Deep Learning Seminar

2. Linear Classification

Contents

1. Computer Vision

2. Classification

    2-1) Classification

    2-2) Nearest Neighbor

    2-3) Cross-Validation

3. Linear Classifier

4. Loss Function

    4-1) Hinge Function

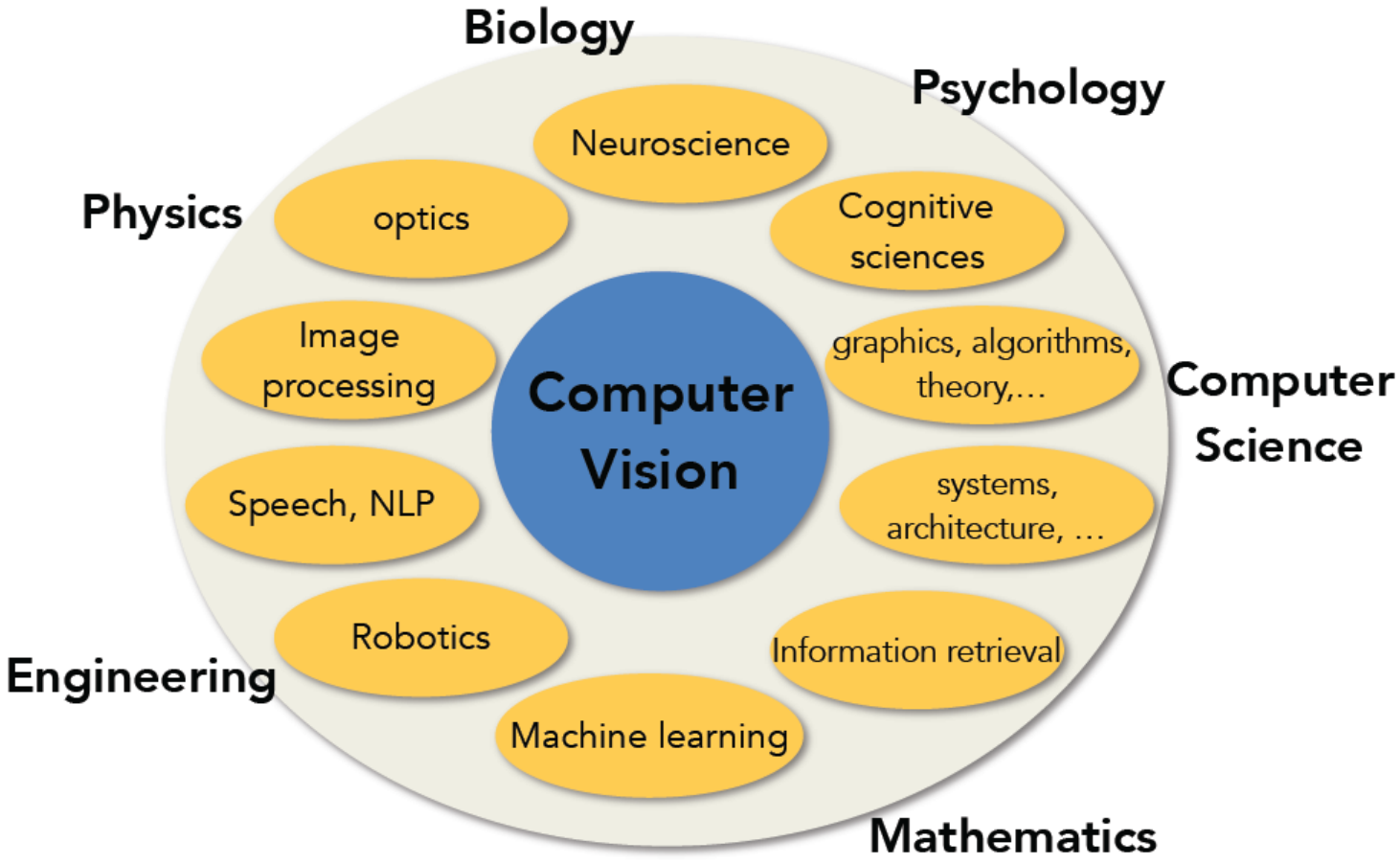
    4-2) Softmax Function

    4-3) Regularization

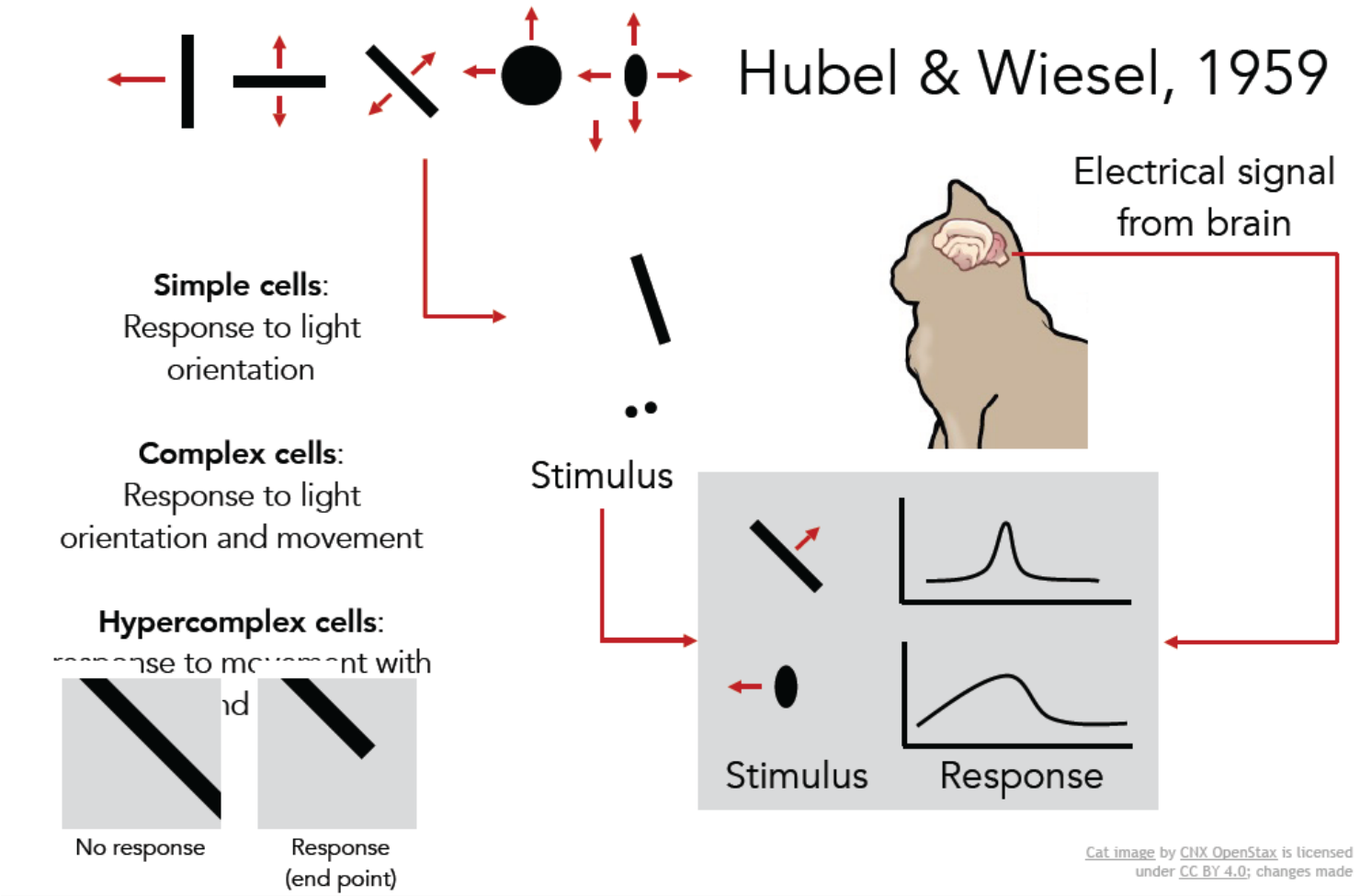
Reference: lecture note (Fei-Fei Li)
lecture note (Andrew Ng)
모두를 위한 머신러닝 (Sung kim)

