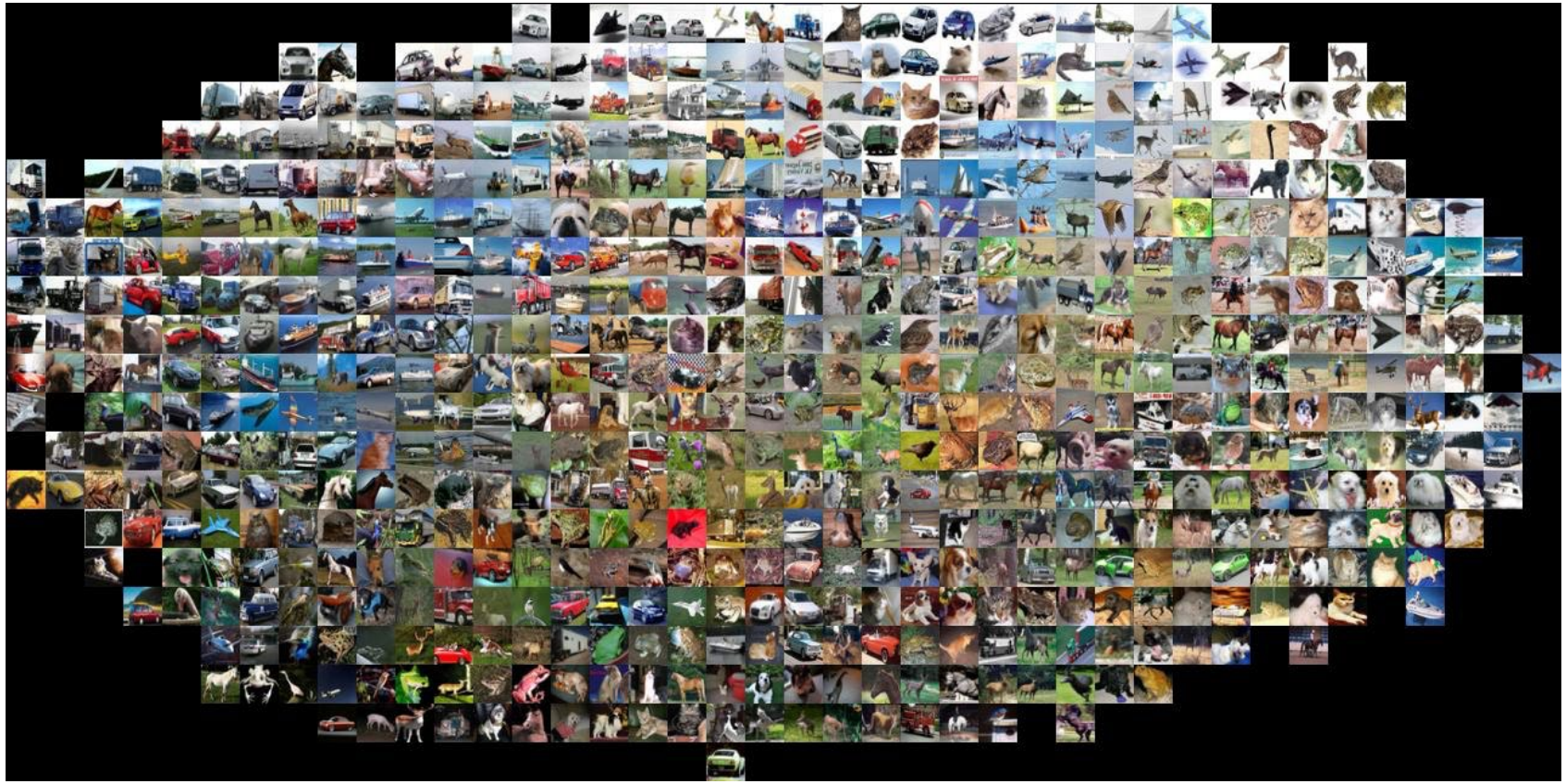
CS7GV1: Computer Vision

Image Classification



Credits: Some Slides from Noah Snavely and Abe Davis (Cornell) and Fei-Fei Li, Justin Johnson, Serena Yeung (Stanford)
http://vision.stanford.edu/teaching/cs231n/

Image Classifiers in a Nutshell

- Input: an image
- Output: the class label for that image

```
def classifier(image):
    //Do some stuff
    return class_label;
```

