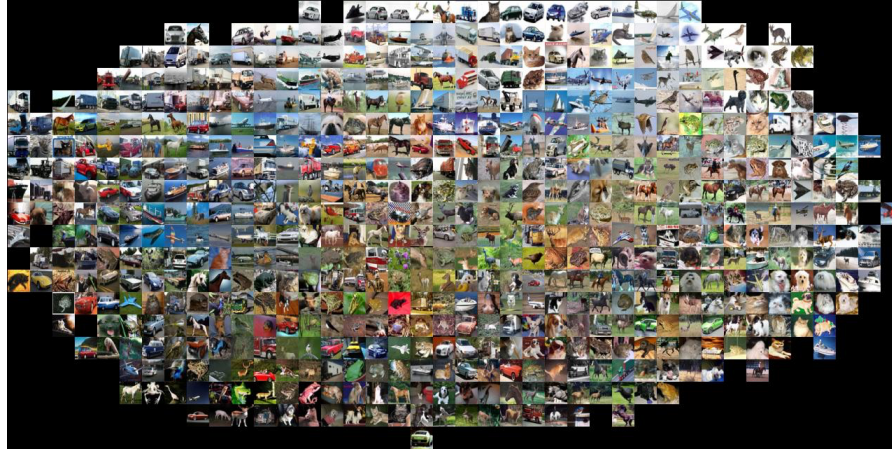


CS4670/5670: Computer Vision

Kavita Bala

Lecture 33: Recognition Basics



Slides from Andrej Karpathy and Fei-Fei Li
<http://vision.stanford.edu/teaching/cs231n/>

Announcements

- PA 3 Artifact voting

Today

- Image classification pipeline
- Training, validation, testing
- Score function and loss function

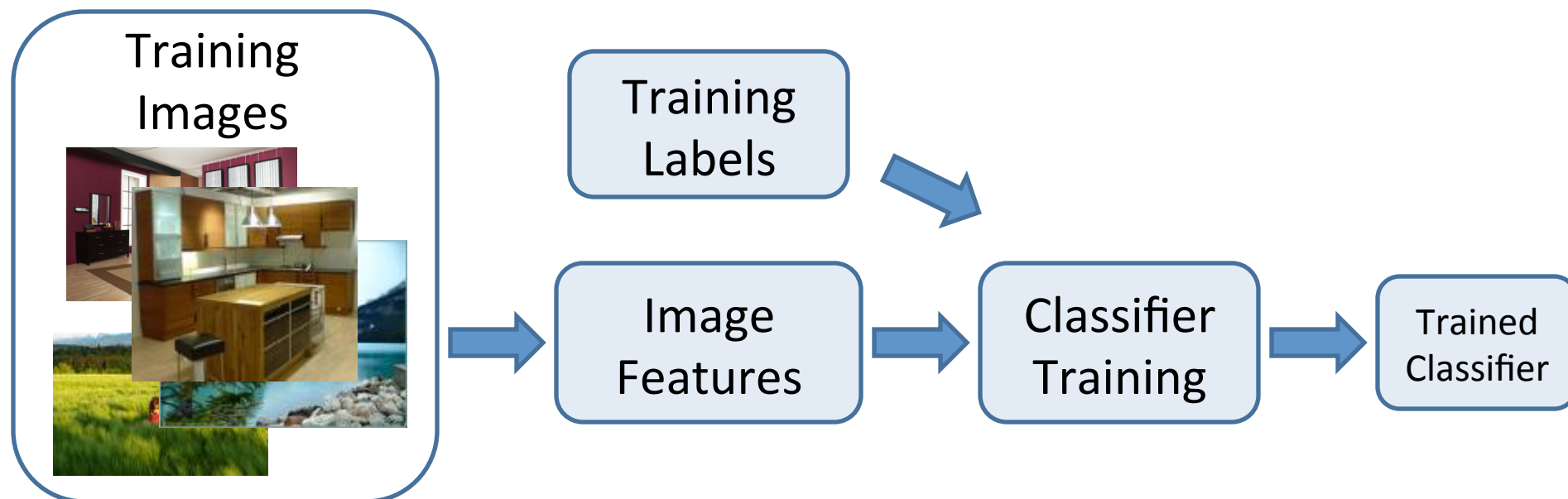
- Building up to CNNs for learning
 - 5-6 lectures on deep learning

Where to from here?

- Scene Understanding
 - Big data – lots of images
 - Crowd sourcing – lots of people
 - Deep Learning – lots of compute