1. Computer Vision

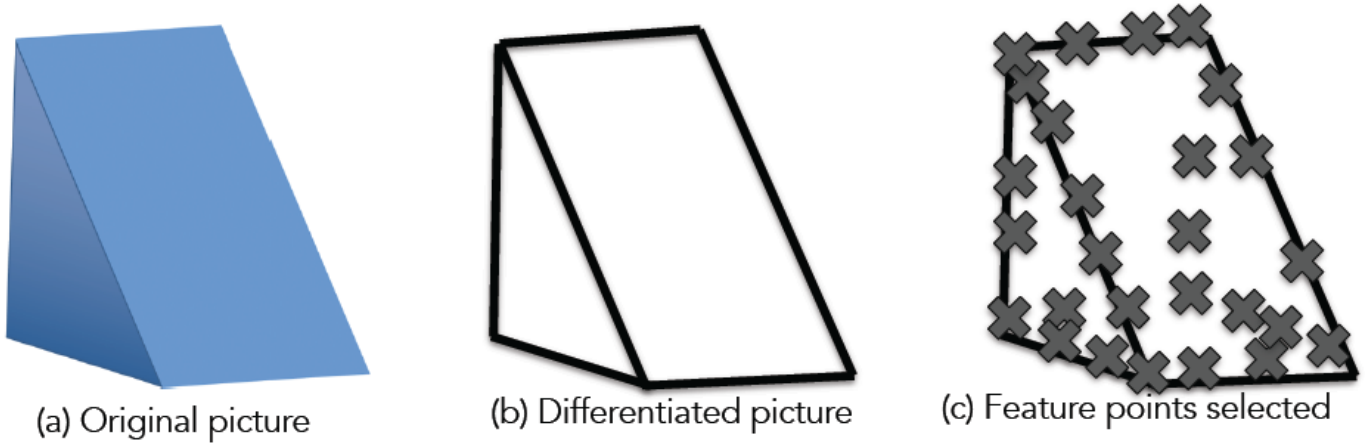
Computer Vision



Hubel & Wiesel, 1959



Edge


(a) Original picture
(b) Differentiated picture
(c) Feature points selected

Edge

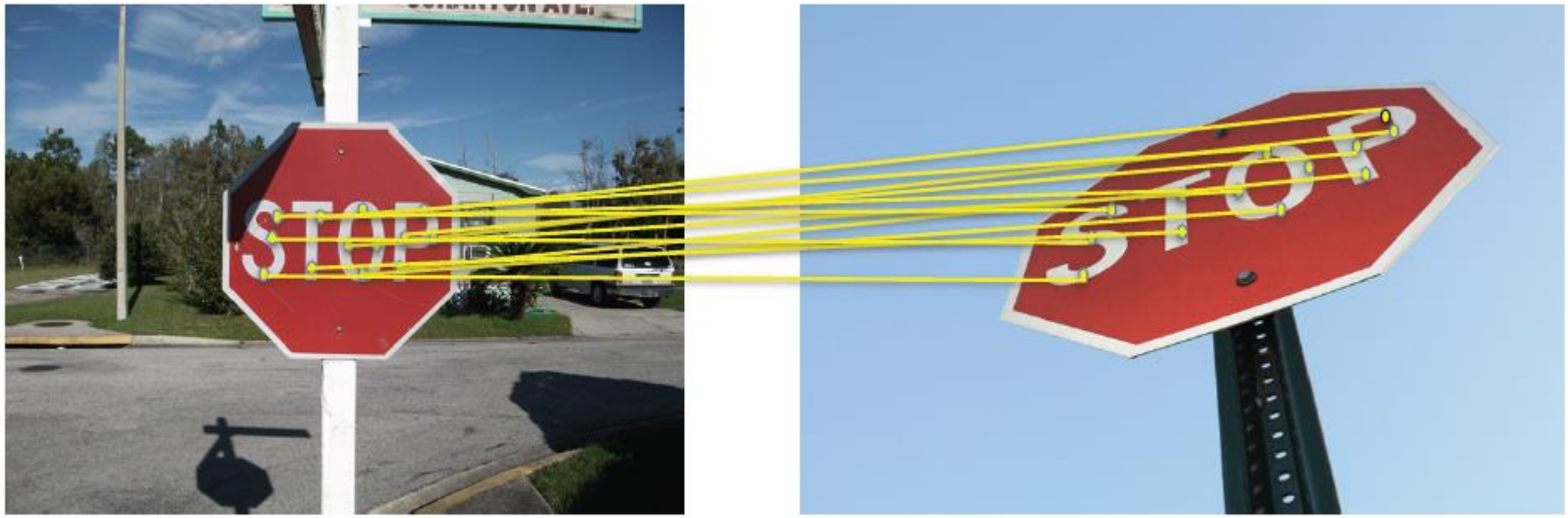
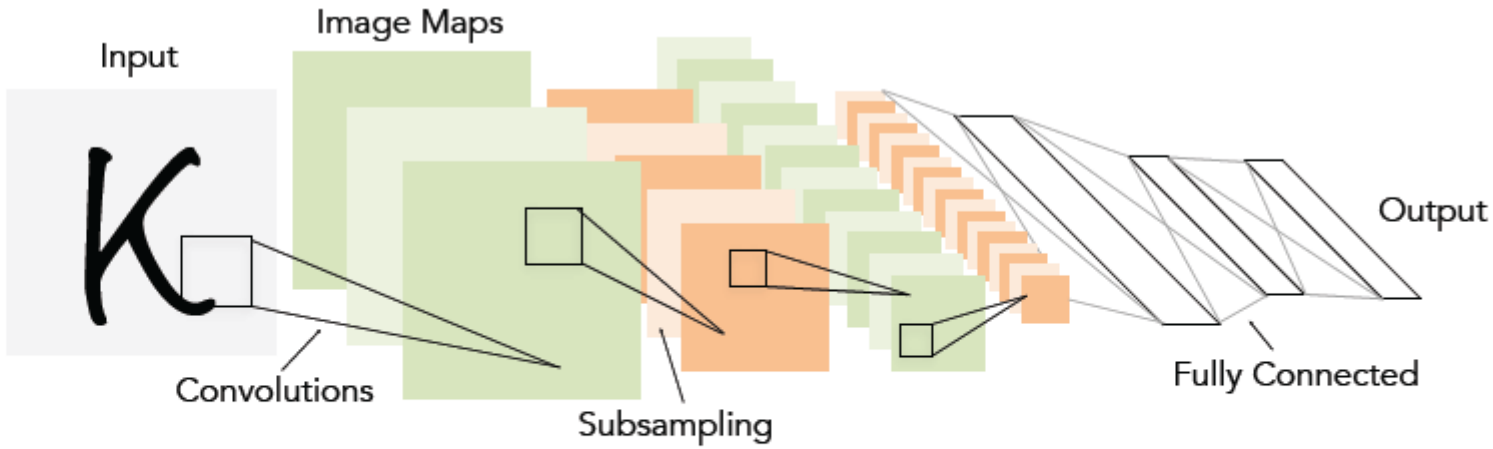


Image is public domain

Image is CC BY-SA 2.0

Convolutional Neural Network

1998
LeCun et al.



# of transistors

10^6

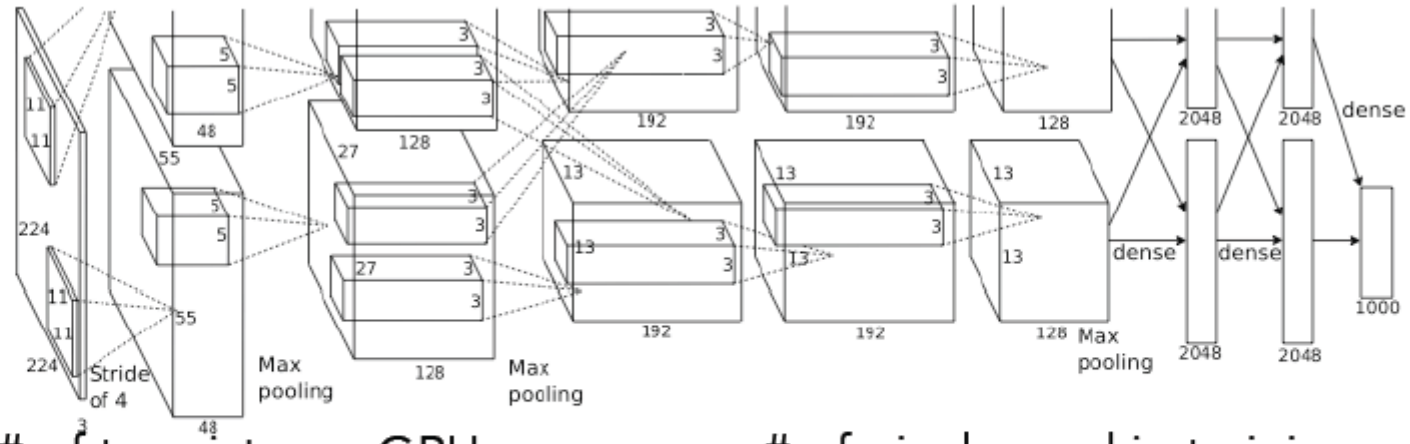
# of pixels used in training

10^7 NIST

Convolutional Neural Network

2012

Krizhevsky et al.



