0 \\ &= 2.9 \end{aligned}$$

Multiclass Support Vector Machine (SVM) Loss

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9		

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$\begin{aligned} L_i &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \\ &= \max(0, 1.3 - 4.9 + 1) \\ &\quad + \max(0, 2.0 - 4.9 + 1) \\ &= \max(0, -2.6) + \max(0, -1.9) \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

Multiclass Support Vector Machine (SVM) Loss

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$\begin{aligned} L_i &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \\ &= \max(0, 2.2 - (-3.1) + 1) \\ &\quad + \max(0, 2.5 - (-3.1) + 1) \\ &= \max(0, 6.3) + \max(0, 6.6) \\ &= 6.3 + 6.6 \\ &= 12.9 \end{aligned}$$

Multiclass Support Vector Machine (SVM) Loss

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Loss over full dataset is average:

$$\begin{aligned} L &= \frac{1}{N} \sum_{i=1}^N L_i \\ L &= (2.9 + 0 + 12.9)/3 \\ &= 5.27 \end{aligned}$$

Multiclass Support Vector Machine (SVM) Loss

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q: What happens to loss if car scores change a bit?

Multiclass Support Vector Machine (SVM) Loss

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q2: what is the min/max possible loss?

Multiclass Support Vector Machine (SVM) Loss

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q3: At initialization W is small so all $s \approx 0$. What is the loss?

Multiclass Support Vector Machine (SVM) Loss

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q4: What if the sum was over all classes?
(including $j = y_i$)

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

E.g. Suppose that we found a W such that $L = 0$.
Is this W unique?

No! $2W$ is also has $L = 0!$

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9		
	0		

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Before:

$$\begin{aligned} &= \max(0, 1.3 - 4.9 + 1) \\ &\quad + \max(0, 2.0 - 4.9 + 1) \\ &= \max(0, -2.6) + \max(0, -1.9) \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

With W twice as large:

$$\begin{aligned} &= \max(0, 2.6 - 9.8 + 1) \\ &\quad + \max(0, 4.0 - 9.8 + 1) \\ &= \max(0, -6.2) + \max(0, -4.8) \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

E.g. Suppose that we found a W such that $L = 0$.
Is this W unique?

No! $2W$ is also has $L = 0$!
How do we choose between W and $2W$?

Regularization

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \lambda R(W)$$

λ = regularization strength (hyperparameter)

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too well* on training data

Simple examples

L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

More complex:

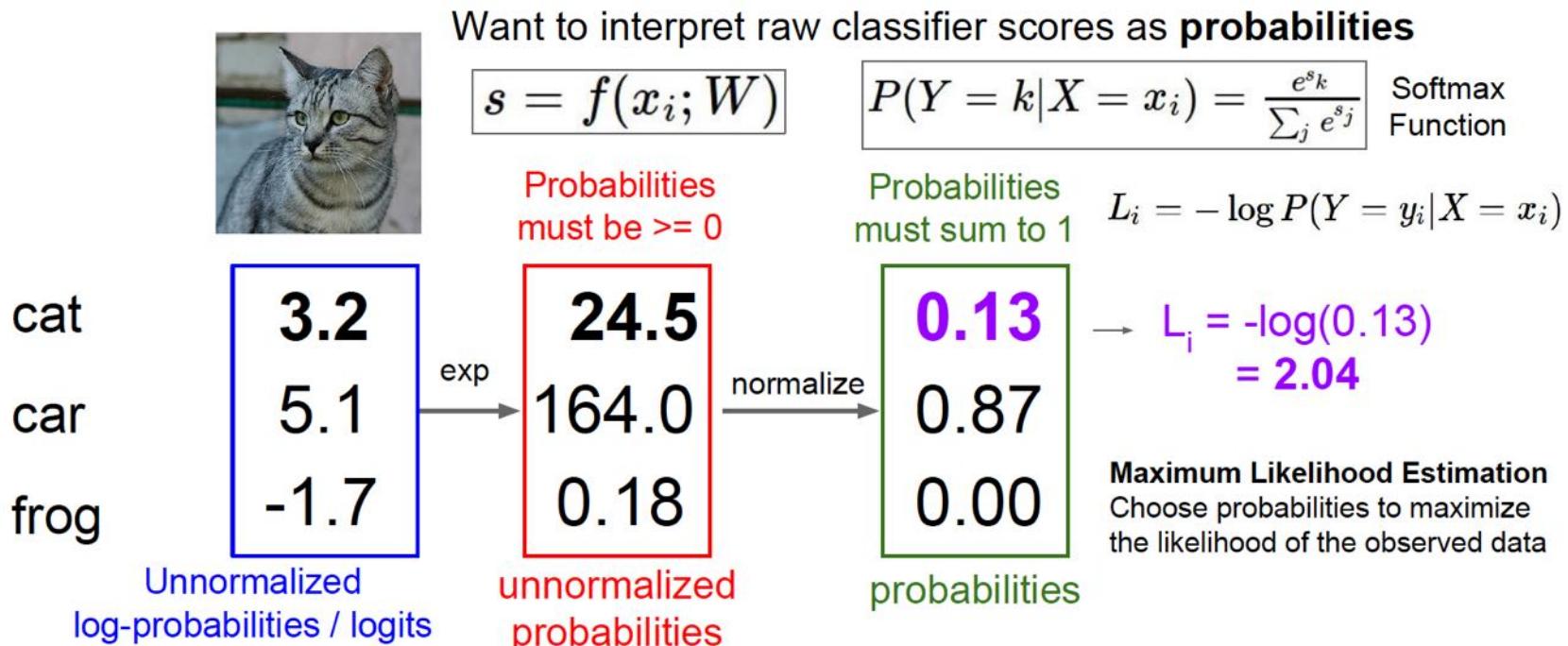
Dropout

Batch normalization

Stochastic depth, fractional pooling, etc

Softmax Classifier

Softmax Classifier (Multinomial Logistic Regression)



Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$
 Softmax Function

cat	3.2
car	5.1
frog	-1.7

Unnormalized log-probabilities / logits

Probabilities must be ≥ 0

24.5
164.0
0.18

unnormalized probabilities

exp

Probabilities must sum to 1

0.13
0.87
0.00

probabilities

normalize

$$L_i = -\log P(Y = y_i|X = x_i)$$

Kullback–Leibler divergence

$$D_{KL}(P||Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$$

1.00
0.00
0.00

Correct probs

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$
 Softmax Function

cat	3.2
car	5.1
frog	-1.7

Unnormalized log-probabilities / logits

Probabilities must be ≥ 0

24.5
164.0
0.18

unnormalized probabilities

exp

Probabilities must sum to 1

0.13
0.87
0.00

probabilities

$L_i = -\log P(Y = y_i|X = x_i)$

compare

1.00

0.00

0.00

Cross Entropy

$H(P, Q) = H(p) + D_{KL}(P||Q)$

Correct probs

0.13
0.87
0.00

probabilities

$L_i = -\log P(Y = y_i|X = x_i)$

compare

1.00

0.00

0.00

Cross Entropy

$H(P, Q) = H(p) + D_{KL}(P||Q)$

Correct probs

Let's double check...