Image categorization

Training



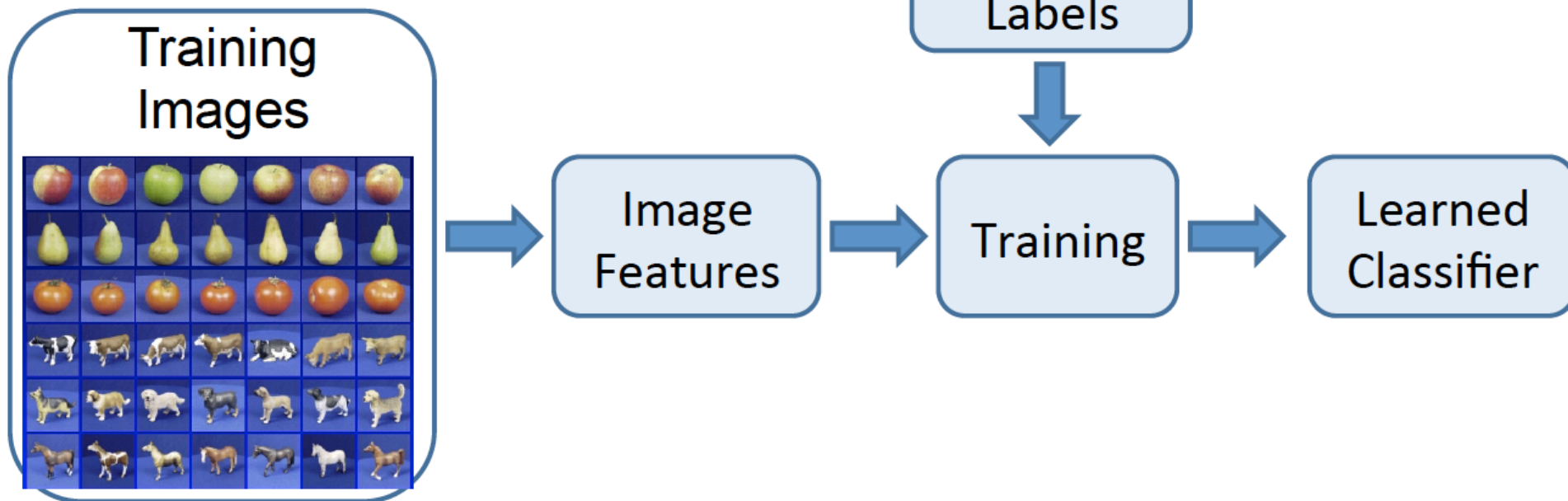
Categorization

$f(\text{apple image}) = \text{“apple”}$

$f(\text{tomato image}) = \text{“tomato”}$

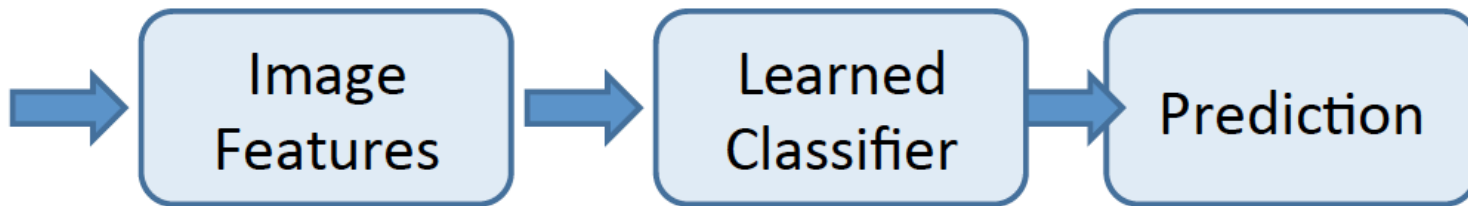
$f(\text{cow image}) = \text{“cow”}$

Training



Testing

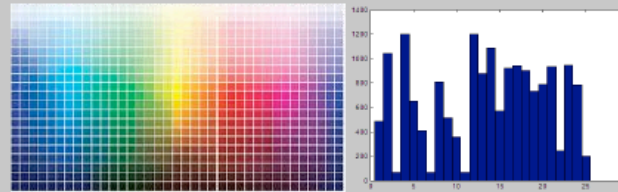
Test Image



Input image



Color: Quantize RGB values



Invariance?

- 😊 Translation
- 😊 Scale
- 😊 Rotation
- 😞 Occlusion

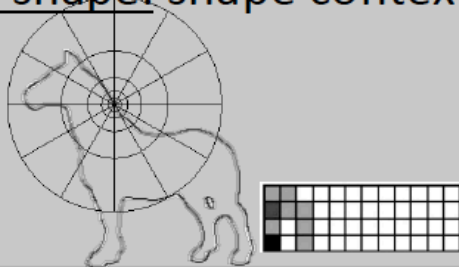
Global shape: PCA space



Invariance?

- 😊 Translation
- ? Scale
- 😊 Rotation
- 😞 Occlusion

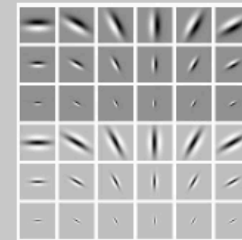
Local shape: shape context



Invariance?

- 😊 Translation
- 😊 Scale
- ? Rotation (in-planar)
- 😞 Occlusion

Texture: Filter banks



Invariance?

- 😊 Translation
- ? Scale
- ? Rotation (in-planar)
- 😞 Occlusion

Results



	Color	$D_x D_y$	Mag-Lap	PCA Masks	PCA Gray	Cont. Greedy	Cont. DynProg	Avg.
apple	57.56%	85.37%	80.24%	78.78%	88.29%	77.07%	76.34%	77.66%
pear	66.10%	90.00%	85.37%	99.51%	99.76%	90.73%	91.71%	89.03%
tomato	98.54%	94.63%	97.07%	67.80%	76.59%	70.73%	70.24%	82.23%
cow	86.59%	82.68%	94.39%	75.12%	62.44%	86.83%	86.34%	82.06%
dog	34.63%	62.44%	74.39%	72.20%	66.34%	81.95%	82.93%	67.84%
horse	32.68%	58.78%	70.98%	77.80%	77.32%	84.63%	84.63%	69.55%
cup	79.76%	66.10%	77.80%	96.10%	96.10%	99.76%	99.02%	87.81%
car	62.93%	98.29%	77.56%	100.0%	97.07%	99.51%	100.0%	90.77%
total	64.85%	79.79%	82.23%	83.41%	82.99%	86.40%	86.40%	80.87%

Crowdsourcing

Unlabeled
Images



Show images,
Collect and
filter labels

Training
Images



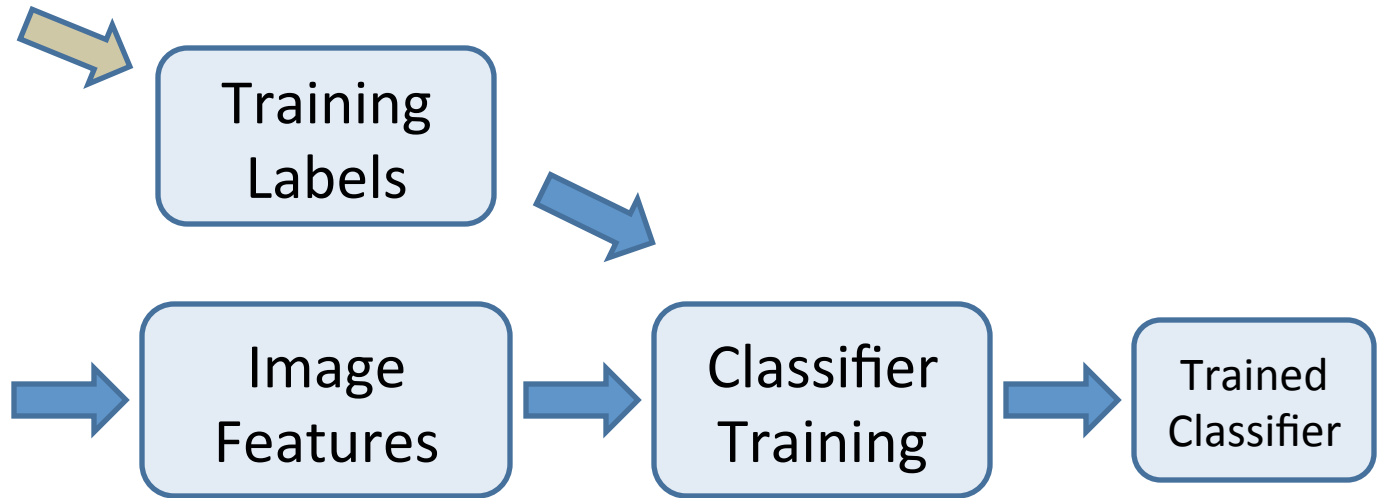
Training
Labels

Image
Features

Classifier
Training

Trained
Classifier

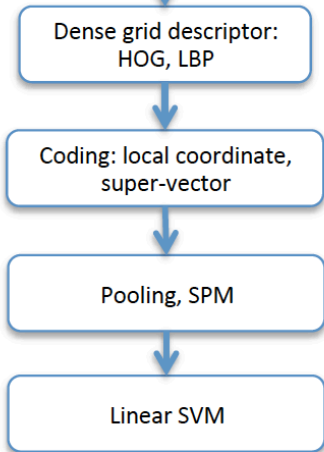
Training



IMAGENET Large Scale Visual Recognition Challenge

Year 2010

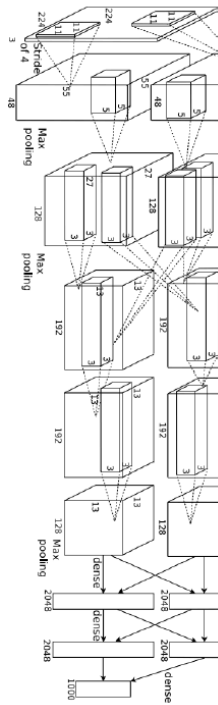
NEC-UIUC



[Lin CVPR 2011]

Year 2012

SuperVision



[Krizhevsky NIPS 2012]