# of transistors

10^9

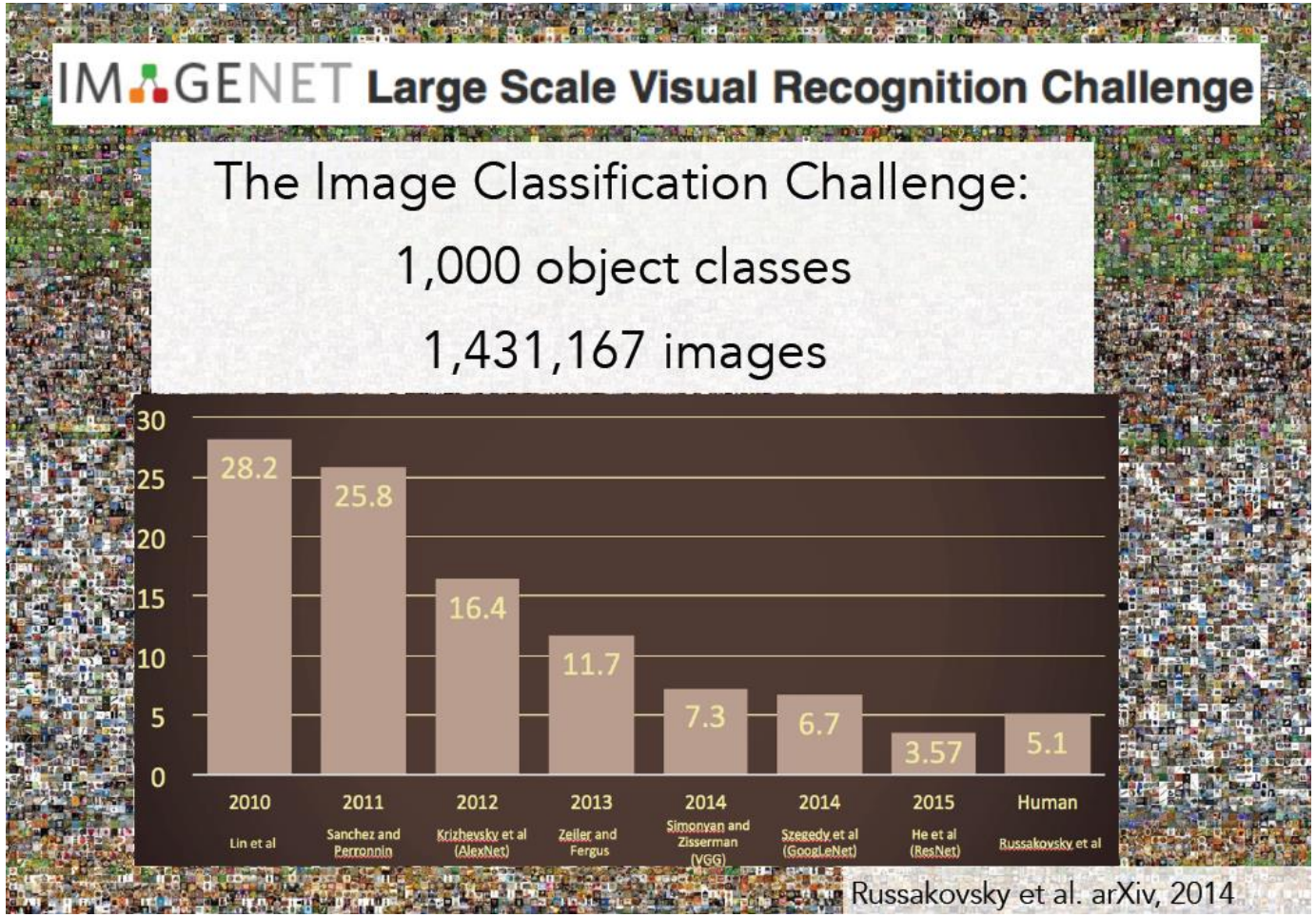
GPUs

# of pixels used in training

10^{14}

IMAGENET

IMAGENET


IMAGENET Large Scale Visual Recognition Challenge
The Image Classification Challenge:
1,000 object classes
1,431,167 images
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.57
5.1
2010
Lin et al
2011
Sanchez and Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler and Fergus
2014
Simonyan and Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
Human
Russakovsky et al
Russakovsky et al. arXiv, 2014