- Entropy & KL-divergence:

$$H(P^t) = - \sum_y P^t(y) \log P^t(y)$$

$$D_{KL}(P^t || Q) = \sum_y P^t(y) \log \frac{P^t(y)}{Q(y)}$$

- Cross Entropy the sum of both:

$$\begin{aligned} H(P^t, Q) &= H(P^t) + D_{KL}(P^t || Q) \\ &= \sum_y P^t(y) \left(\log \frac{P^t(y)}{Q(y)} - \log P^t(y) \right) \\ &= - \sum_y P^t(y) \log Q(y) \end{aligned}$$

Let's double check...

- Cross Entropy in our classification case:

- ▶ Target "distribution" / output:

$$P^t(y) = \begin{cases} 1 & y = y_i \\ 0 & y \neq y_i \end{cases}$$

- ▶ Output of the network:

$$Q(y|x_i) = P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

- Then Cross Entropy Loss for image x_i

$$\begin{aligned} L_i = L(x_i) &= - \sum_y P^t(y) \log Q(y|x_i) \\ &= -1 \cdot \log Q(y_i|x_i) \\ &= -\log P(Y = y_i|X = x_i) = -\log \left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right) \end{aligned}$$

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W) \quad P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
Function

Maximize probability of correct class

Putting it all together:

$$L_i = -\log P(Y = y_i|X = x_i) \quad L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

cat	3.2
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W) \quad P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$
 Softmax Function

cat	3.2
car	5.1
frog	-1.7

Maximize probability of correct class

$$L_i = -\log P(Y = y_i|X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Q: What is the min/max possible loss L_i ?

A: min 0, max infinity

Softmax Classifier (Multinomial Logistic Regression)



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
Function

Maximize probability of correct class

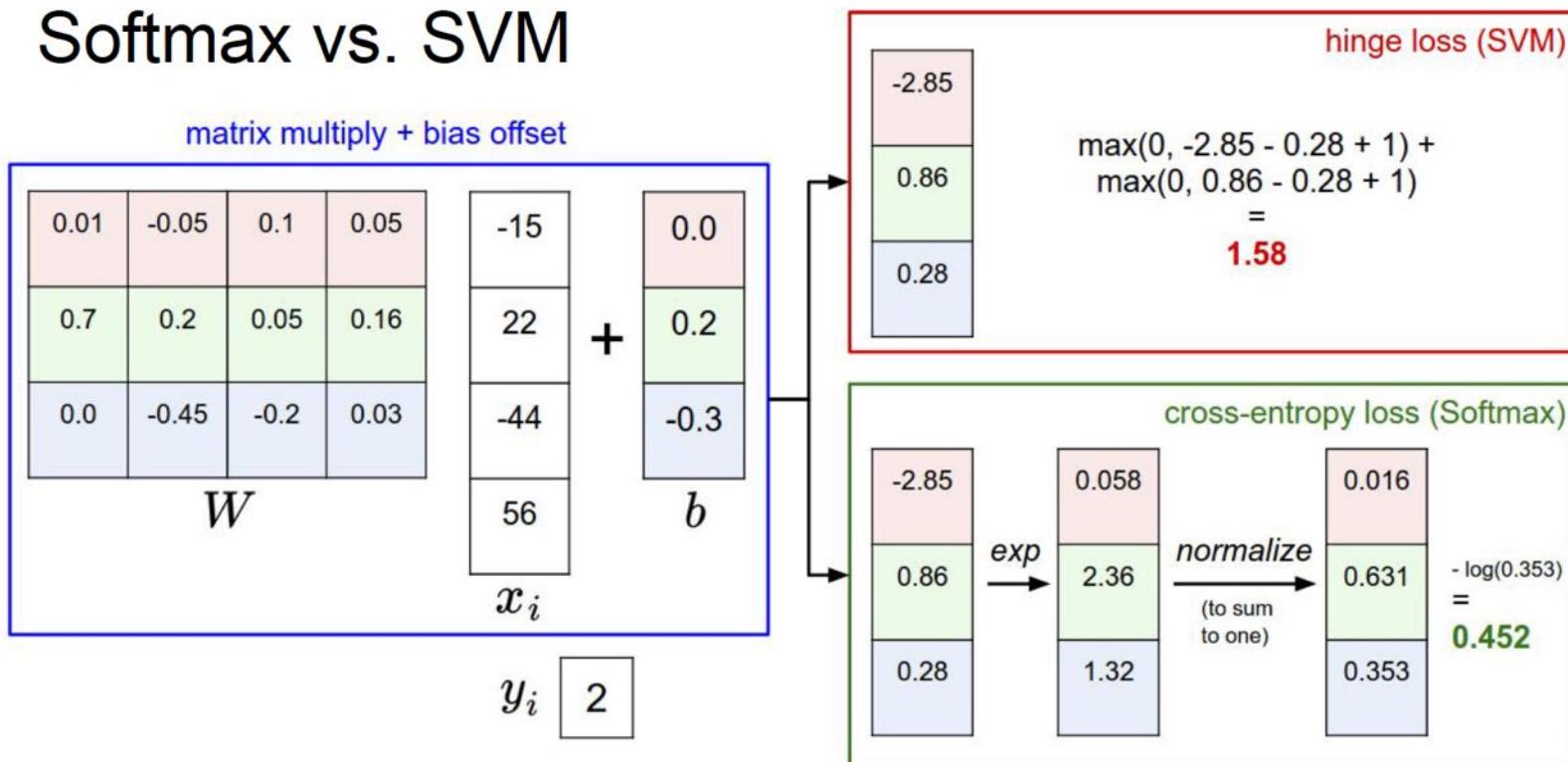
Putting it all together:

$$L_i = -\log P(Y = y_i|X = x_i) \quad L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

cat	3.2
car	5.1
frog	-1.7

Q2: At initialization all s will be approximately equal; what is the loss?
A: $\log(C)$, eg $\log(10) \approx 2.3$

Softmax vs. SVM



Recap... (as computational graph)

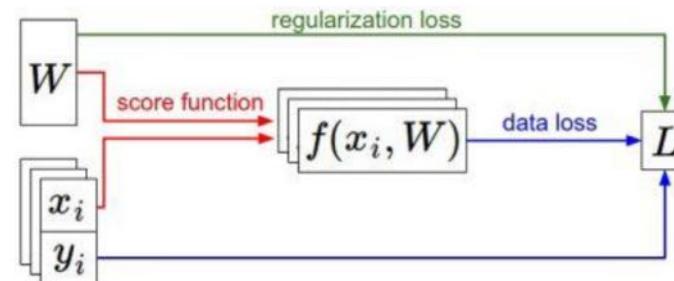
How do we find the best W ?

