



FACULTY OF MATHEMATICS,  
PHYSICS AND INFORMATICS  
Comenius University  
Bratislava

Neural Networks for Computer Vision

# Lecture 2: Image Classification

Ing. Viktor Kocur, PhD., RNDr. Zuzana Černeková, PhD.

20.9.2022

# Acknowledgment



The majority of slides are directly adopted from slides for CS231n<sup>1</sup> course at Stanford University!

<sup>1</sup>Stanford CS231n lecture slides. <http://cs231n.stanford.edu/slides/>

# Contents



- Image Classification Task
- Challenges
- kNN Classifier
- Hyperparameters
- Linear Classifier

# Image Classification

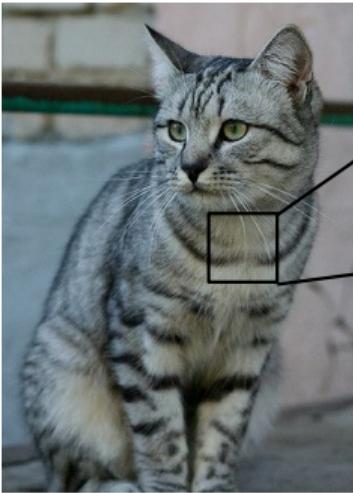


# Image Classification



⇒ Cat

# The Problem: Semantic Gap



[ [115 112 188 111 104 99 186 99 96 103 112 119 104 97 93 87]
[ 91 98 102 106 104 79 98 183 99 105 123 136 119 185 94 85]
[ 76 85 90 105 128 87 96 95 99 115 112 106 183 99 85]
[ 99 81 81 93 128 131 127 108 95 98 182 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 82]
[128 137 144 144 109 95 86 78 62 65 63 63 60 73 86 101]
[125 133 148 137 119 121 117 94 65 79 88 65 54 64 72 98]
[127 125 131 147 133 127 126 131 111 96 89 75 61 64 72 84]
[115 114 109 127 150 148 131 118 113 109 108 92 74 65 72 78]
[ 89 93 99 97 108 147 131 118 113 114 113 102 106 95 77 80]
[ 63 77 86 81 77 79 102 123 117 115 117 125 125 138 115 87]
[ 62 65 82 88 78 71 88 101 124 126 119 101 107 114 131 111]
[ 65 75 88 89 71 82 81 128 138 135 126 107 81 98 101 118]
[ 87 96 107 106 95 69 45 76 138 126 107 92 89 95 105 112]
[118 97 82 86 117 123 116 66 41 51 45 93 89 95 102 107]
[164 146 112 86 82 128 124 104 76 48 45 66 88 101 102 100]
[157 170 157 126 93 86 114 132 112 97 69 55 78 82 99 94]
[130 128 134 161 139 108 109 118 121 134 114 93 87 81 72 79]
[128 112 96 117 154 144 120 115 104 107 102 93 93 87 80 107 112 99]
[123 107 96 86 83 112 153 149 122 109 104 75 80 107 112 99]
[122 121 102 80 82 86 94 117 145 148 153 102 58 78 92 107]
[122 164 148 103 71 56 78 83 93 103 119 139 102 61 69 84]]

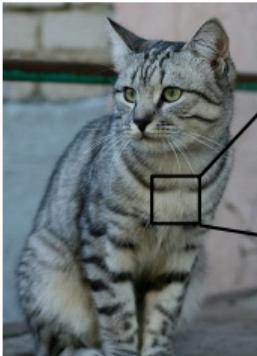
What the computer sees

An image is a tensor of integers between [0, 255]:

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

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

## Challenges: Viewpoint variation

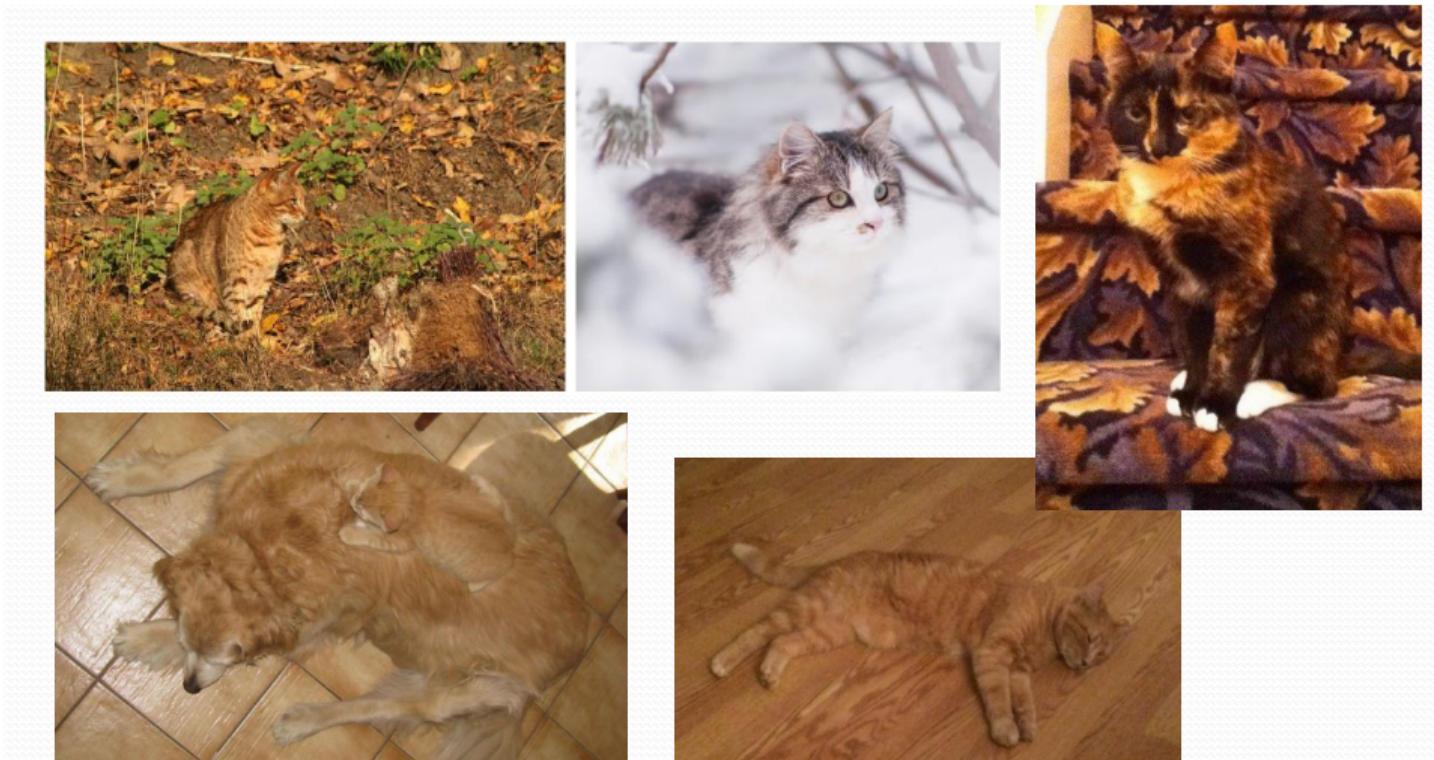


1195 112 180 111 184 30 186 90 90 183 123 136 184 93 93 181
1 91 68 182 186 184 79 56 183 99 183 123 136 118 186 64 64 181
1 76 85 94 185 128 180 87 96 95 99 115 152 186 183 99 99 181
1 69 92 123 123 123 123 123 123 123 123 123 123 123 123 123 123 181
1 88 91 81 84 89 91 88 85 181 187 189 98 98 75 84 96 95 1
1 114 189 93 95 55 85 64 54 64 87 152 129 88 74 84 91 1
1 123 147 180 180 180 180 180 180 180 180 180 180 180 180 180 181
1 128 137 141 148 189 95 86 78 62 93 83 63 63 73 86 181 1
1 125 133 148 137 119 123 117 94 65 79 98 85 54 64 72 90 1
1 127 138 148 148 148 148 148 148 148 148 148 148 148 148 148 148 1
1 115 114 189 123 158 146 131 118 113 113 114 114 114 186 186 186 181
1 49 93 94 97 188 147 111 118 113 114 114 114 186 186 186 181
1 62 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 115 181
1 63 65 82 89 78 71 88 181 124 124 119 181 187 187 114 131 159 1
1 83 65 85 85 85 85 85 85 85 85 85 85 85 85 85 85 85 118 118 1
1 49 85 75 87 88 71 82 81 128 128 128 128 128 128 128 128 128 128 1
1 97 97 82 88 117 123 116 66 41 31 95 93 89 95 182 187 1
1 106 106 106 106 106 106 106 106 106 106 106 106 106 106 106 106 106 181 181
1 157 178 157 128 128 114 123 112 47 47 55 55 55 55 55 55 55 55 181 181
1 138 128 134 141 139 189 189 118 121 134 114 87 87 65 53 69 98 1
1 128 112 98 113 113 113 113 113 113 113 113 113 113 113 113 113 113 181 1
1 133 121 121 121 86 83 112 153 149 122 189 184 75 69 187 112 99 1
1 122 121 182 88 82 86 94 117 145 149 153 182 58 58 78 92 92 187 1
1 122 184 149 143 73 15 78 83 93 183 119 128 182 61 60 64 1

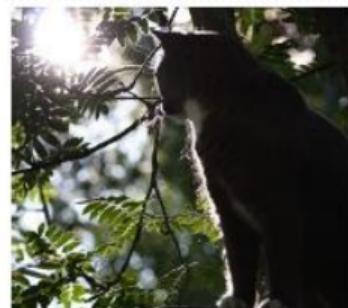


All pixels change when  
the camera moves!