2. Classification

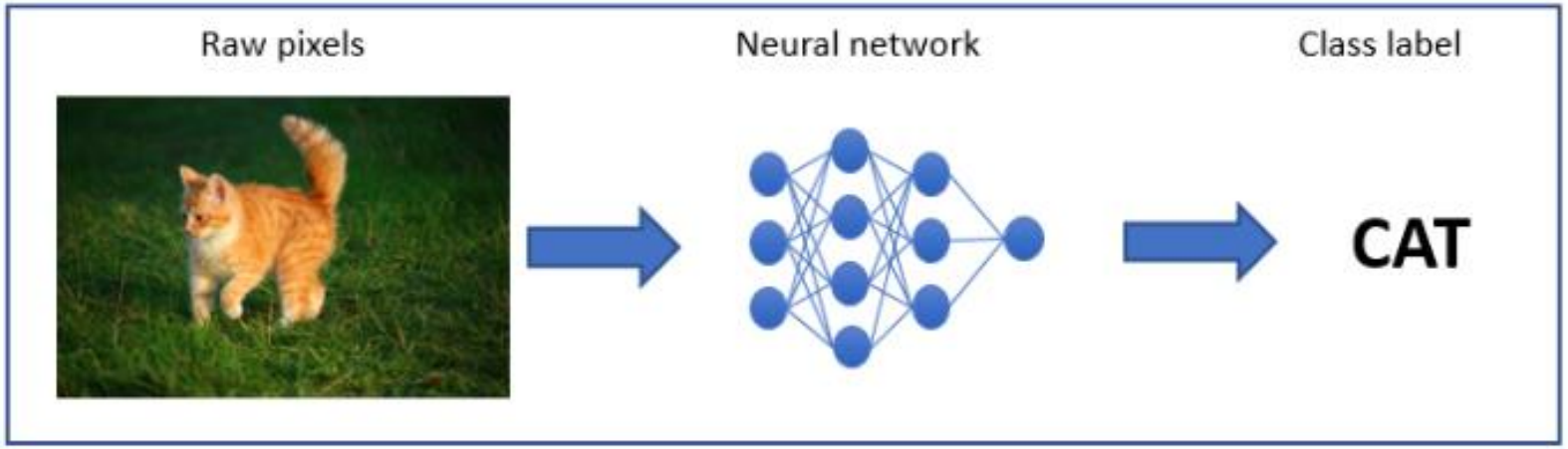
2-1) Classification

2-2) Nearest Neighbor

2-3) Cross-Validation

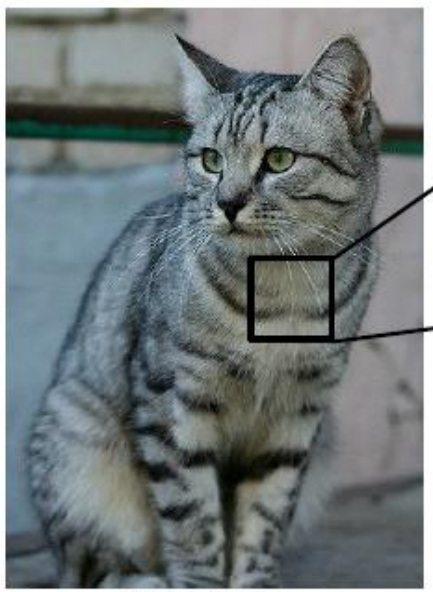
Classification

- Example



Classification

The Problem: Semantic Gap




```
[[105 112 108 111 104  99 106  90  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 120 105  87  96  95  99 115 112 106 103  90  85]
 [ 90  81  81  93 120 131 127 108  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  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  78  62  65  63  63  60  73  86 101]
 [125 133 140 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 140 131 110 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 110 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 118 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [110  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  88  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  90  94]
 [138 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 140 153 102  58  78  92 107]
 [122 164 140 103  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)

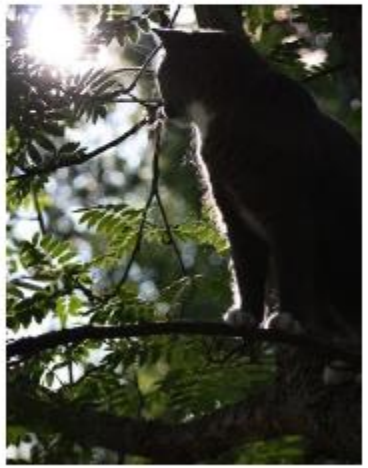
Classification

Challenges: Illumination

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

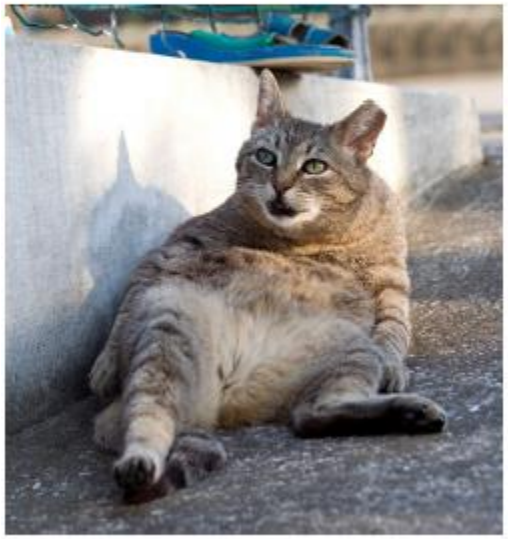
This image is CC0 1.0 public domain

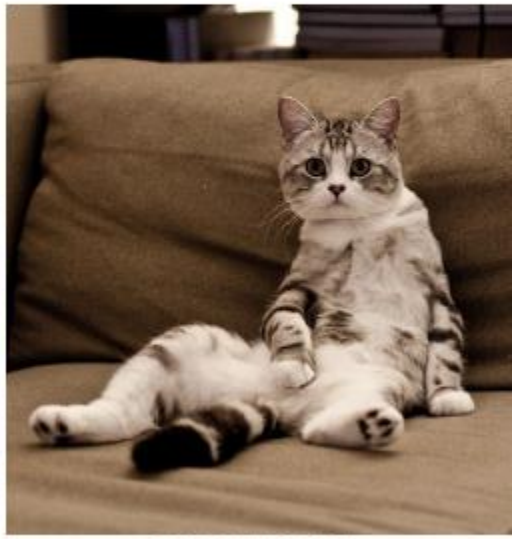


This image is CC0 1.0 public domain

Classification

Challenges: Deformation





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

Classification

Challenges: Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Classification

Challenges: Intraclass variation

Classification

An image classifier

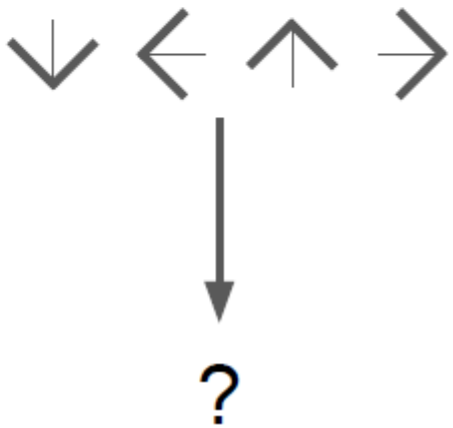
```
def classify_image(image):
  # Some magic here?
  return class_label
```

Classification

Attempts have been made