- We have some dataset of (x, y)
- We have a **score function**: $s = f(x; W) = Wx$ e.g.
- We have a **loss function**:

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right) \quad \text{Softmax}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad \text{SVM}$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + R(W) \quad \text{Full loss}$$



Optimization — Strategies

- #1 Random Search
 - ▶ can work...
 - ▶ but very expensive
- #2 “Follow the Slope”
 - ▶ aka: gradient descent...



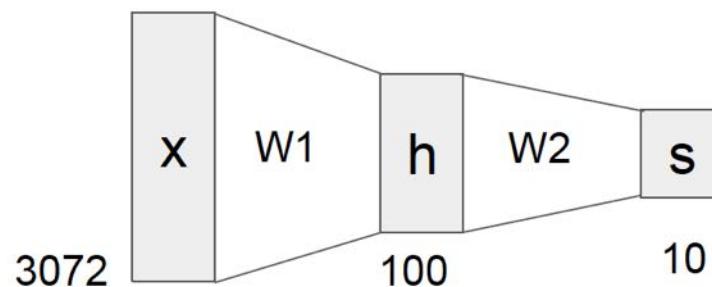
Overview Today's Lecture

- Image Classification
 - ▶ using data-driven approaches (machine learning)
 - ▶ K-nearest neighbor classifier
 - ▶ linear classification (parametric approach)
 - ▶ loss functions and regularization
 - ▶ softmax classifier
 - ▶ optimization via gradient descent
- **What is Deep Learning**
 - ▶ intuition why deep learning can help
 - ▶ integrated learning of features and classifier

Simple Neural Networks

(Before) Linear score function: $f = Wx$

(Now) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$



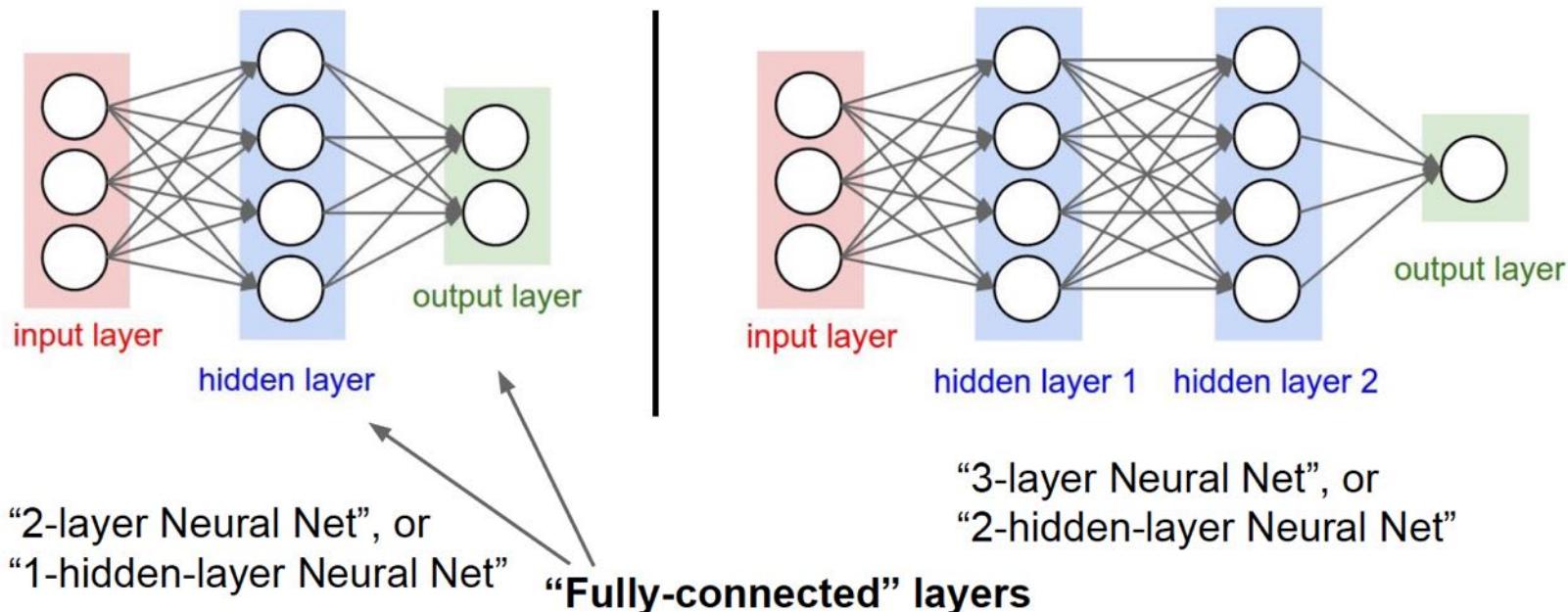
or 3-layer Neural Network

$f = W_3 \max(0, W_2 \max(0, W_1 x))$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Neural Networks - Classic Architectures

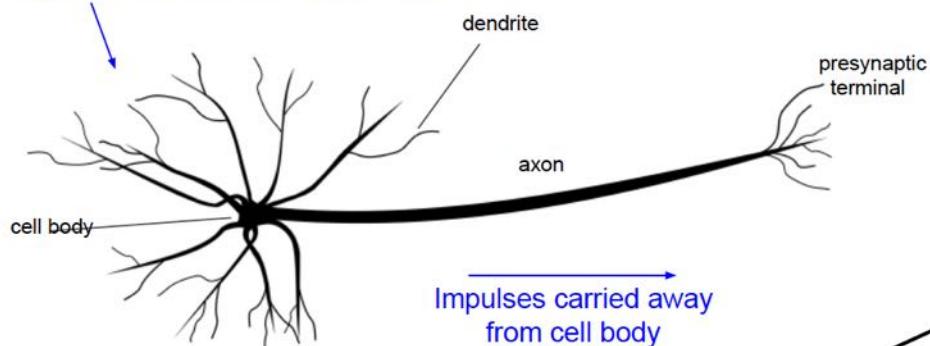
- (Multi-Layer) Perceptron (also called Fully Connected Network/Layers)



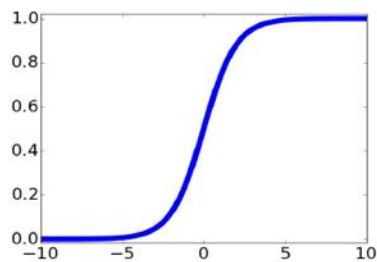
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Some Inspiration from the (Human) Brain