Year 2014

GoogLeNet



Convolution
Pooling
Softmax
Other

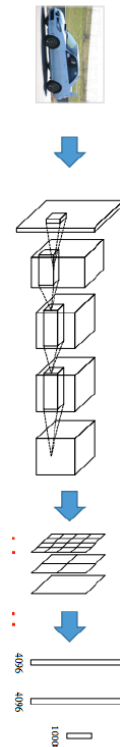
[Szegedy arxiv 2014]

VGG



[Simonyan arxiv 2014]

MSRA



[He arxiv 2014]

Deep Learning or CNNs

- Since 2012, huge impact..., best results
- Can soak up all the data for better prediction

Image Classification

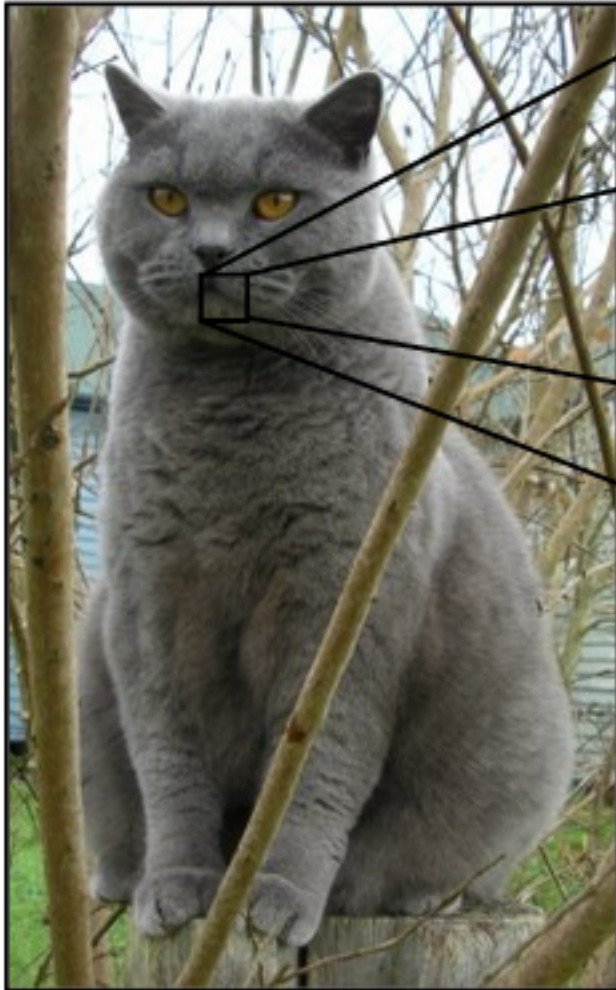


(assume given set of discrete labels)
{dog, cat, truck, plane, ...}



cat

Image Classification: Problem



08	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	81	88
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	54	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	55	85	30	03	49	13	36	65
52	70	95	23	04	60	11	42	69	21	65	56	01	32	56	71	37	02	36	91
22	31	16	71	51	67	05	89	41	92	36	54	22	40	40	28	66	33	13	80
24	47	37	80	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
55	36	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	38	35	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	82	89	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	48	84	81	16	23	57	05	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	27	67	48

What the computer sees

image classification → 82% cat
15% dog
2% hat
1% mug

Data-driven approach

- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images

Example training set



Data-driven approach

- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images

```
def train(train_images, train_labels):  
    # build a model of images -> labels  
  
def predict(image):  
    # evaluate the model on the image  
    return class_label
```


Train and Test

- Split dataset between training images and test images
- Be careful about inflation of results

Classifiers

- Nearest Neighbor
- kNN
- SVM
- ...

First: Nearest Neighbor (NN) Classifier

- Train
 - Remember all training images and their labels
- Predict
 - Find the closest (most similar) training image
 - Predict its label as the true label

How to find the most similar training image? What is the distance metric?

L1 distance:

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

Where I_1 denotes image 1, and p denotes each pixel

test image		training image		pixel-wise absolute value differences				
56	32	10	20	46	12	14	1	→ 456
90	23	8	10	82	13	39	33	
24	26	12	16	12	10	0	30	
2	0	4	32	2	32	22	108	

Choice of distance metric

- Hyperparameter

L1 (Manhattan) distance

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

L2 (Euclidean) distance

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

- Two most commonly used special cases of p-norm

$$\|x\|_p = (|x_1|^p + \dots + |x_n|^p)^{\frac{1}{p}} \quad p \geq 1, x \in \mathbb{R}^n$$

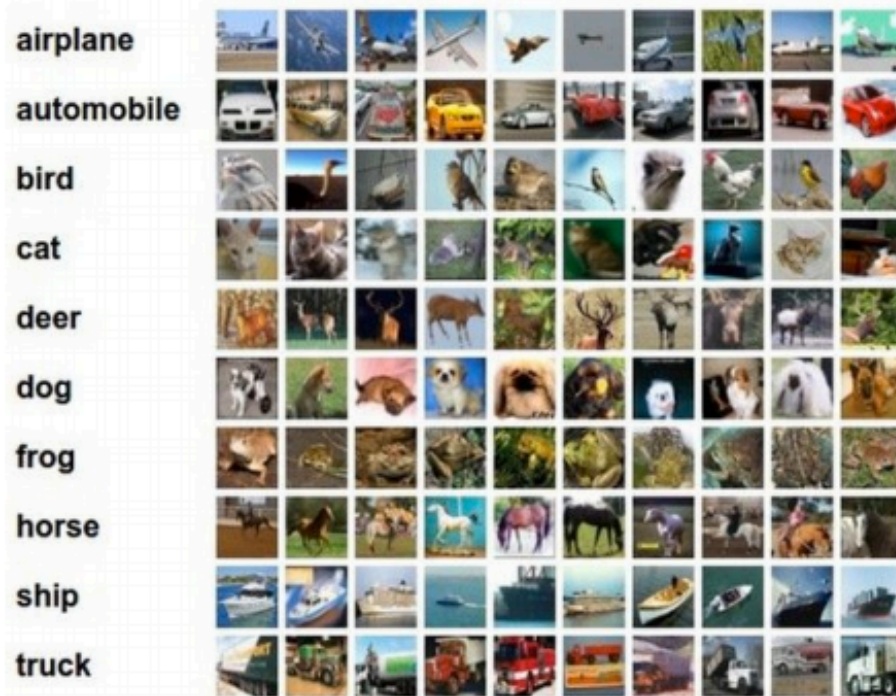
CIFAR-10 and NN results

Example dataset: **CIFAR-10**

10 labels

50,000 training images, each image is tiny: 32x32

10,000 test images.