Find edges

Find corners



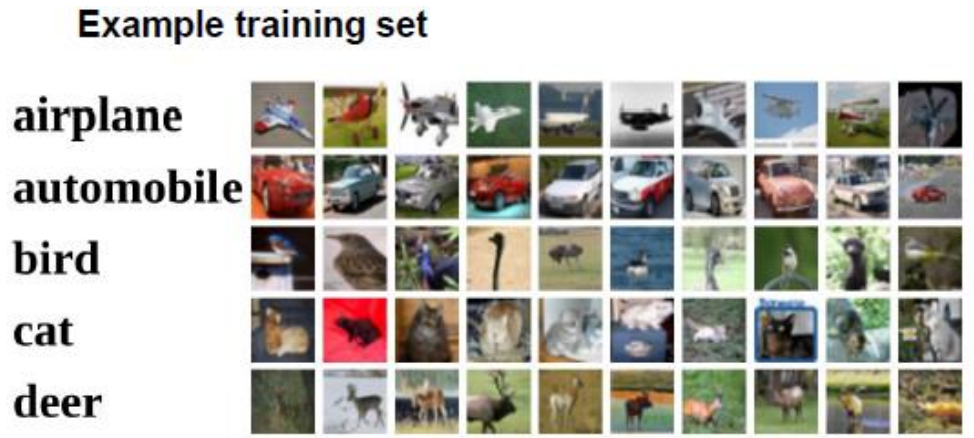
Classification

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

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

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



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

Nearest Neighbor

- How to quantify similarity?

1) L1 Loss

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

2) L2 Loss

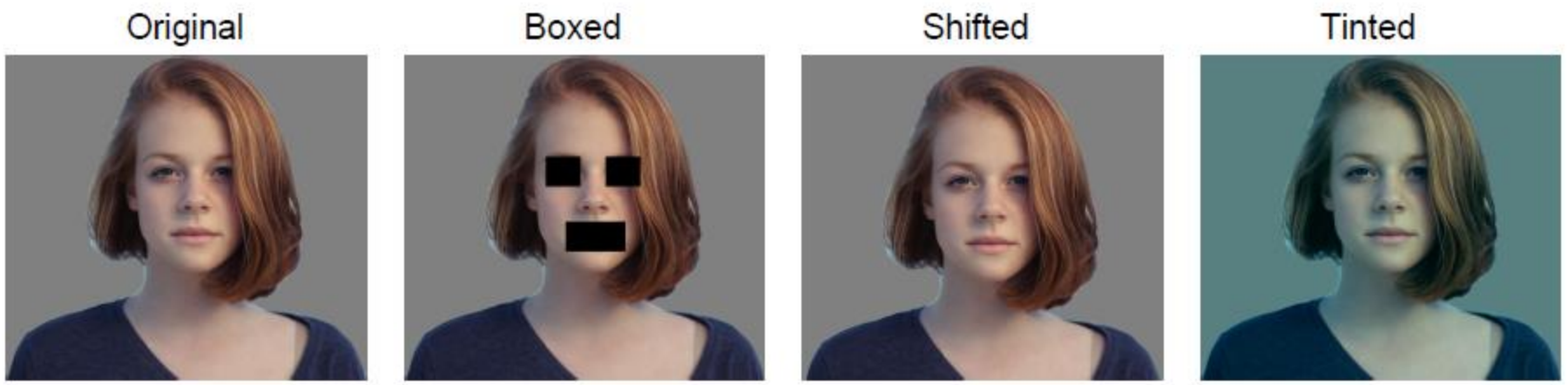
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

test image					training image					pixel-wise absolute value differences				
56	32	10	18	-	10	20	24	17	=	46	12	14	1	add → 456
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	

Nearest Neighbor

k-Nearest Neighbor on images **never used.**

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



Cross-Validation

- Hyperparameters

  Choices about the algorithm that we set rather than learn

  (≈ Heuristic Values)

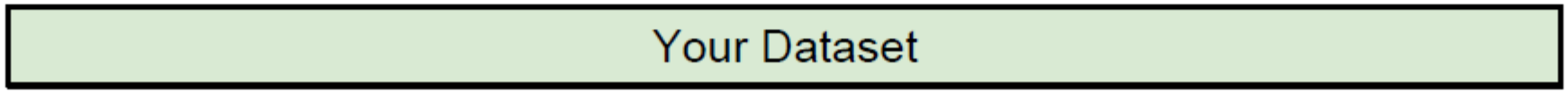
  ex) k-fold cross validation, learning rate, epoch, and number of layers

Cross-Validation

Setting Hyperparameters

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

BAD: K = 1 always works perfectly on training data

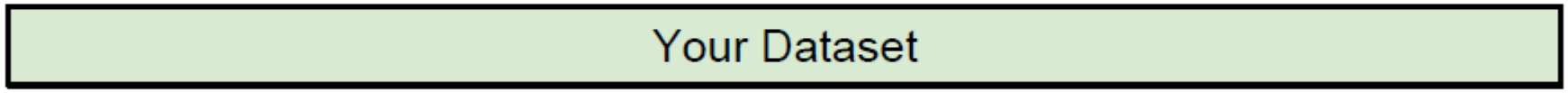


Cross-Validation

Setting Hyperparameters

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

BAD: K = 1 always works perfectly on training data



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

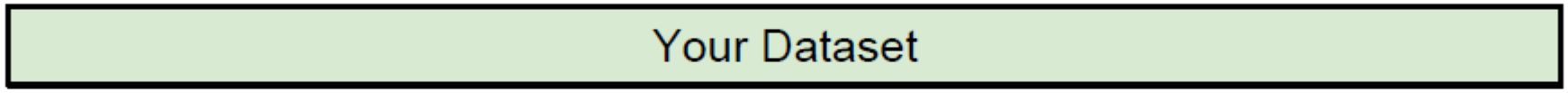
BAD: No idea how algorithm will perform on new data

Cross-Validation

Setting Hyperparameters

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

BAD: K = 1 always works perfectly on training data



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

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