Impulses carried toward cell body



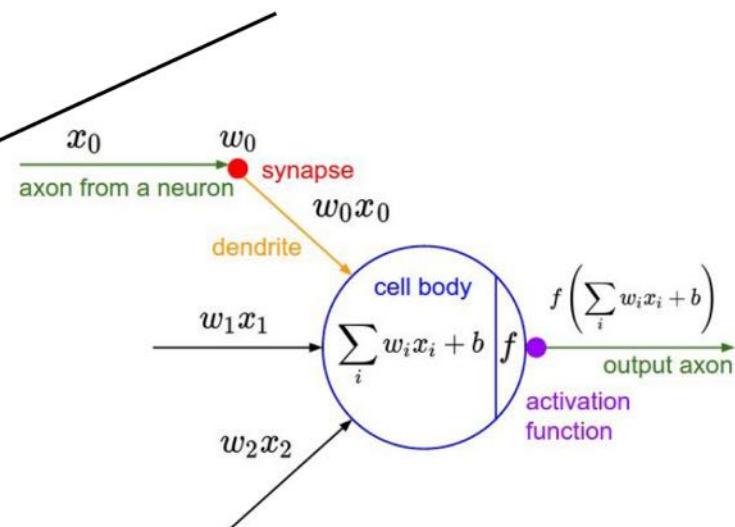
This image by Felipe Perucho
is licensed under CC-BY 3.0



sigmoid activation function

$$\frac{1}{1 + e^{-x}}$$

Impulses carried away from cell body



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Some Inspiration from the (Human) Brain

Be very careful with your brain analogies!

Biological Neurons:

- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system
- Rate code may not be adequate

[Dendritic Computation. London and Häusser]

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

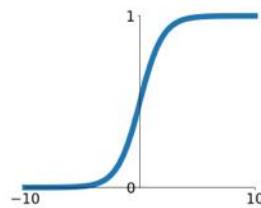


Activation Functions

Activation functions

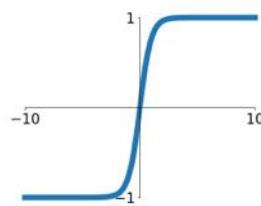
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



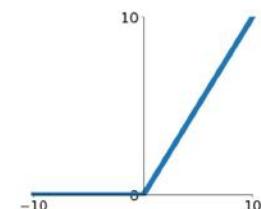
tanh

$$\tanh(x)$$



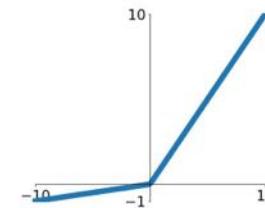
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

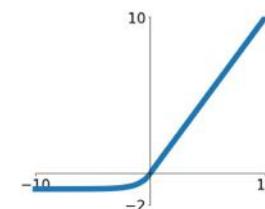


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



slide credit: Fei-Fei, Justin Johnson, Serena Yeung



Summary

- Neural Networks
 - We arrange neurons into fully-connected layers
 - The abstraction of a **layer** has the nice property that it allows us to use efficient vectorized code (e.g. matrix multiplies)
 - Neural networks are not really *neural*

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

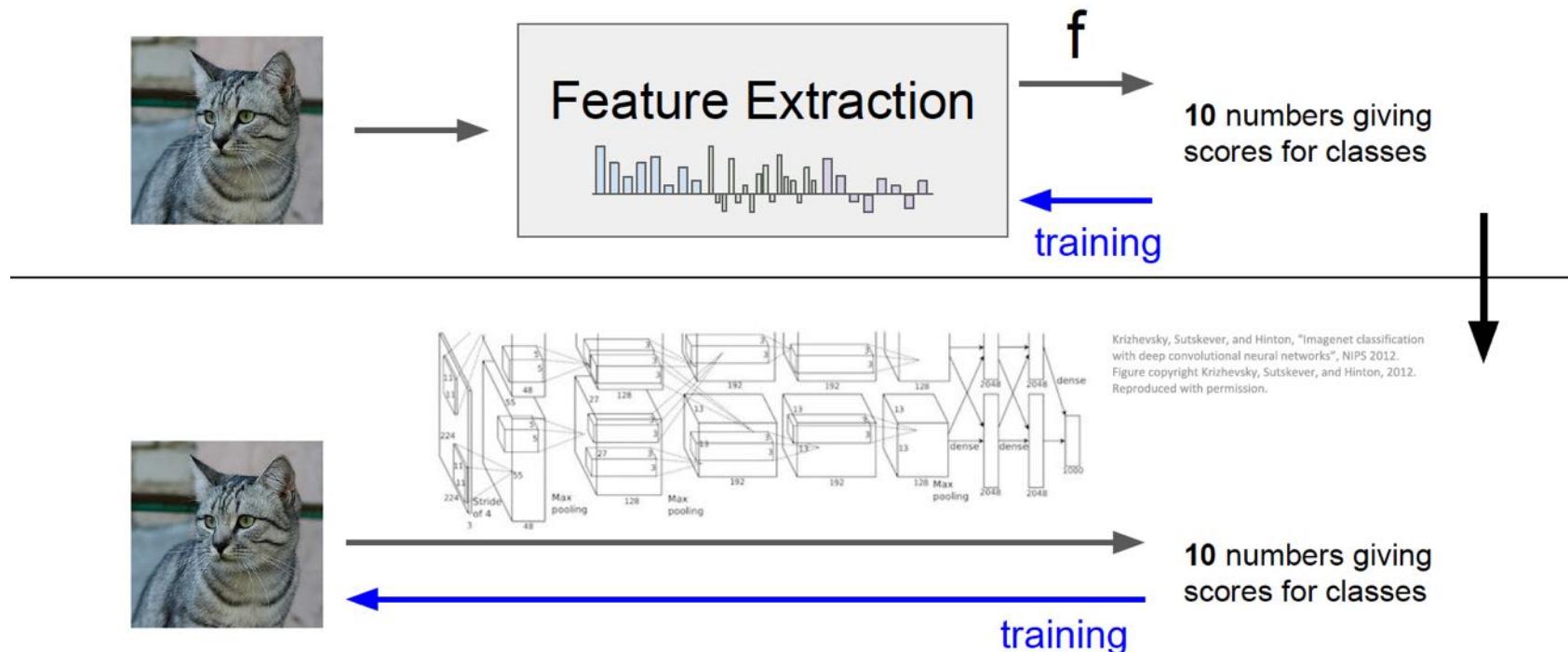


Deep Learning Ingredients

- Deep Learning is based on
 - ▶ Availability of large datasets
 - ▶ Massive parallel compute power
 - ▶ Advances in machine learning over the years
- Strong improvements due to
 - ▶ Internet (availability of large-scale data)
 - ▶ GPUs (availability of parallel compute power)
 - ▶ Deep / hierarchical models with end-to-end learning