CIFAR-10 and NN results

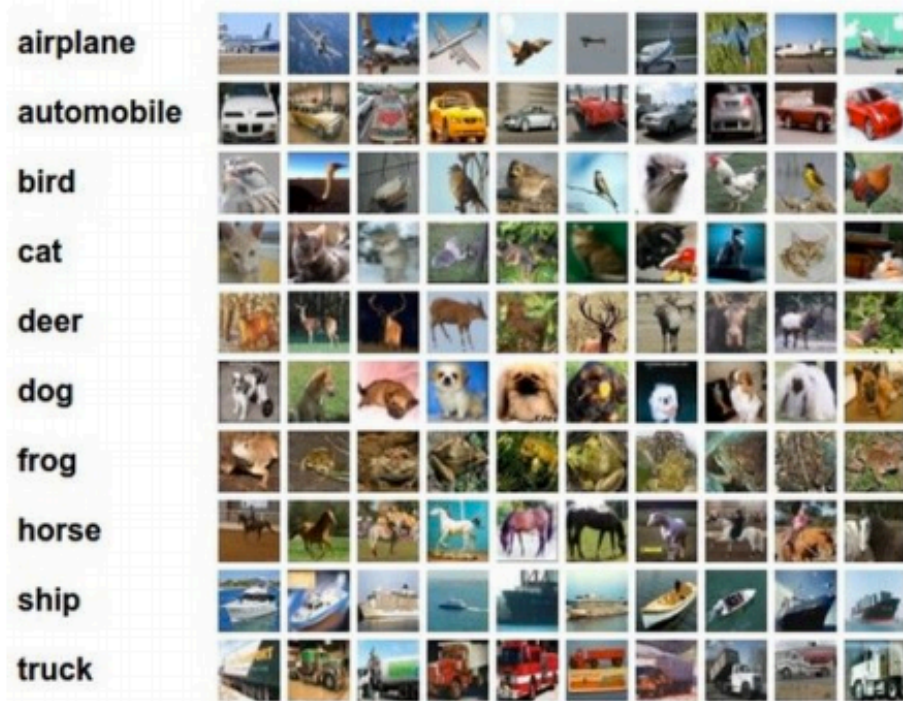
Example dataset: **CIFAR-10**

10 labels

50,000 training images

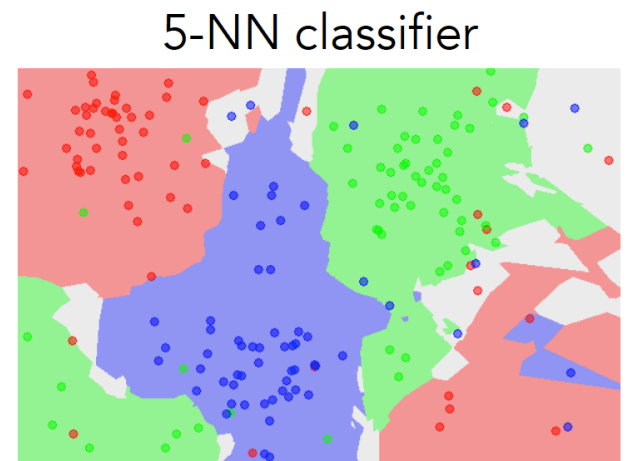
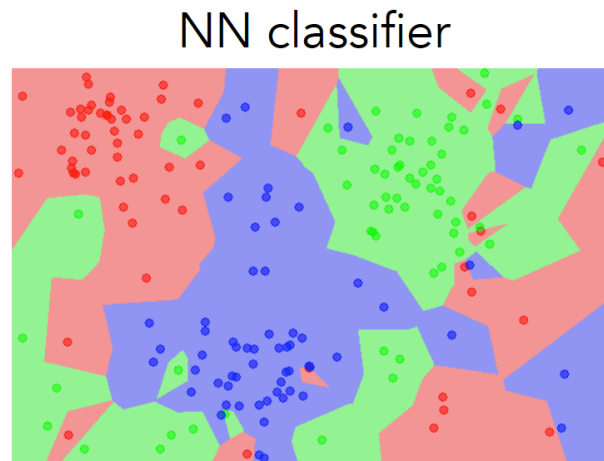
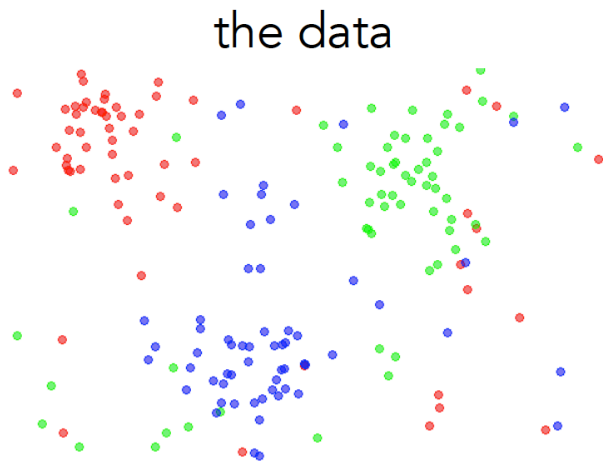
10,000 test images.

For every test image (first column),
examples of nearest neighbors in rows



k-nearest neighbor

- Find the k closest points from training data
- Labels of the k points “vote” to classify



Hyperparameters

The choice of distance is a **hyperparameter**
common choices:

L1 (Manhattan) distance

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

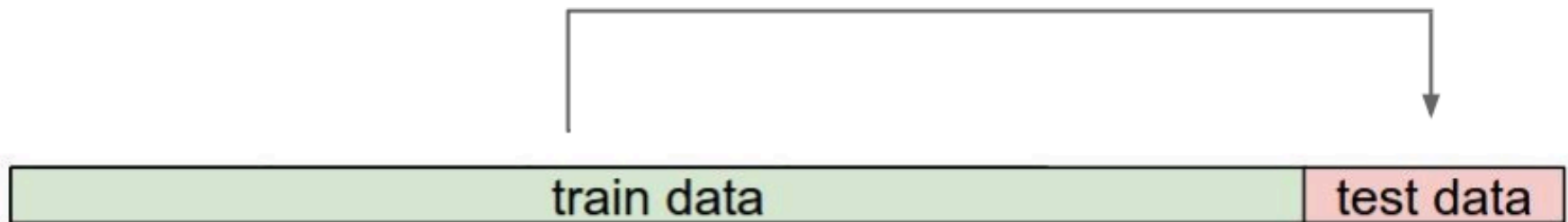
L2 (Euclidean) distance

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

Hyperparameters

- What is the best distance to use?
- What is the best value of k to use?
- i.e., how do we set the hyperparameters?
- Very problem-dependent
- Must try them all and see what works best

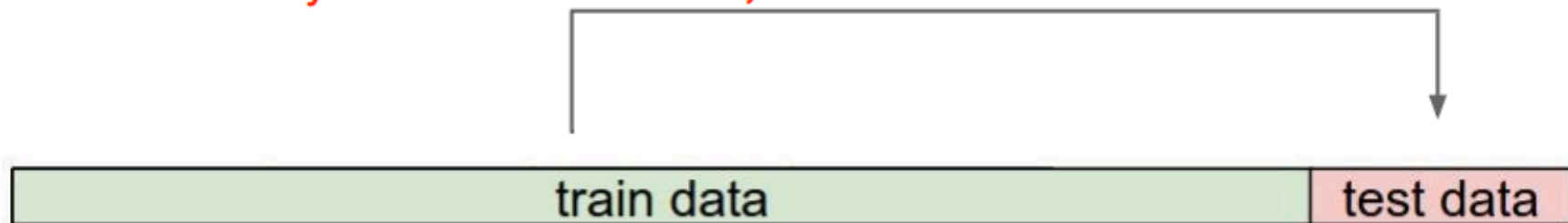
Try out what hyperparameters work best on test set.