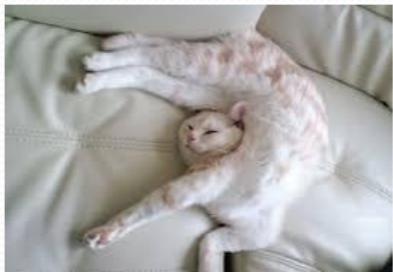
# Challenges - background



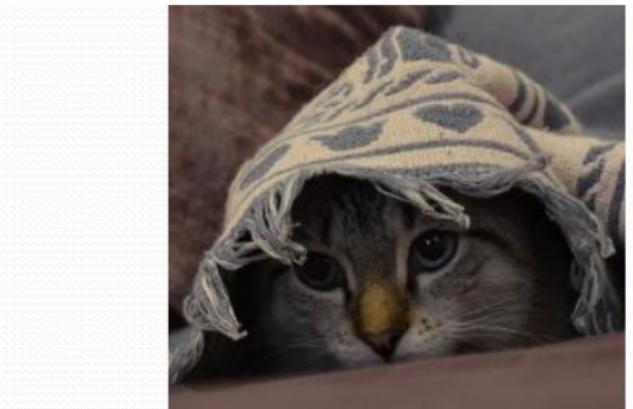
# Challenges - illumination



# Challenges - deformation



# Challenges - occlusion



# Challenges - intraclass variation



# Image Classifier



```
def classify_image(image):
    # some magic
    return class_label
```

# Image Classifier



```
def classify_image(image):  
    # some magic  
    return class_label
```

There is no obvious way how to do this!

# Classical CV



Find edges



Find corners



?

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

# Machine Learning



Machine learning algorithms are algorithms which can improve their performance from experience. In case of CV experience usually means large annotated datasets.

Typical ML workflow:

1. Collect a dataset of images and corresponding labels
2. Train a classifier using ML
3. Evaluate the classifier on a new set of images

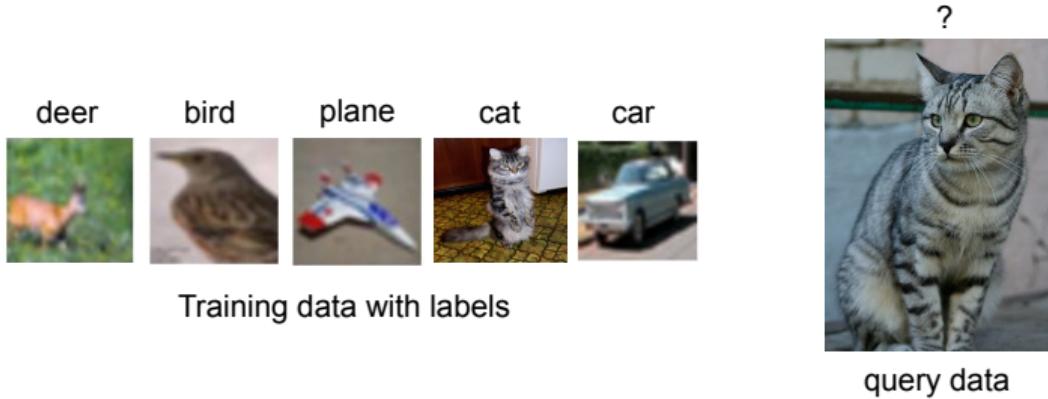
# Nearest Neighbors classifier



One of the simplest classifiers is the nearest neighbor classifier:

- We train the classifier by simply remembering all of the training images and labels.
- To classify a new image we assign it the label of the image from the training set which is closest the new image.

# Distance metric



Distance Metric | , |  $\rightarrow \mathbb{R}$

A diagram showing a vertical bar separating two images of a cat. This visual represents the inputs to a function that takes two images and outputs a real number, which corresponds to the distance between them.