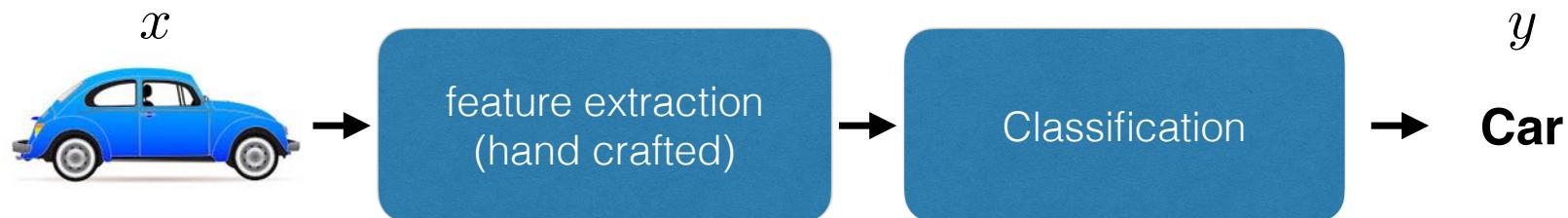


Image Features vs. Deep Learning



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

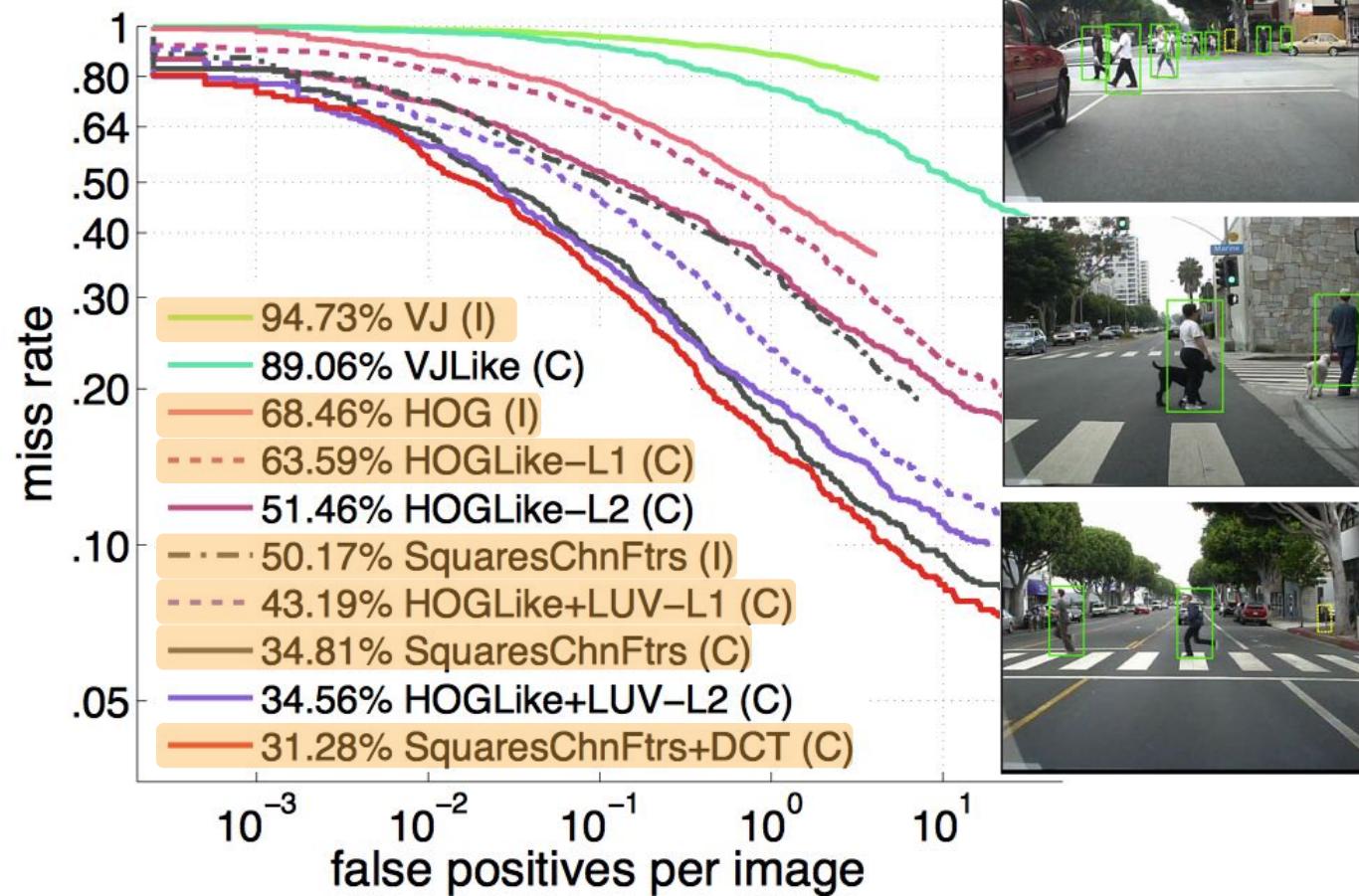
Traditional Approach



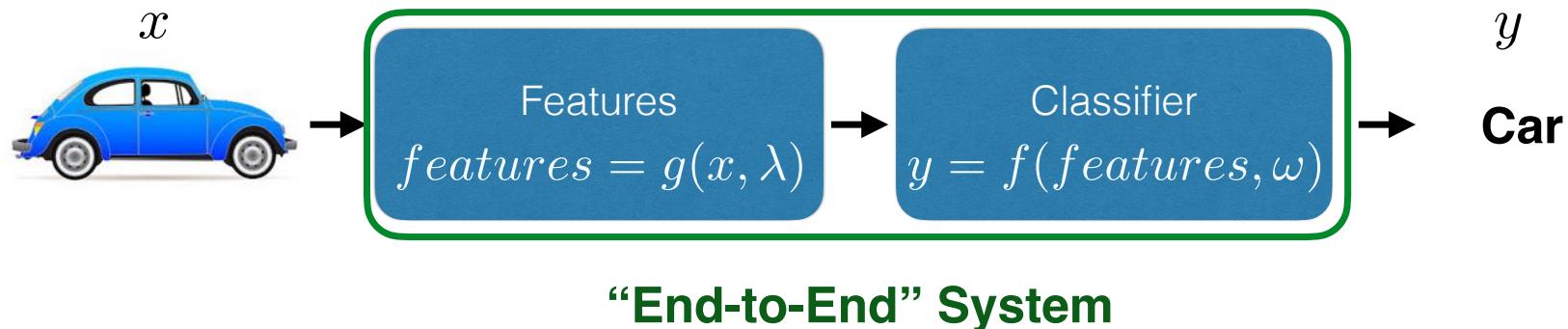
- Feature extraction
 - ▶ often hand crafted and fixed
 - ▶ might be too general (not task-specific enough)
 - ▶ might be too specific (does not generalize to other tasks)
- How to achieve best classification performance
 - ▶ more complex classifier (e.g. multi-feature, non-linear)?
 - ▶ how specialized for the task?