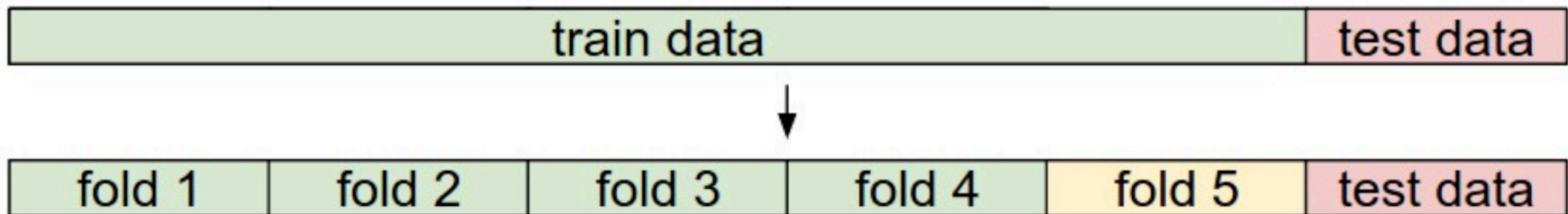
Trying out what hyperparameters work best on test set:

Very bad idea. The test set is a proxy for the generalization performance!

Use only **VERY SPARINGLY**, at the end.



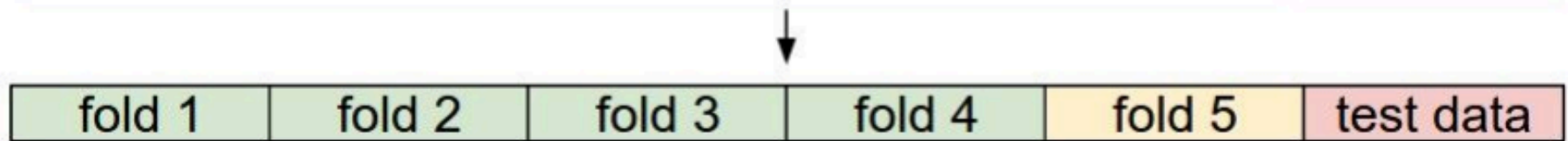
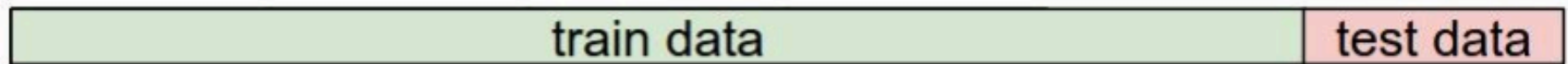
Validation



Validation data

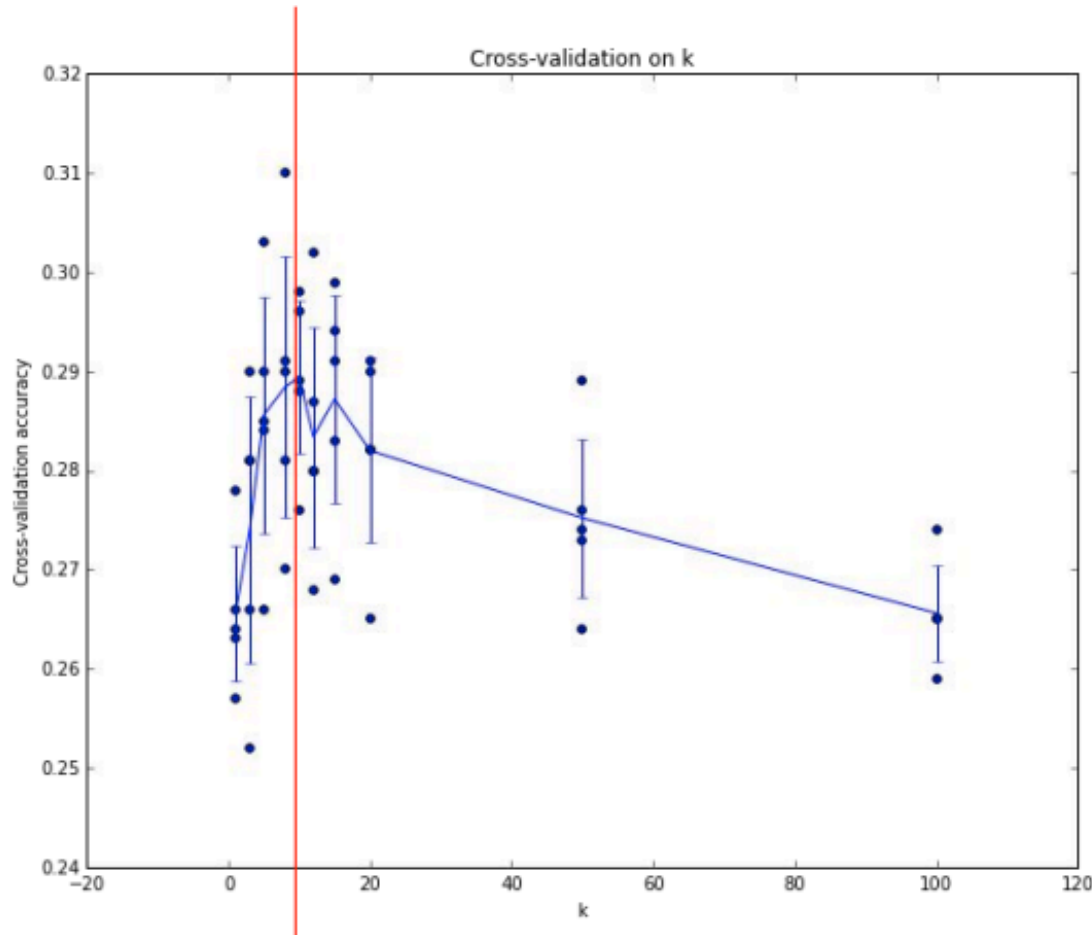
use to tune hyperparameters
evaluate on test set ONCE at the end

Cross-validation



Cross-validation

cycle through the choice of which fold is the validation fold, average results.



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)

How to pick hyperparameters?

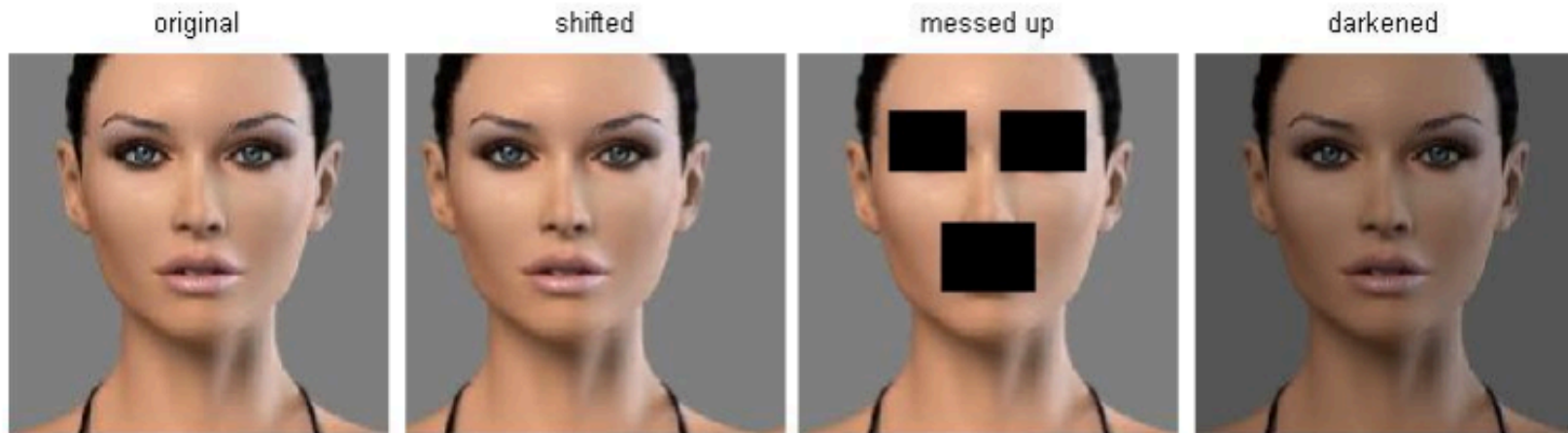
- Methodology
 - Train and test
 - Train, validate, test
- Train for original model
- Validate to find hyperparameters
- Test to understand generalizability