Better!

Cross-Validation

Setting Hyperparameters

Your Dataset

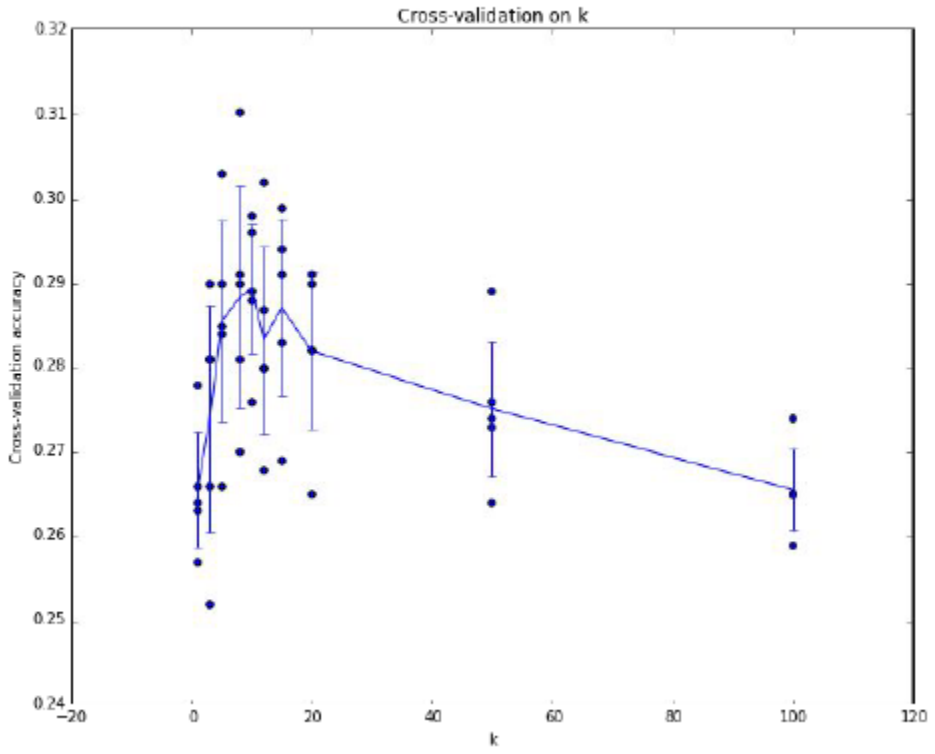
Idea #4: **Cross-Validation**: Split data into **folds**,
try each fold as validation and average the results

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

Cross-Validation

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 ~= 7 works best
for this data)

Classification

- **Deep Learning Pipeline**

1. Training Data Loading

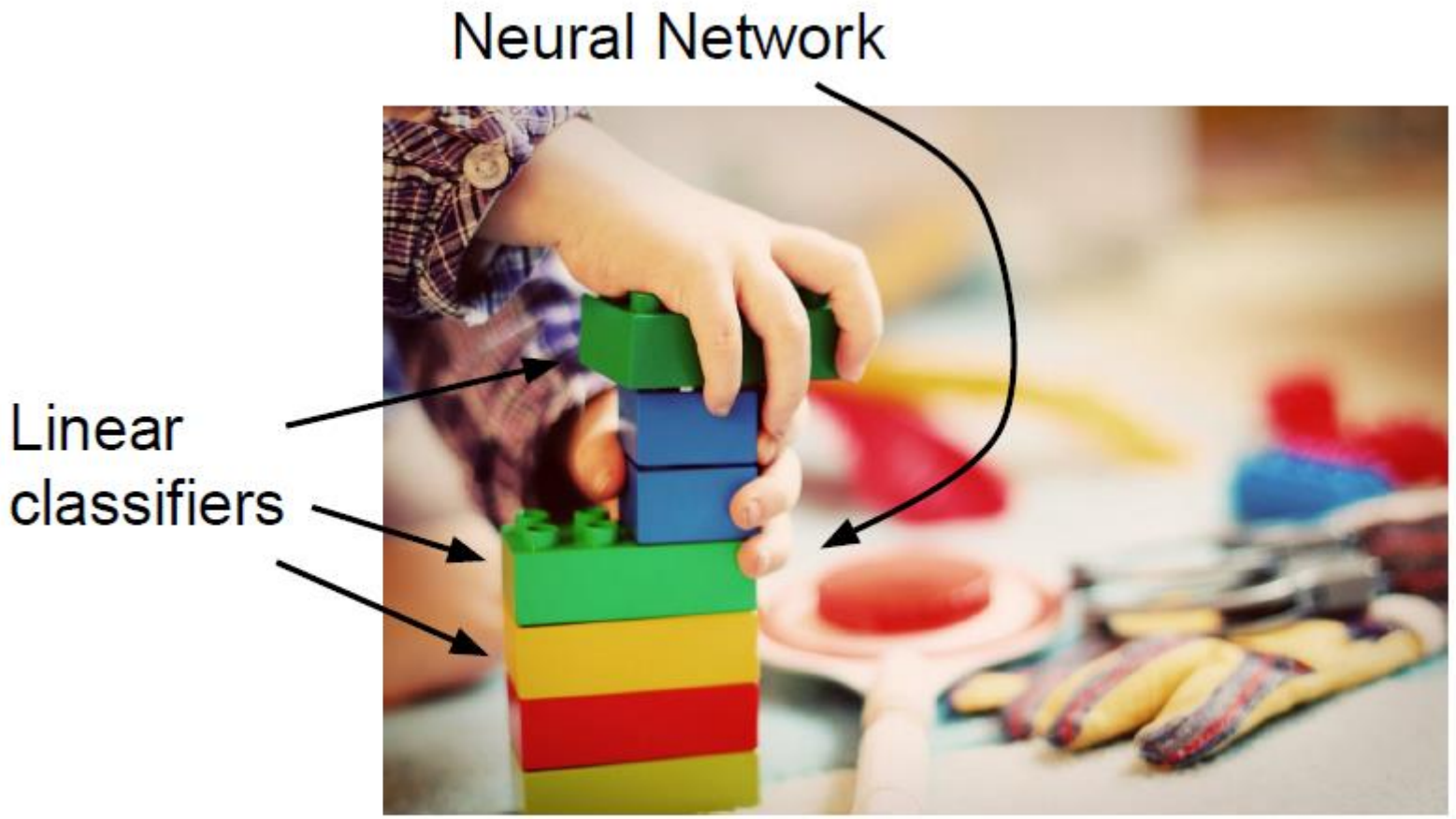
2. Training Data Augmentation

3. Deep Neural Network Training with Training Data
   Validation
4. Deep Neural Network Testing with Testing Data

5. Inference with verified Deep Neural Network

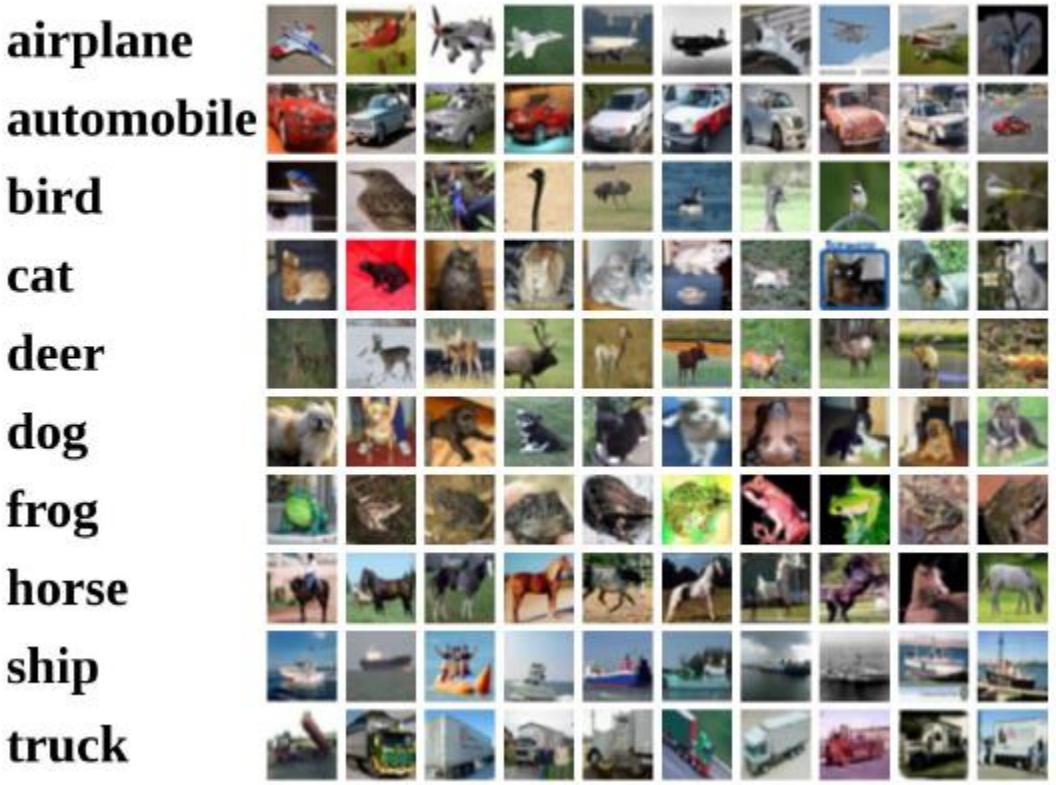
3. Linear Classification

Linear Classification



Linear Classification

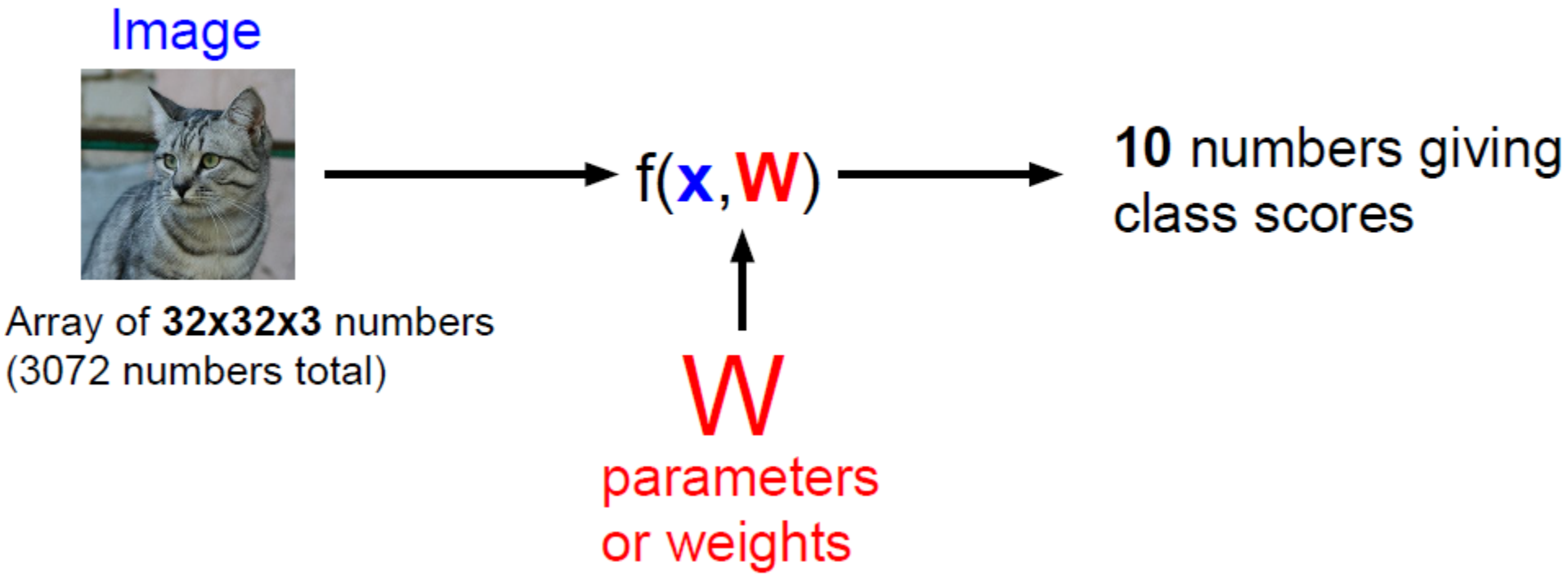
Recall CIFAR10



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

10,000 test images.

Linear Classification



Linear Classification

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

Image

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

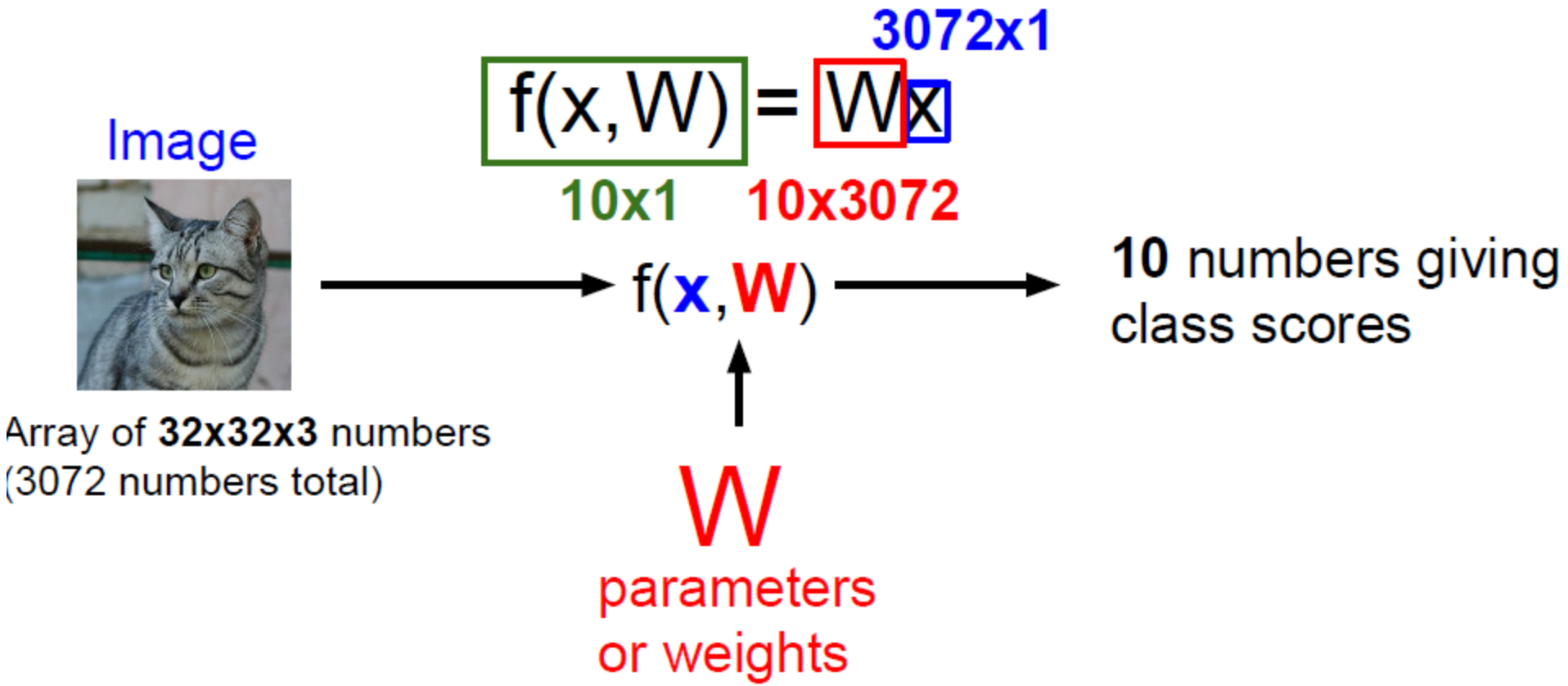
f(**x**,**W**)

10 numbers giving
class scores

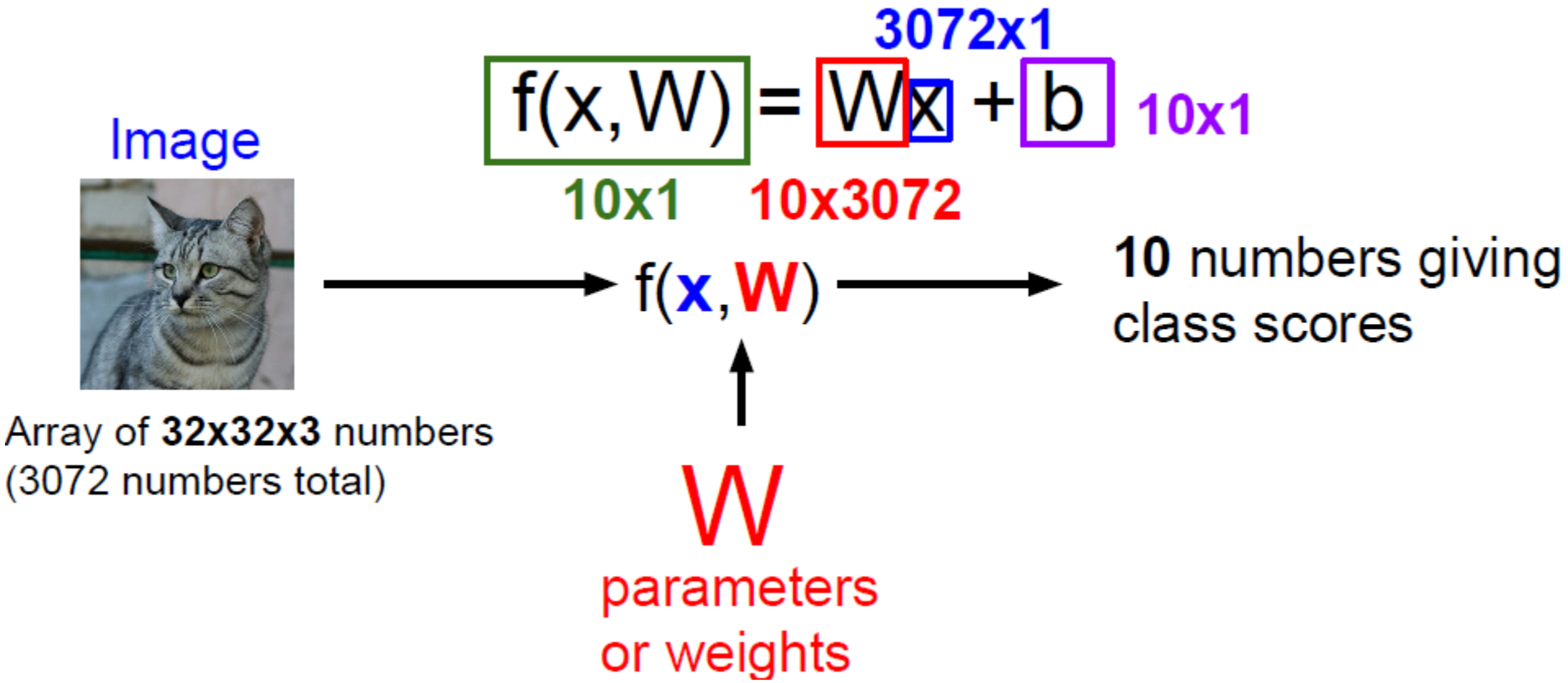
W

parameters
or weights

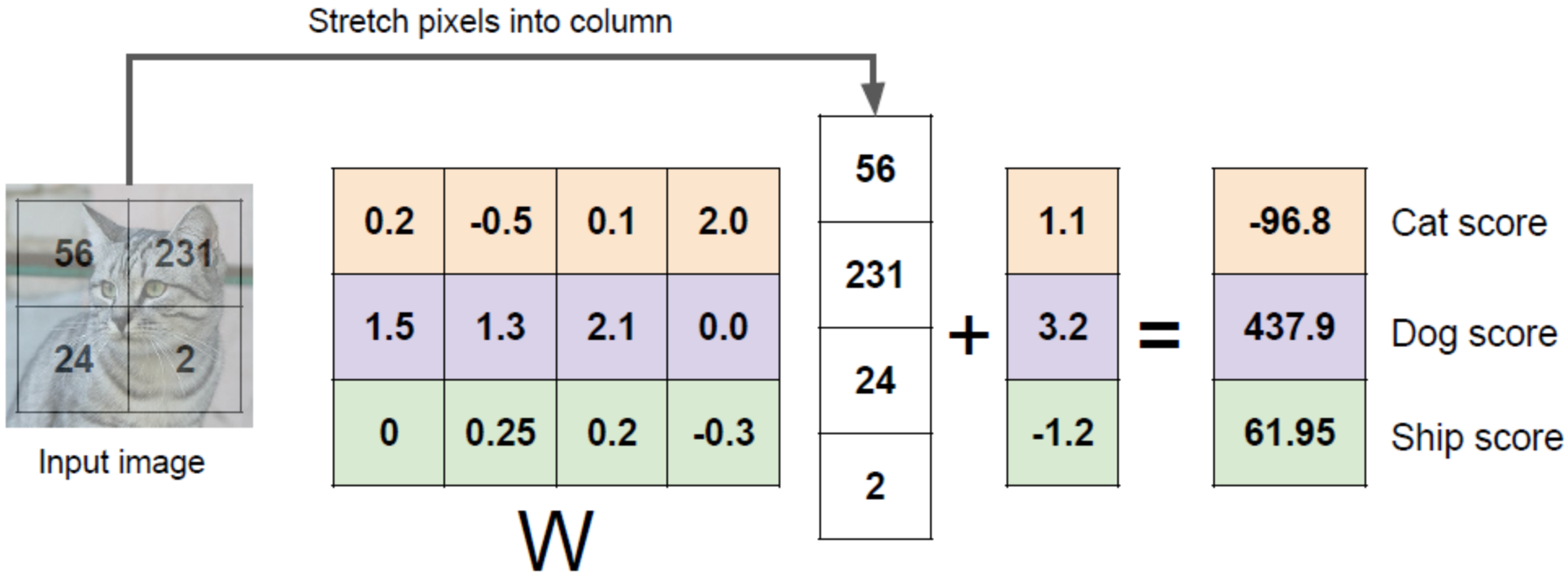
Linear Classification



Linear Classification



Linear Classification



4. Classification Loss Function

4-1) Hinge Loss

4-2) Softmax Loss

4-3) Regularization

Loss Function



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

Loss Function

- Hinge Function



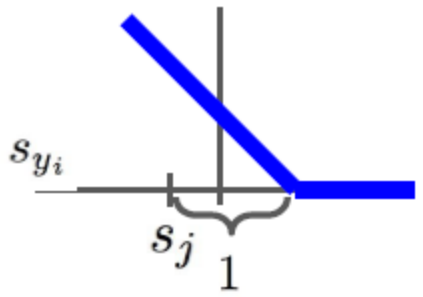
cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

* Hinge Loss (Multiclass SVM Loss)

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

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