Features Alone Can Explain (Almost) 10 Years of Progress

[Benenson, Omran, Hosang, Schiele @ ECCV workshop'14]

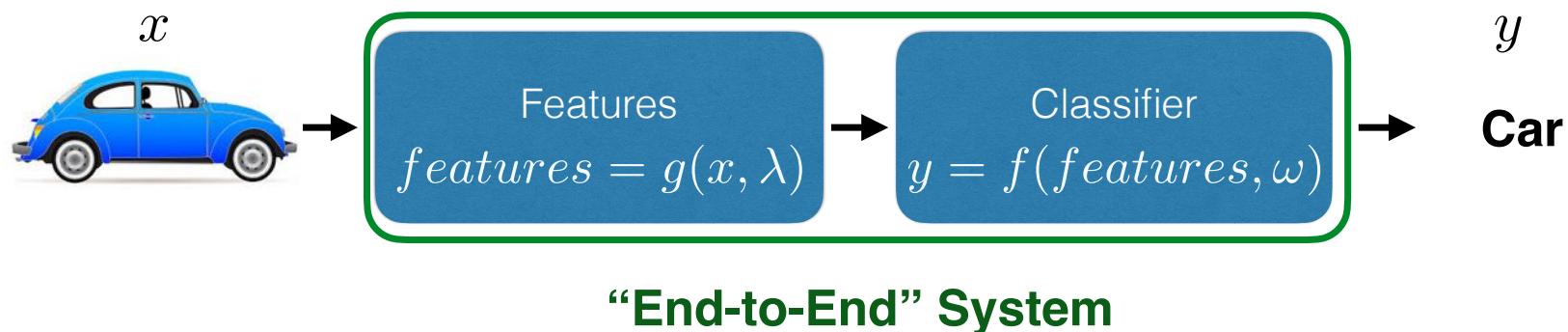


Deep Learning: Trainable Features



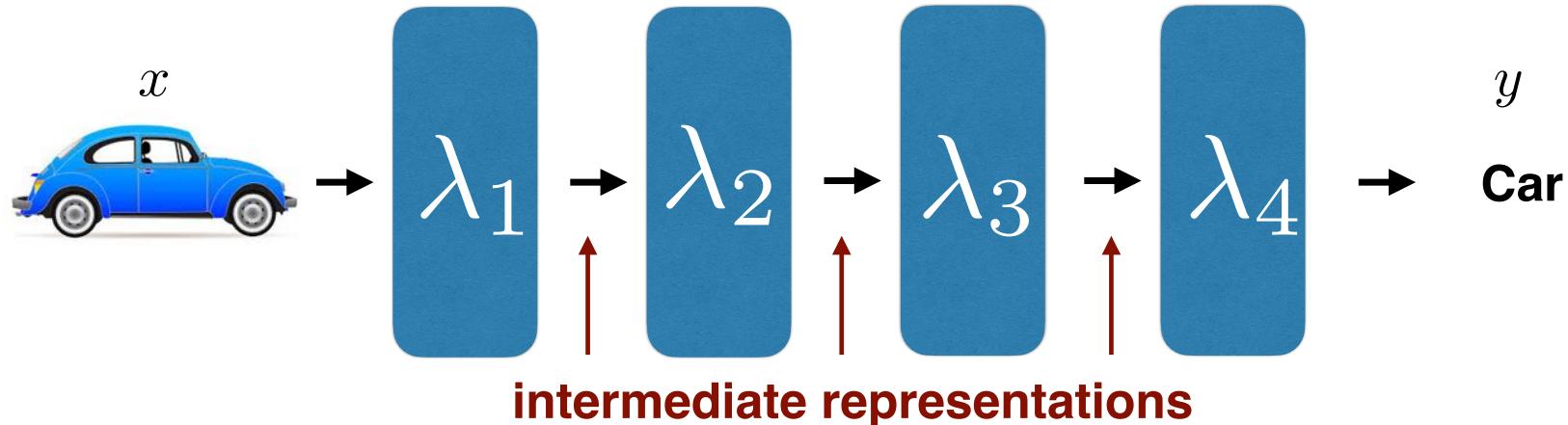
- Parameterized feature extraction
- Features should be
 - ▶ efficient to compute
 - ▶ efficient to train (differentiable)
- **Joint training of **feature** extraction and **classification****
 - ▶ Feature extraction and classification merge into one pipeline

Deep Learning: Joint Training of all Parameters



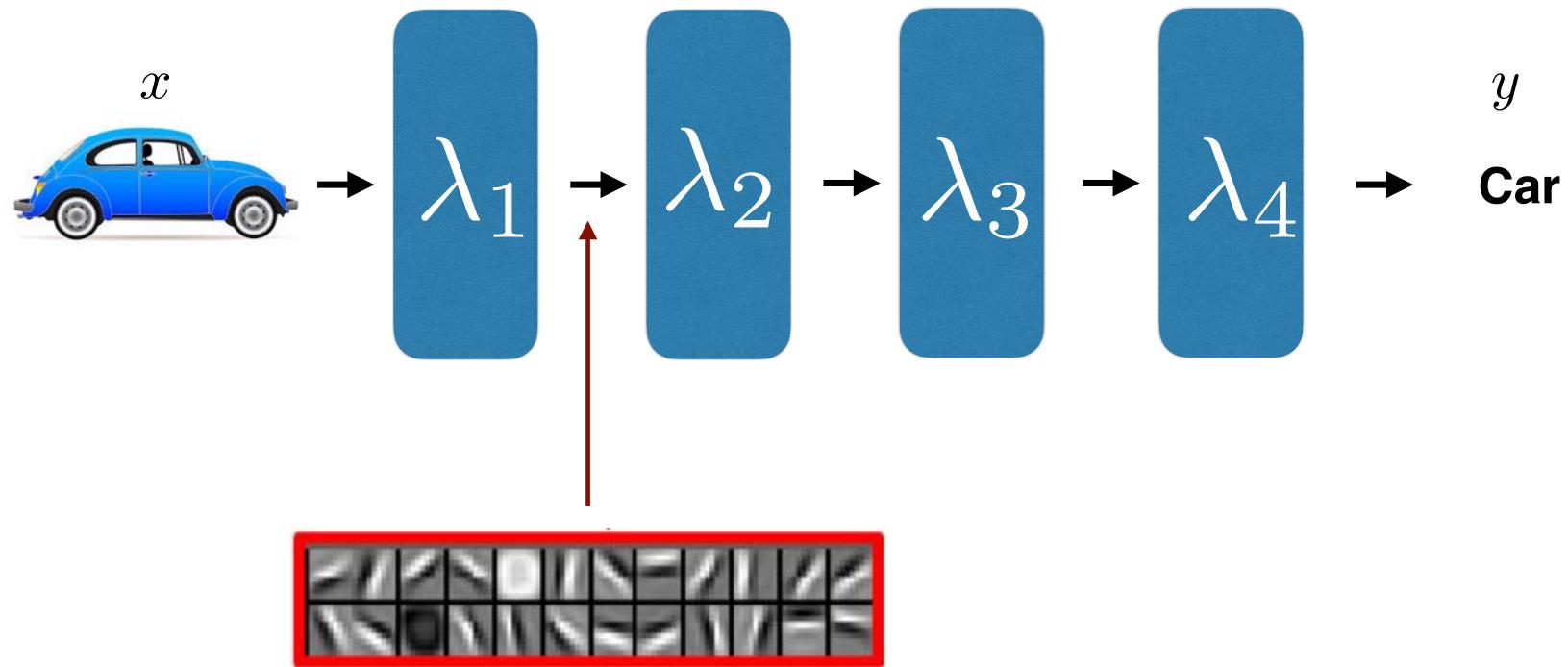
- All parts are adaptive
 - ▶ No differentiation between feature extraction and classification
 - ▶ Non linear transformation from input to desired output
- How can we build such systems?
 - ▶ Composition of simple building blocks can lead to complex systems (e.g. neurons - brain)