Complexity and Storage

- N training images, M test images
- Training: $O(1)$
- Testing: $O(MN)$
- Hmm...
 - Normally need the opposite
 - Slow training (ok), fast testing (necessary)

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

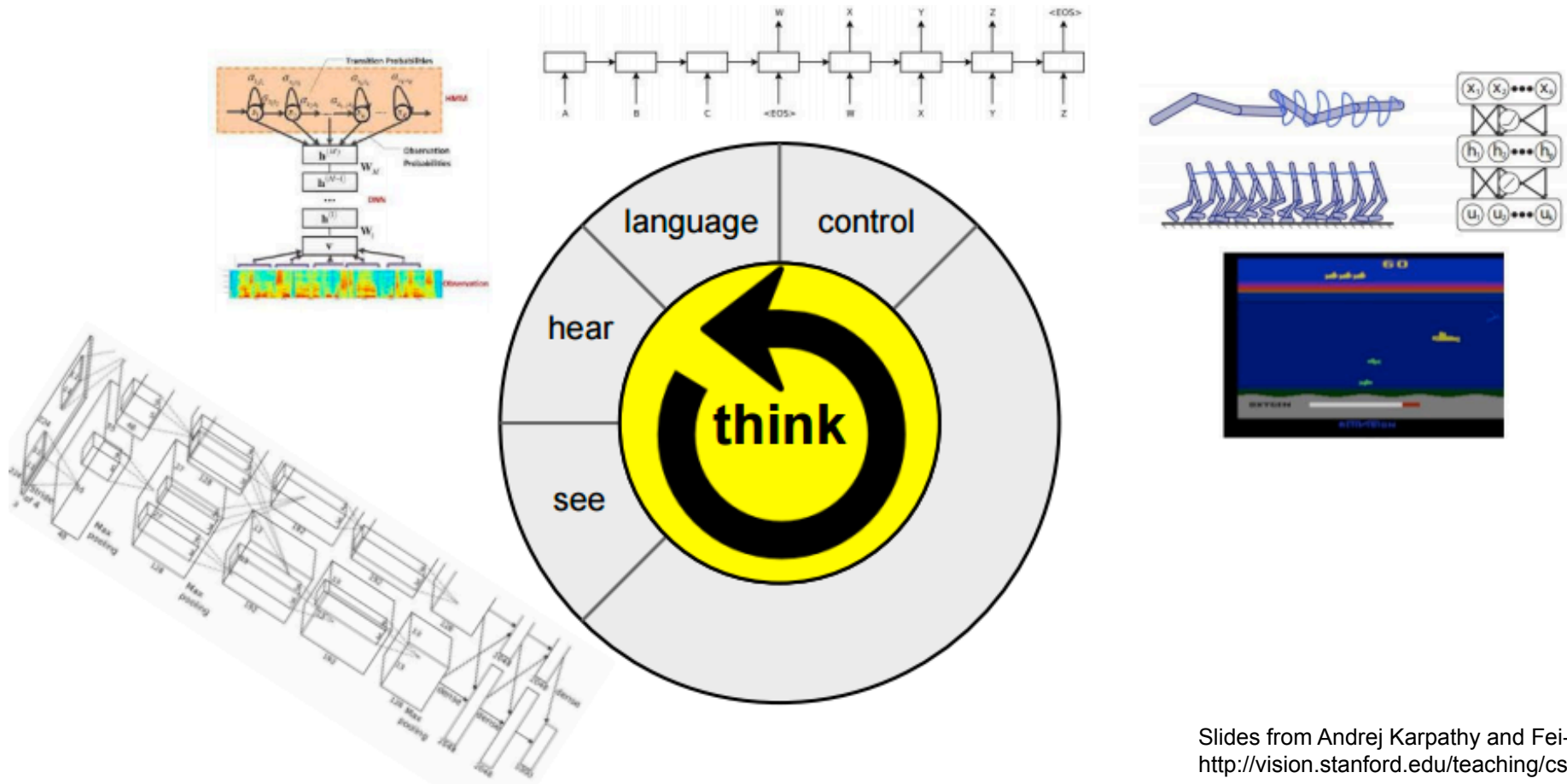
- terrible performance at test time
- distance metrics on level of whole images can be very unintuitive



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

Instead

- Image classification using linear classifiers



Instead

- Image classification using linear classifiers
- Need
 - Score function: raw data to class scores
 - Loss function: agreement between predicted scores and ground truth labels