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



Loss Function

- **Hinge Function**



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

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

= max(0, 5.1 - 3.2 + 1)
+max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

Loss Function

- Hinge Function



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	

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

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Loss Function

- Hinge Function



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

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

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 6.3) + max(0, 6.6)
= 6.3 + 6.6
= 12.9

Loss Function

- **Hinge Function**



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

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i$$

L = (2.9 + 0 + 12.9)/3
= **5.27**

Loss Function

- Hinge Function



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

Problems

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

Q2: what is the min/max possible loss?

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

Loss Function

- Softmax Function

scores = unnormalized log probabilities of the classes.

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

cat	**3.2**
car	5.1
frog	-1.7

Softmax function

Loss Function

- Softmax Function

unnormalized probabilities

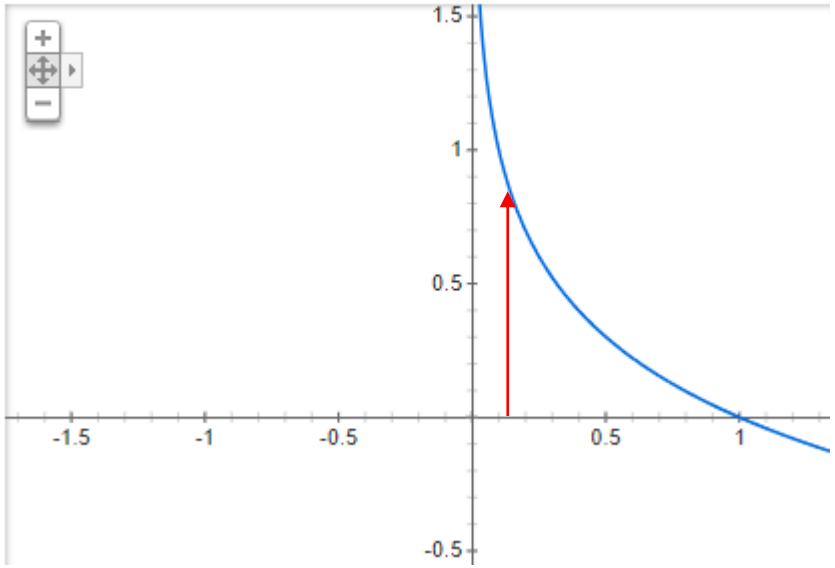
	unnormalized log probabilities	exp	unnormalized probabilities	normalize	probabilities
cat	3.2		24.5		0.13
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

$$= \frac{e^{sy_i}}{\sum_j e^{sj}}$$

Loss Function

- Softmax Function

-log(x) 그래프



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

0.13 = $\frac{e^{sy_i}}{\sum_j e^{s_j}}$

0.87

0.00

probabilities