# Simple metric



**L1 distance:**  $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

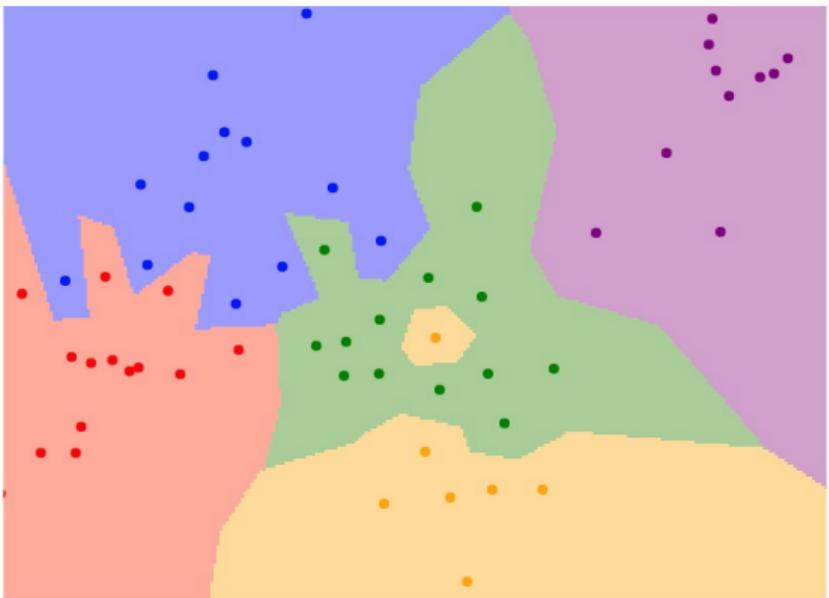
test image				training image				pixel-wise absolute value differences			
56	32	10	18	10	20	24	17	46	12	14	1
90	23	128	133	8	10	89	100	82	13	39	33
24	26	178	200	12	16	178	170	12	10	0	30
2	0	255	220	4	32	233	112	2	32	22	108

-

=

add → 456

# Decision boundary

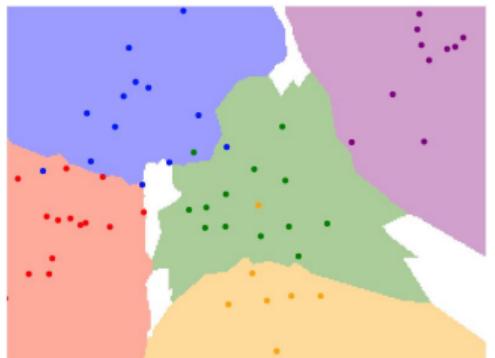
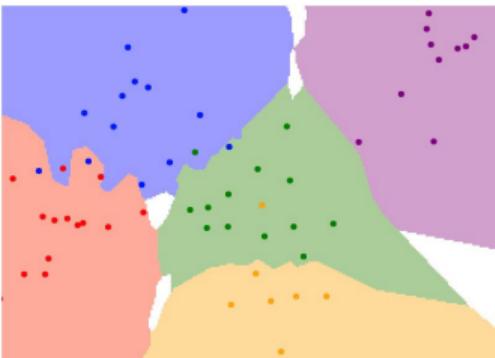


1-nearest neighbor

# k-Nearest Neighbors



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



# Hyperparameters



Before using kNN we have to make few decisions:

- Which  $k$  do we use? E.g. number of neighbors.
- What kind of metric we use to calculate the distances?

# Hyperparameters



Before using kNN we have to make few decisions:

- Which  $k$  do we use? E.g. number of neighbors.
- What kind of metric we use to calculate the distances?

Both of these are **hyperparameters**. In general, hyperparameters change the algorithms themselves. We use the prefix hyper, since more sophisticated usually have parameters which are learned during training.

Setting these is difficult and dependent on the given problem! Usually you have to try and see what works best!

# Setting Hyperparameters



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

train

# Setting Hyperparameters



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

**BAD:**  $K = 1$  always works perfectly on training data

train

# Setting Hyperparameters



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

**BAD:** K = 1 always works perfectly on training data

train

**Idea #2:** choose hyperparameters that work best on **test data**

test

# Setting Hyperparameters



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

**BAD:** K = 1 always works perfectly on training data

train

**Idea #2:** choose hyperparameters that work best on **test data**

**BAD:** No idea how algorithm will perform on new data

train

test

Never do this!

# Setting Hyperparameters



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

**BAD:** K = 1 always works perfectly on training data

train

**Idea #2:** 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**; choose hyperparameters on val and evaluate on test

**Better!**

train

validation

test

# Setting Hyperparameters



train

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

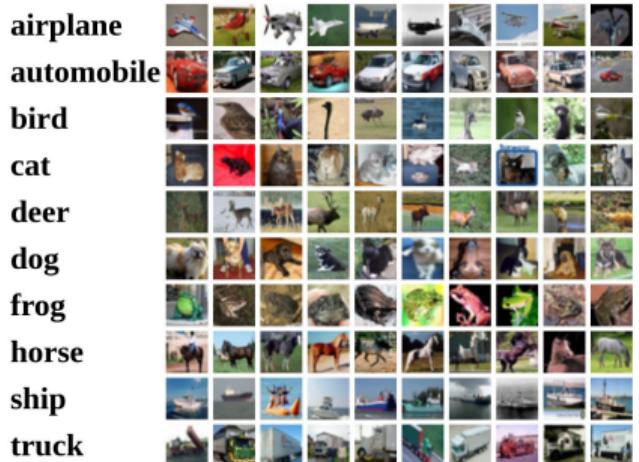
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