Score function



class scores

Score function: f

Parametric approach



image parameters

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

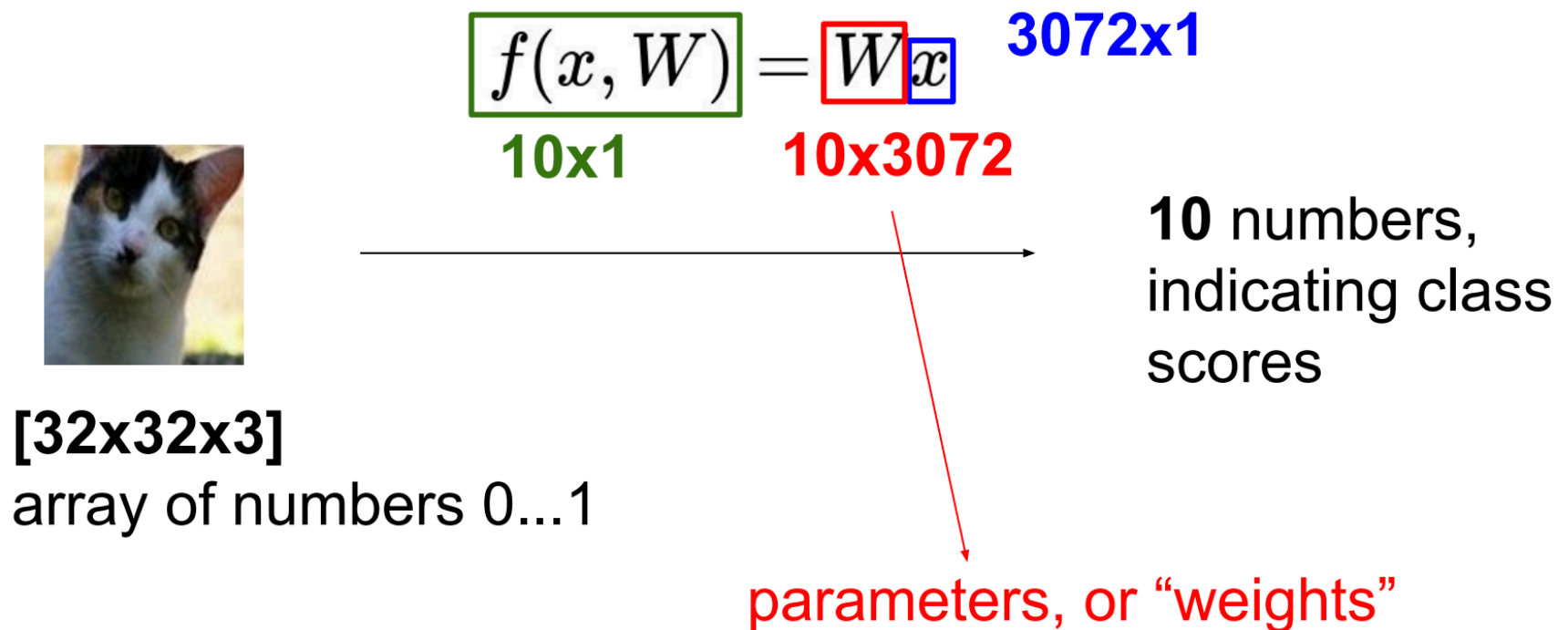


10 numbers,
indicating class
scores

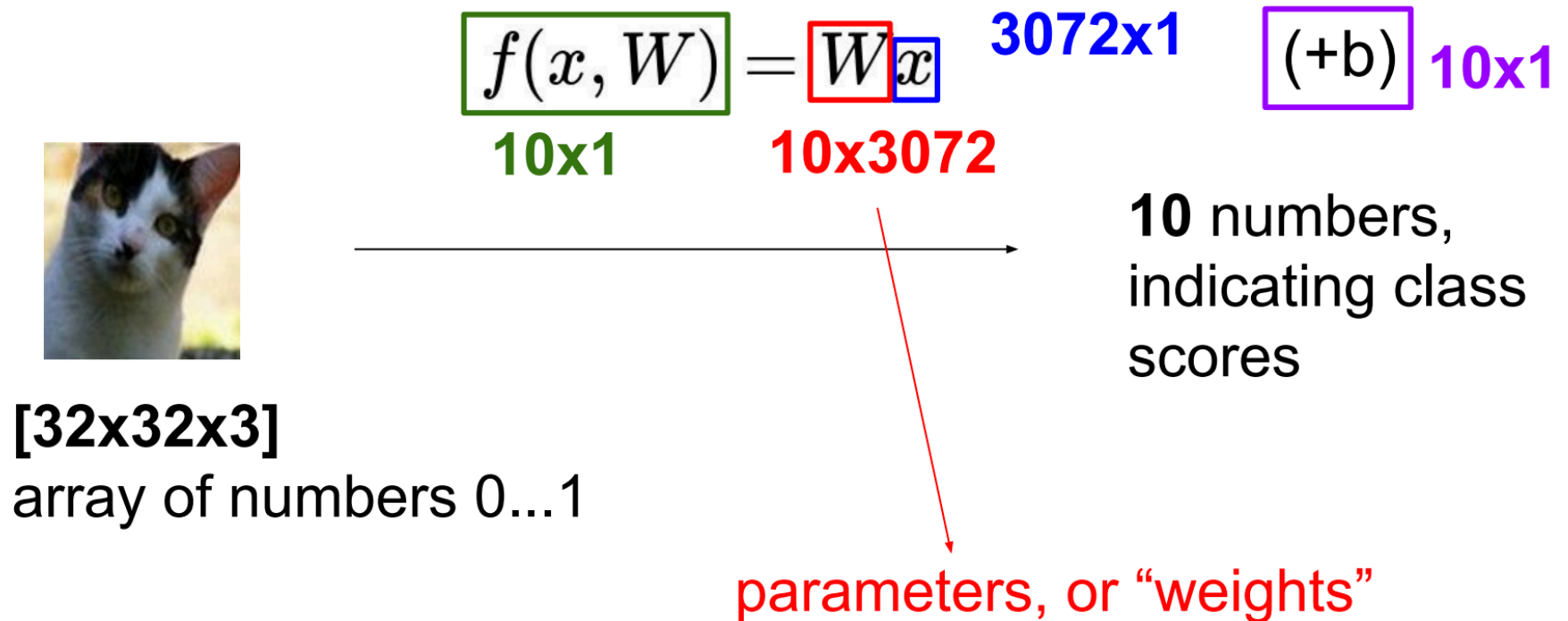
[32x32x3]

array of numbers 0...1
(3072 numbers total)