Deep Learning: Complex Functions by Composition



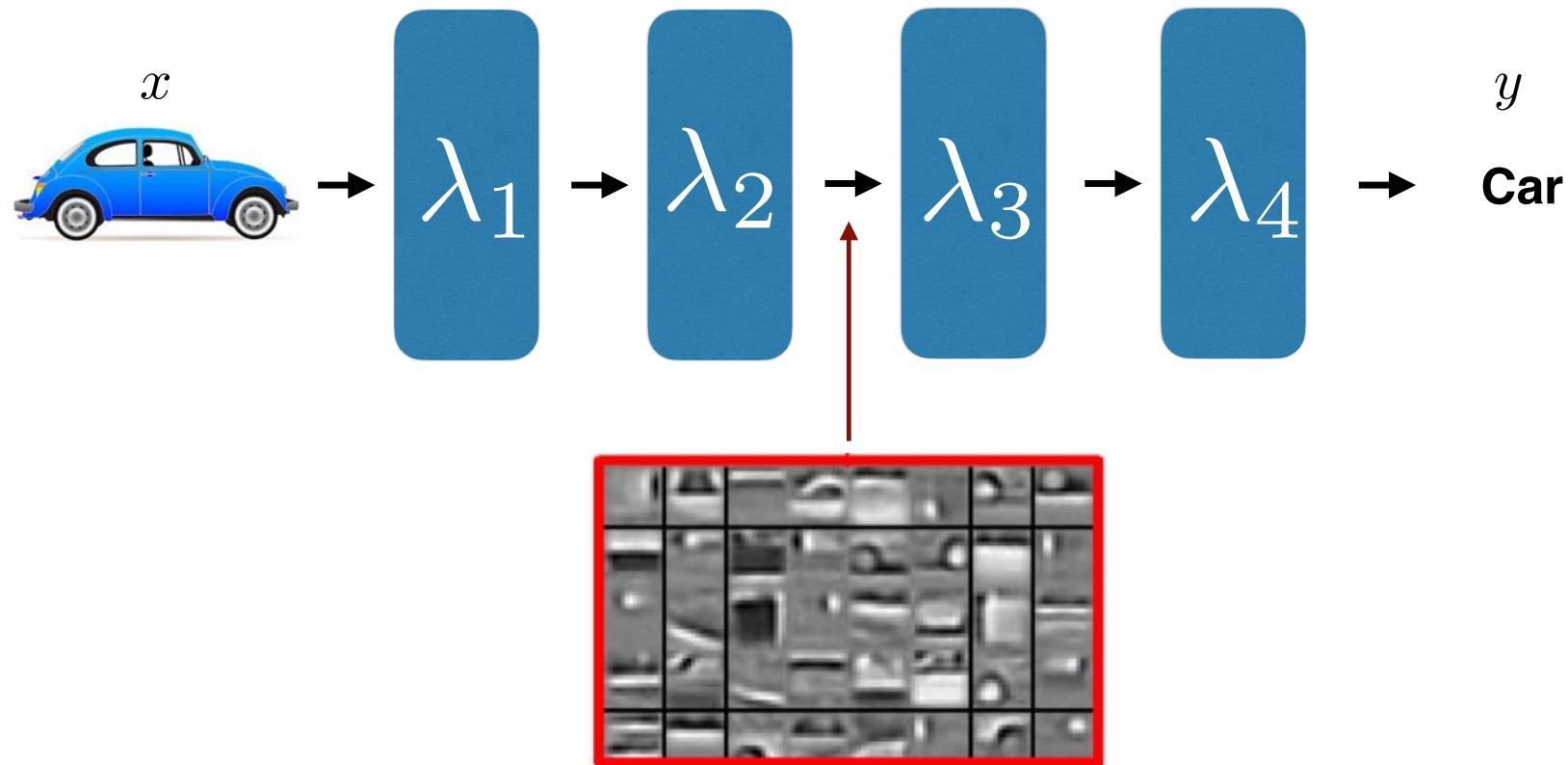
- How can we build such systems?
 - ▶ Composition of simple building blocks can lead to complex systems (e.g. neurons - brain) each block has trainable parameters
 - ▶ Each block has trainable parameters λ_i