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

# Example Dataset - CIFAR10



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

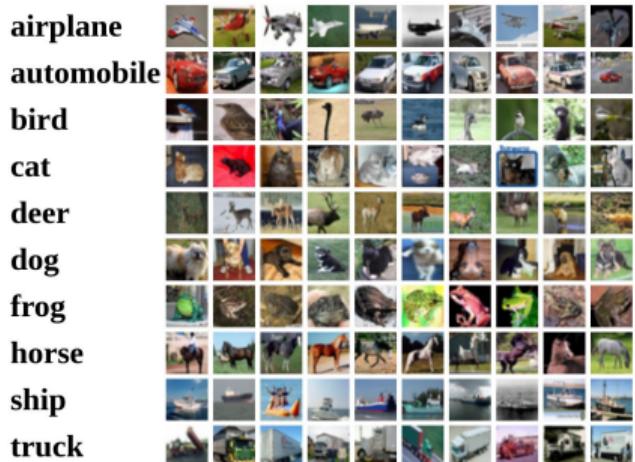


Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.



# 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.

# kNN - not a good choice



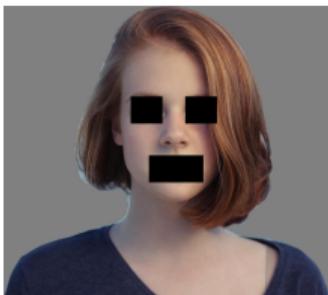
k-Nearest Neighbor with pixel distance **never used**.

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

Original



Occluded



Shifted (1 pixel)



Tinted



Original image is  
CC0 public domain

# kNN - not a good choice



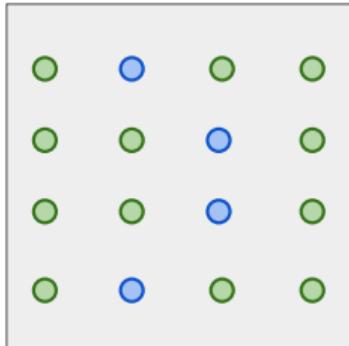
k-Nearest Neighbor with pixel distance **never used**.

- Curse of dimensionality

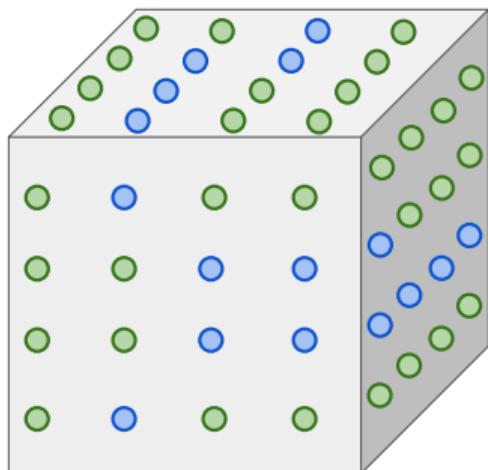
Dimensions = 1  
Points = 4



Dimensions = 2  
Points =  $4^2$



Dimensions = 3  
Points =  $4^3$



# Linear Classifier



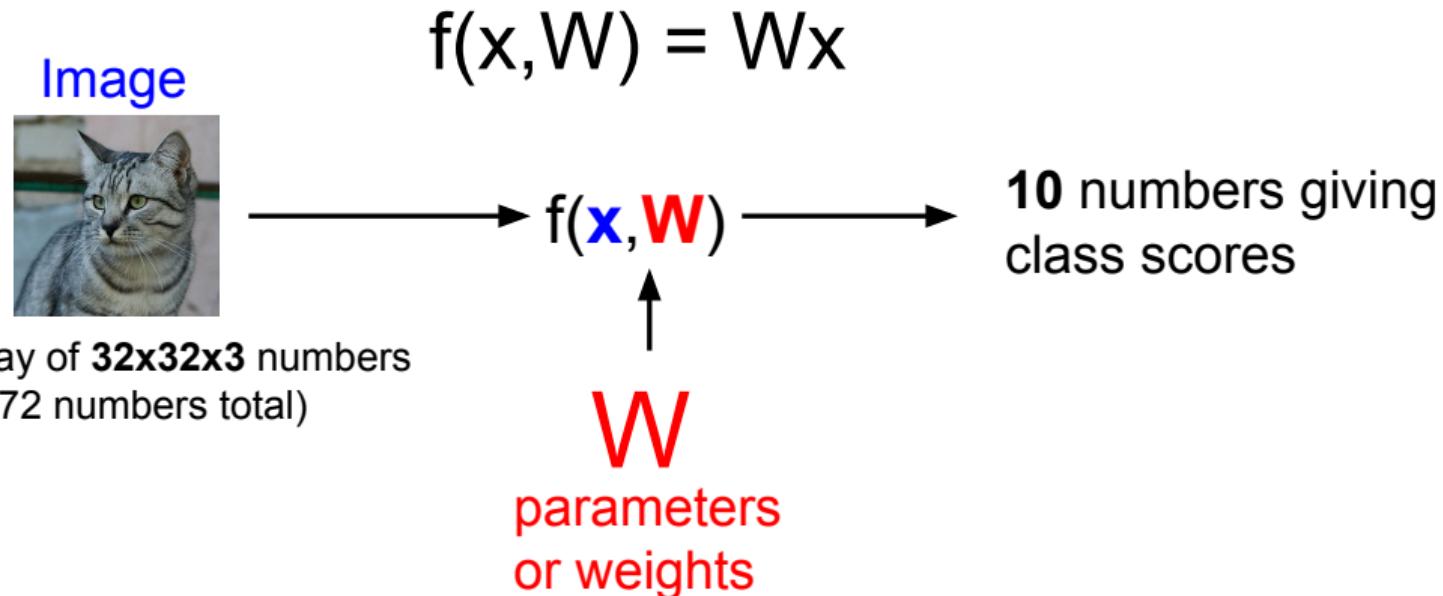
Array of **32x32x3** numbers  
(3072 numbers total)