Parametric approach: Linear classifier

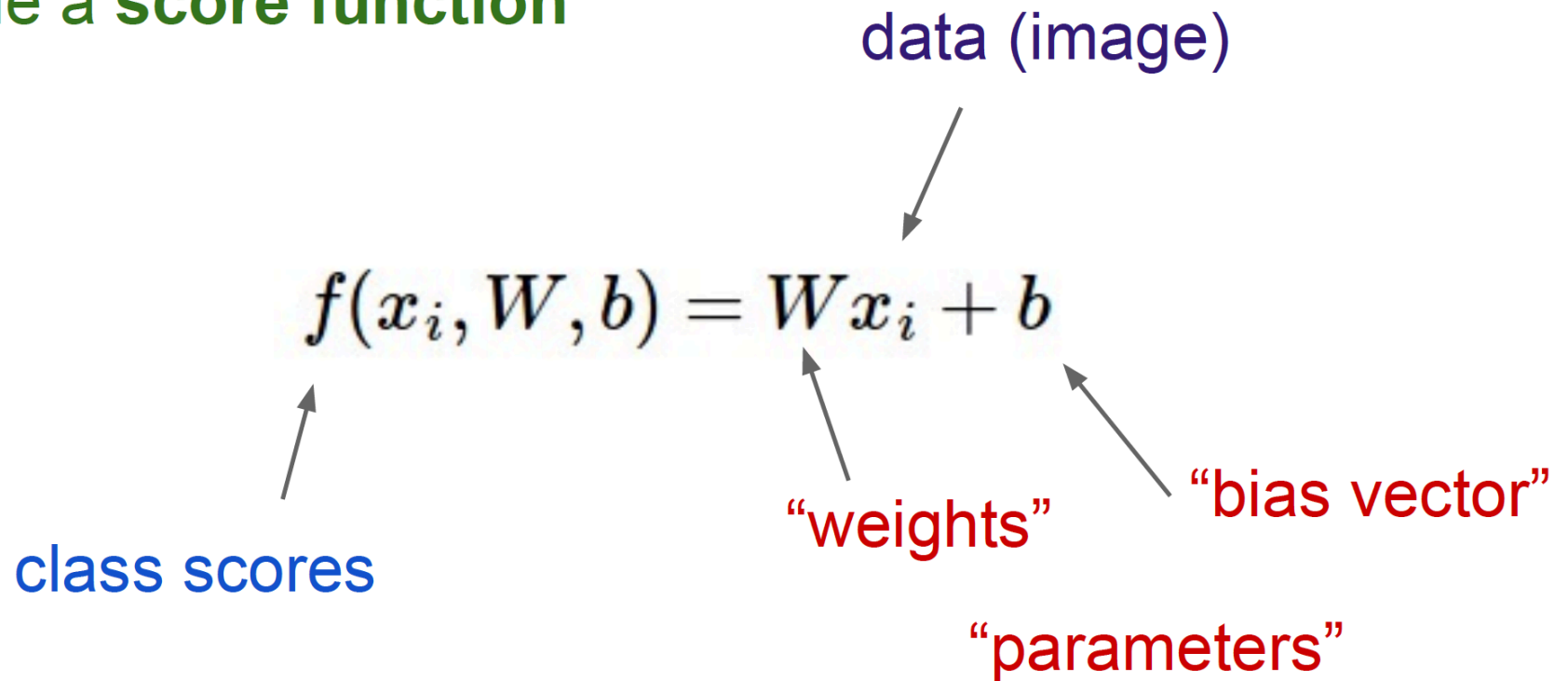


Parametric approach: Linear classifier

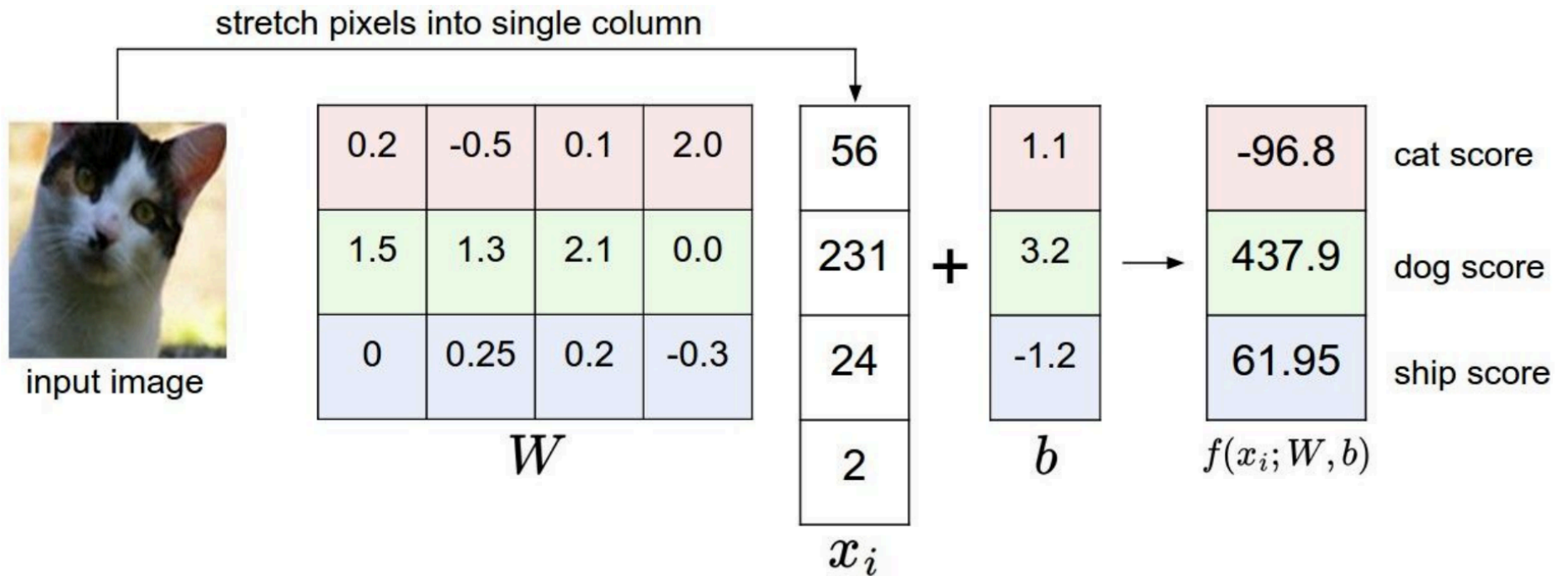


Linear Classifier

define a **score function**



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

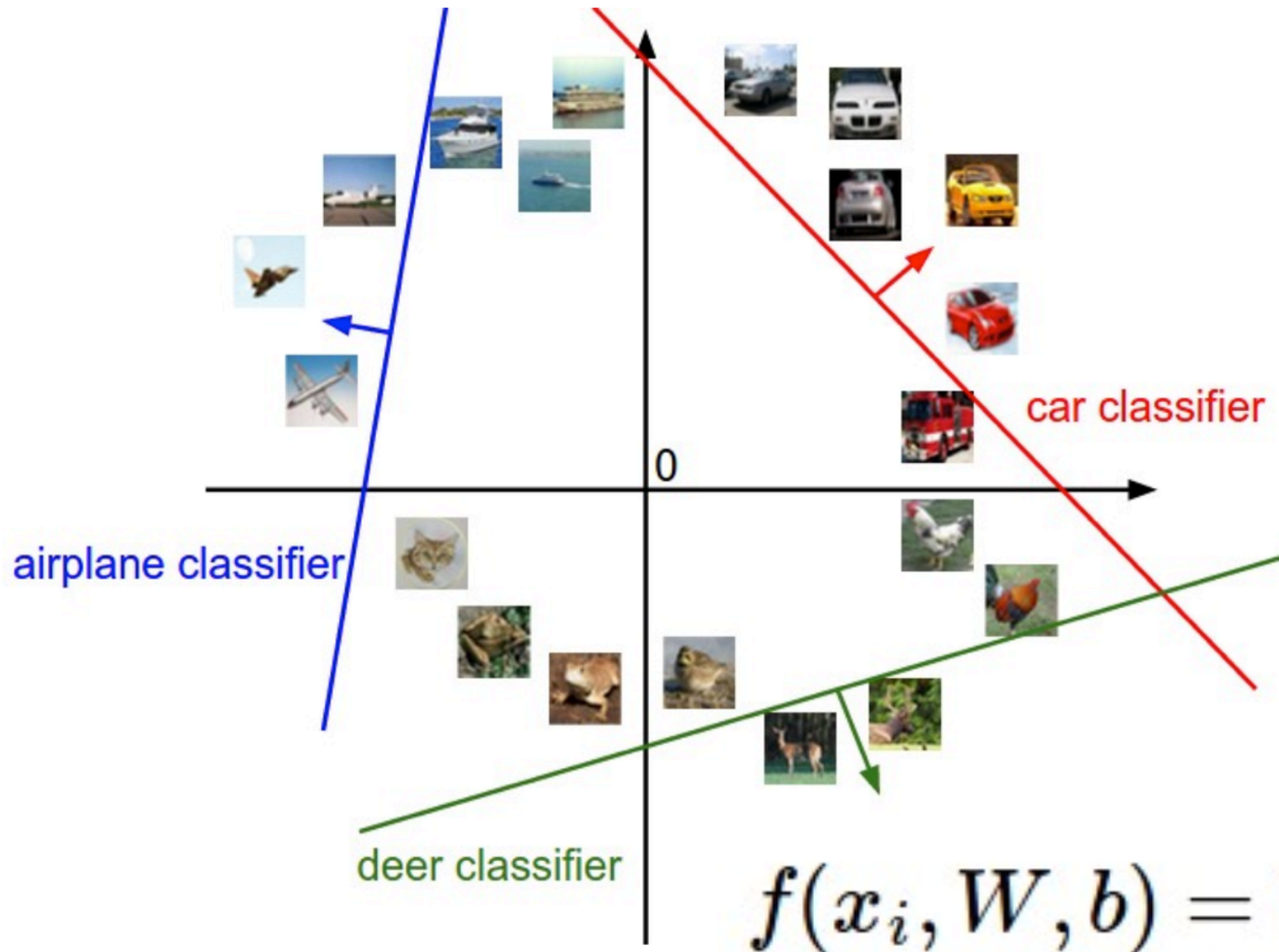


Interpretation: Template matching



$$f(x_i, W, b) = Wx_i + b$$

Geometric Interpretation



$$f(x_i, W, b) = Wx_i + b$$