Loss Function

- **Regularization**

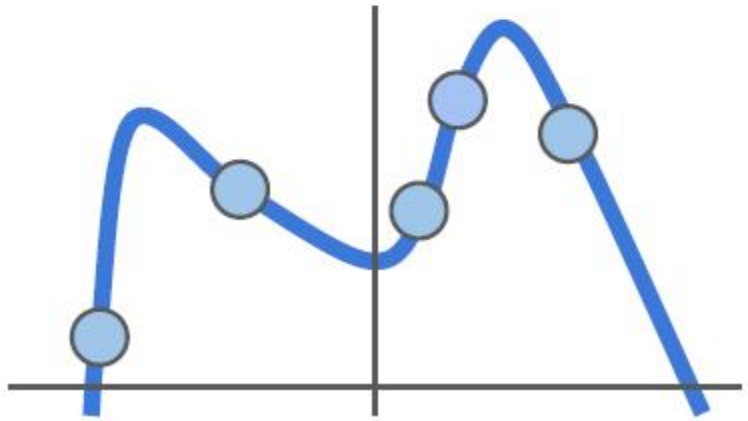
$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}$$

Loss Function

- **Regularization**

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}$$

Data loss: Model predictions should match training data

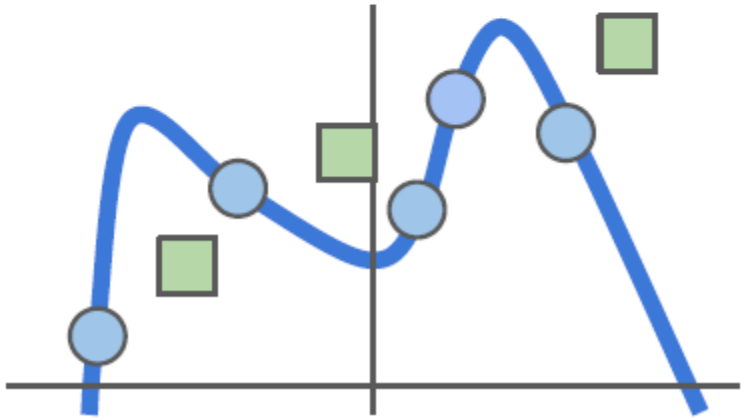


Loss Function

- **Regularization**

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}$$

Data loss: Model predictions should match training data

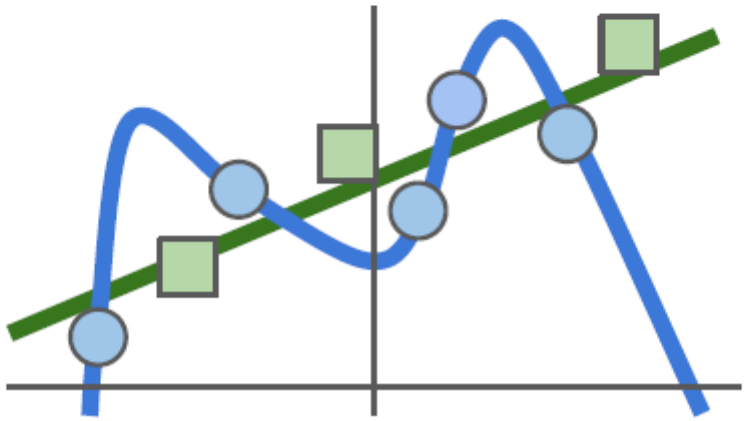


Loss Function

- **Regularization**

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}$$

Data loss: Model predictions should match training data

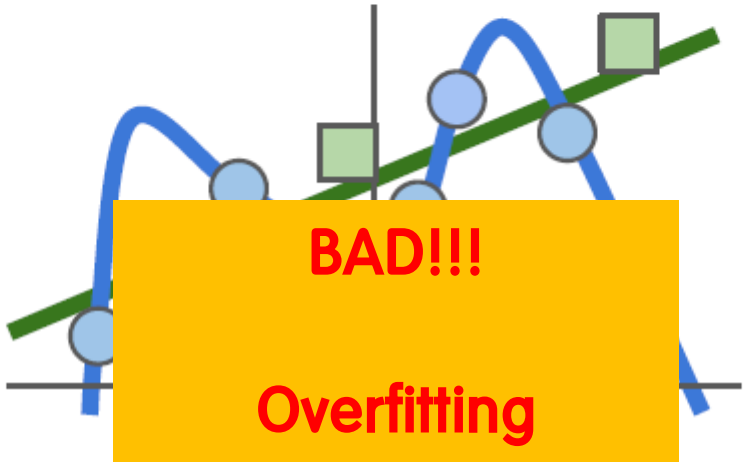


Loss Function

- **Regularization**

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}$$

Data loss: Model predictions should match training data



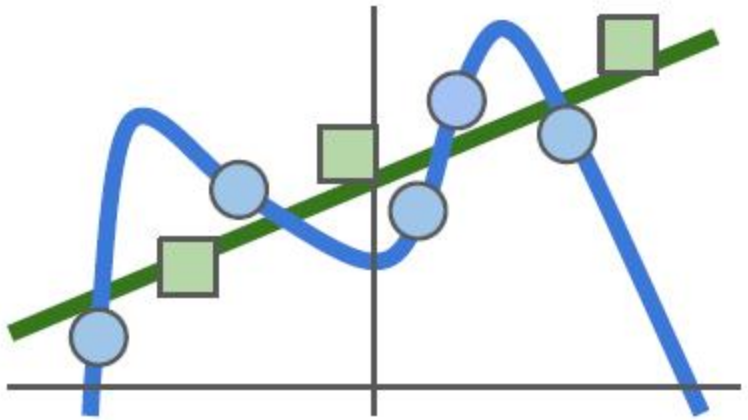
Loss Function

- **Regularization**

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Model should be "simple", so it works on test data



Loss Function

- Regularization

Hyperparameter (Constant)

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

In common use:

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

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

Loss Function

- **Regularization**

$$x = [1, 1, 1, 1]$$

$$w_1 = [1, 0, 0, 0]$$

$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

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

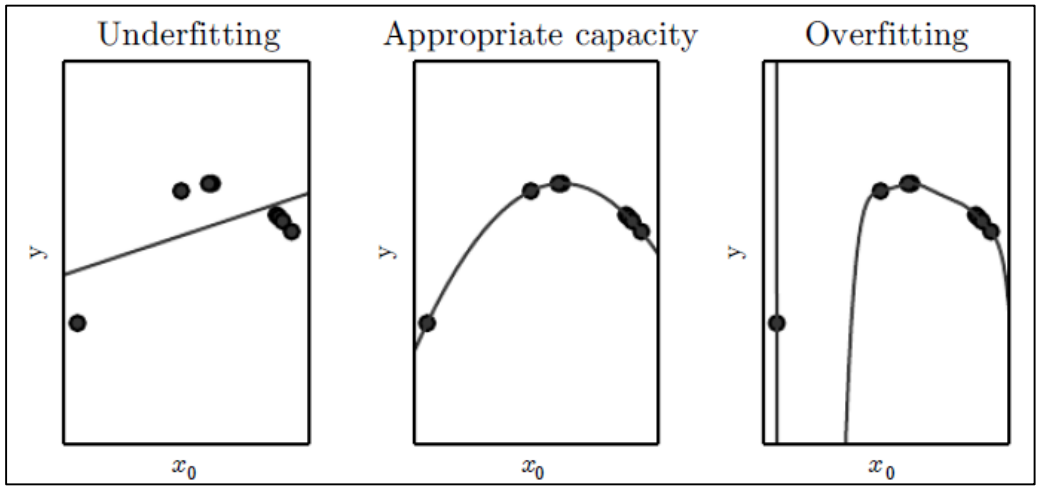
$R(w_1) = 1$

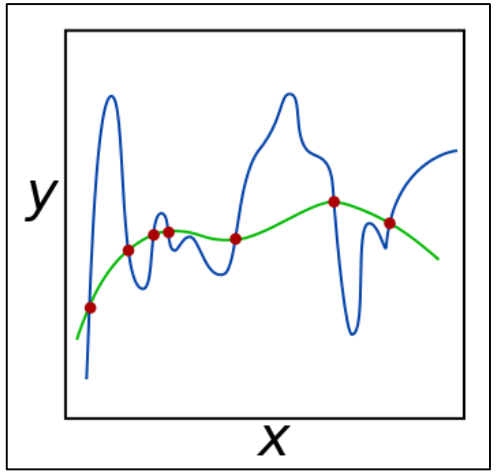
$R(w_2) = 1/4$

$$w_1^T x = w_2^T x = 1$$

Loss Function

- **Regularization**





Loss Function

- Final Loss Function

Softmax

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

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