$$f(\mathbf{x}, \mathbf{W})$$

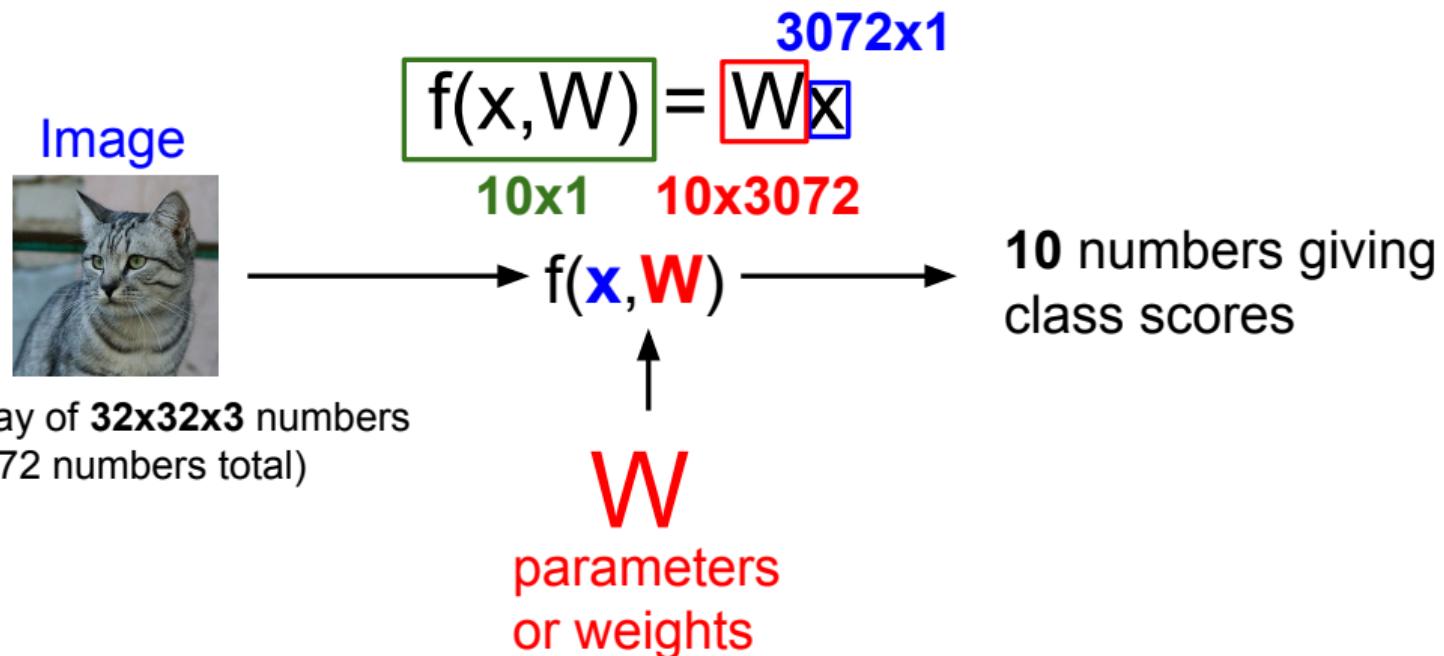
**W**  
parameters  
or weights

10 numbers giving  
class scores

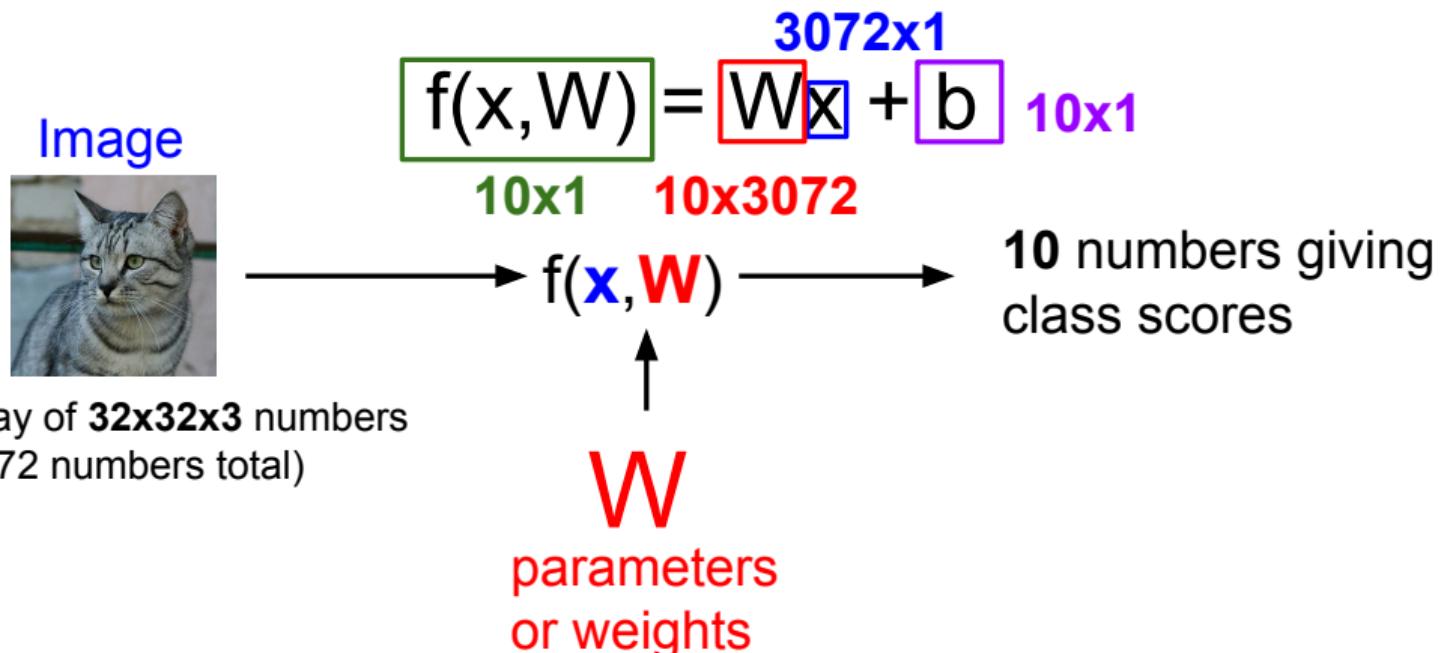
# Linear Classifier



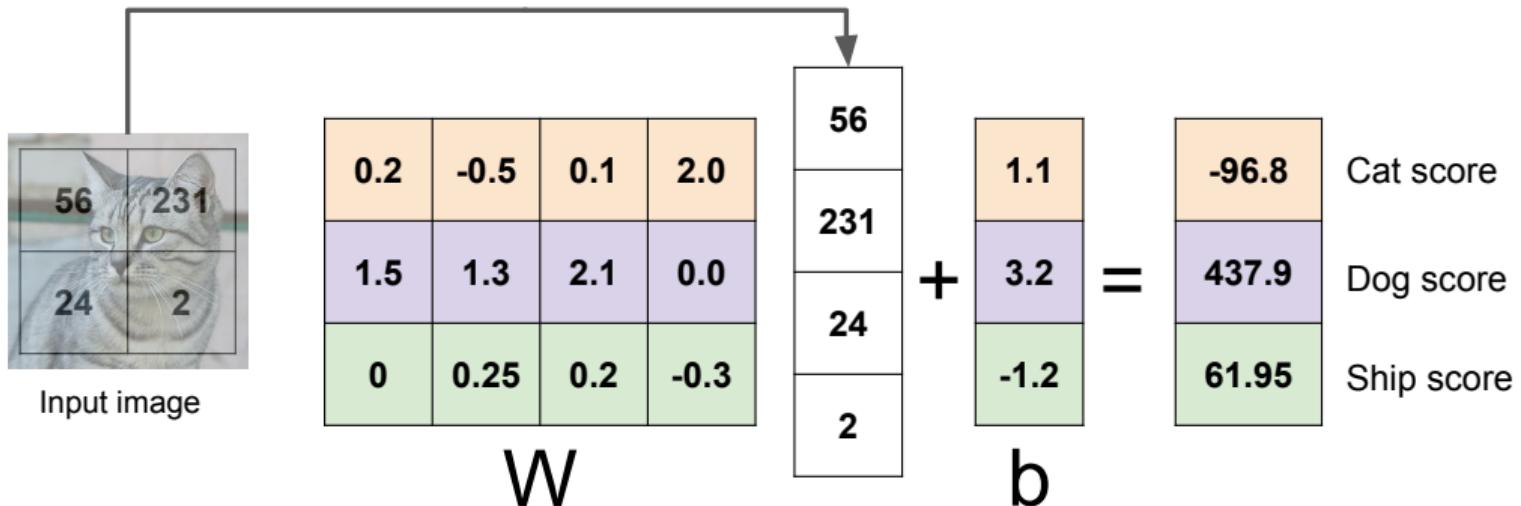
# Linear Classifier



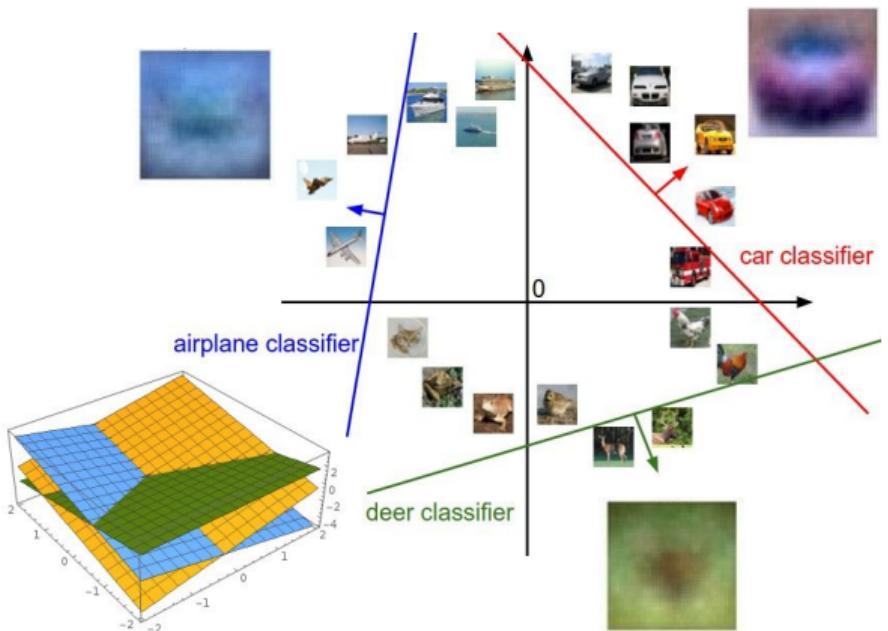
# Linear Classifier



# Linear Classifier - Example



# Linear