Deep Learning: Complex Functions by Composition



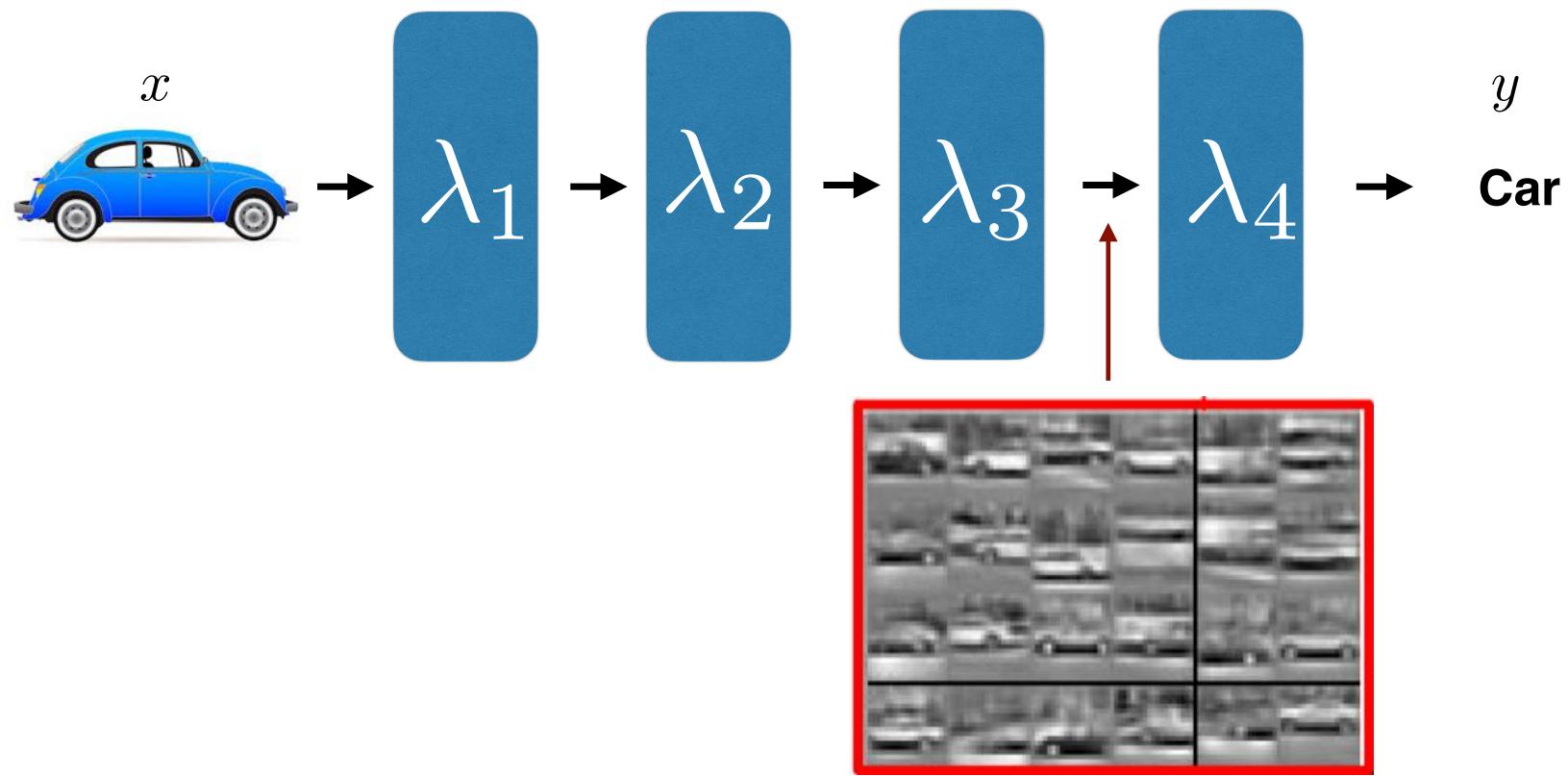
Lee et al. "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations"

Deep Learning: Complex Functions by Composition



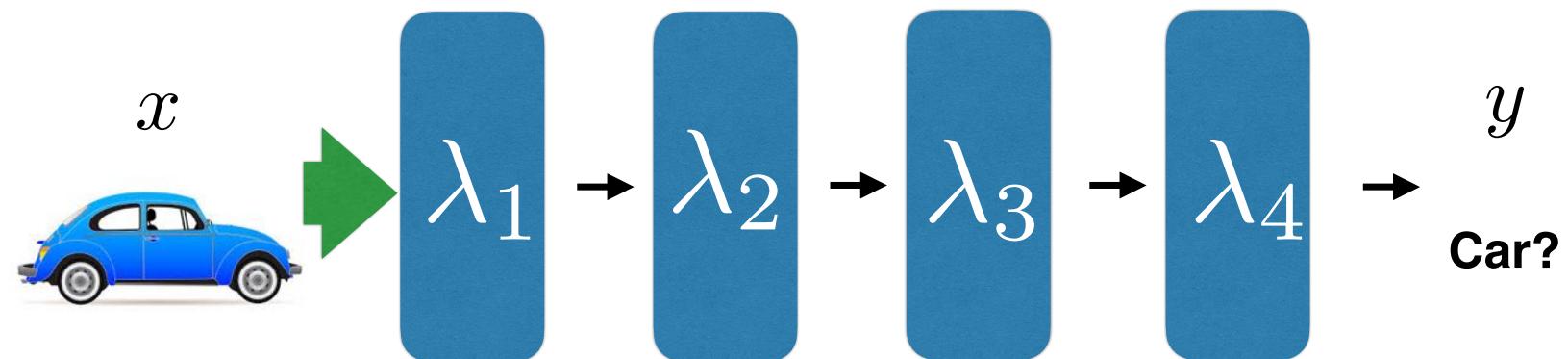
Lee et al. "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations"

Deep Learning: Complex Functions by Composition



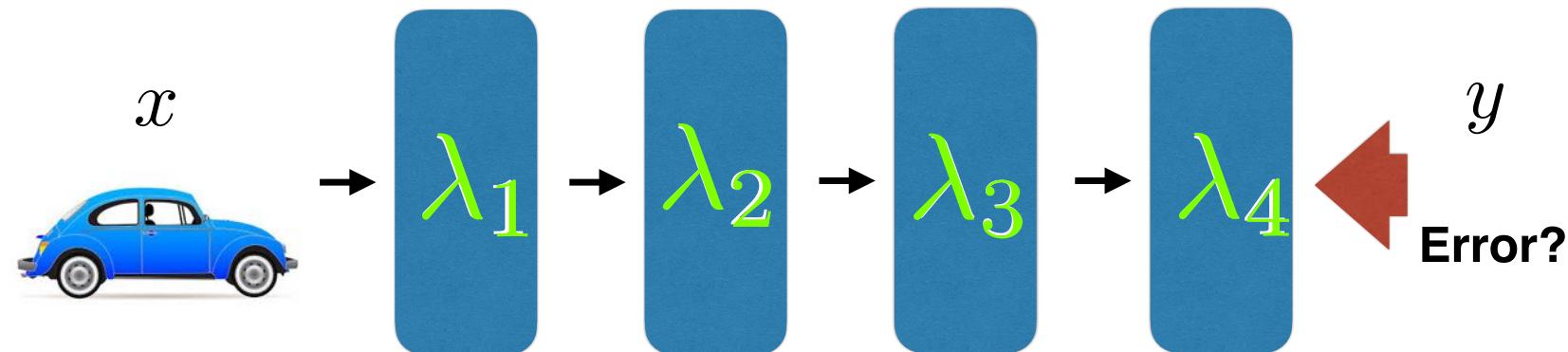
Lee et al. "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations"

Training: Overview



- Setting
 - ▶ generate output y for input x (forward pass)

Training: Overview



- Setting
 - ▶ generate output y for input x (forward pass)
 - ▶ when there is an error, propagate error backwards to update weights (error back propagation)

Summary of Main Ideas in Deep Learning

1. **Learning of feature extraction** (across many layers)
2. **Efficient and trainable** systems by **differentiable** building blocks
3. Composition of deep architectures via **non-linear modules**
4. “**End-to-End**” training: no differentiation between feature extraction and classification