Classifier - Decision Boundary



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



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

# Linear Classifier - Hard cases

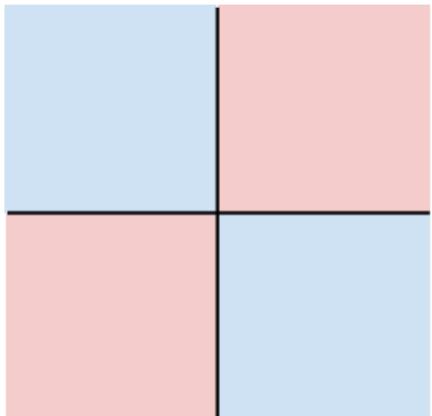


## Class 1:

First and third quadrants

## Class 2:

Second and fourth quadrants

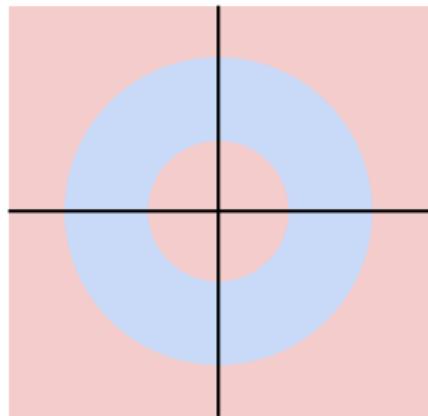


## Class 1:

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

## Class 2:

Everything else

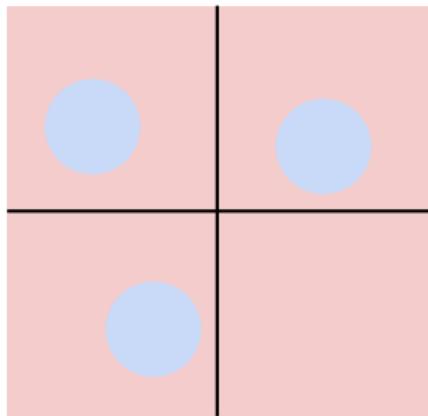


## Class 1:

Three modes

## Class 2:

Everything else





$$f(\mathbf{x}, W) = W\mathbf{x} + \mathbf{b}$$

- Loss function
  - ▶ Quantifying *good* parameters
- Optimization
  - ▶ Finding optimal parameters to minimize the loss function
- Convolutional Neural Networks