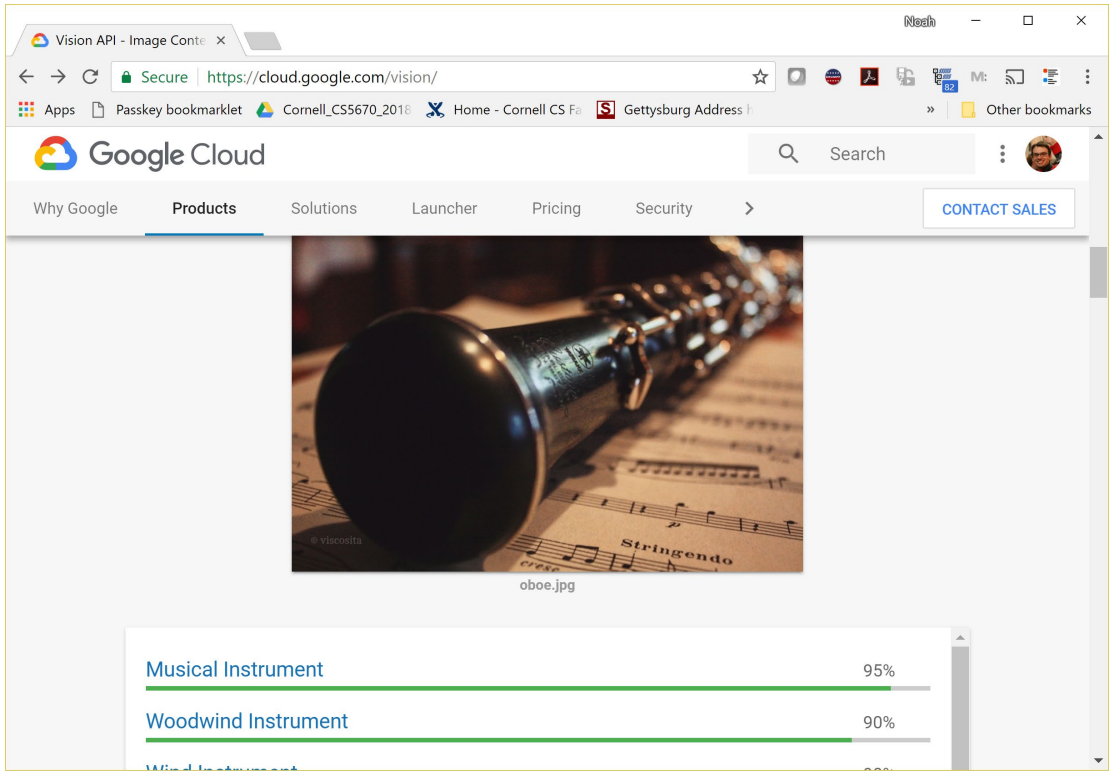
- Label is generally one or more of the discrete labels used in training
  - e.g. {cat, dog, cow, toaster, apple, tomato, truck, … }

$f($  $) =$ “Cat”

$f($  $) =$ “Dog”

$f($  $) =$ “Toaster”

Image classification demo



https://cloud.google.com/vision/docs/drag-and-drop

See also:
https://aws.amazon.com/rekognition/
https://www.clarifai.com/
https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/
…

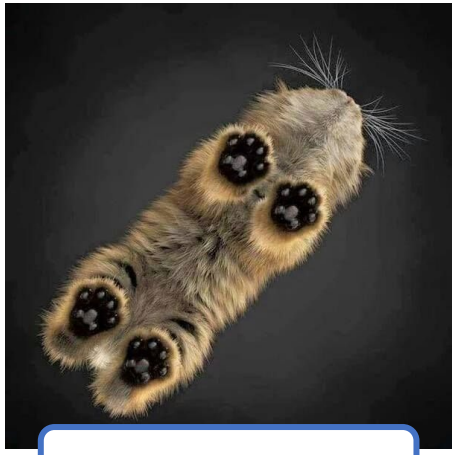
The Semantic Gap

What we see

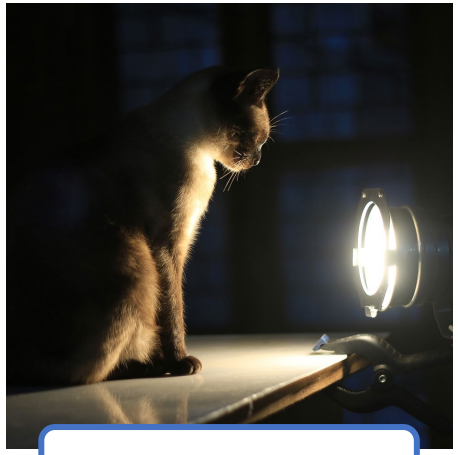
What the computer sees

Variation Makes Recognition Hard

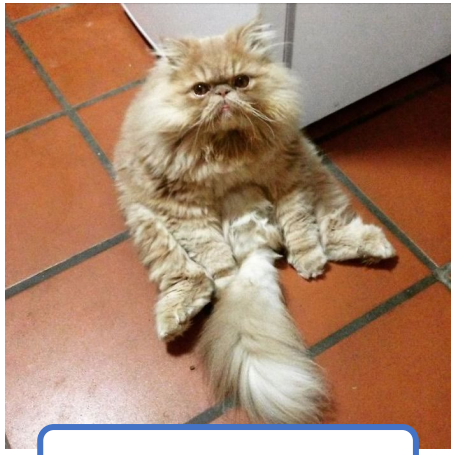
- The same class of object can appear *very* differently in different images



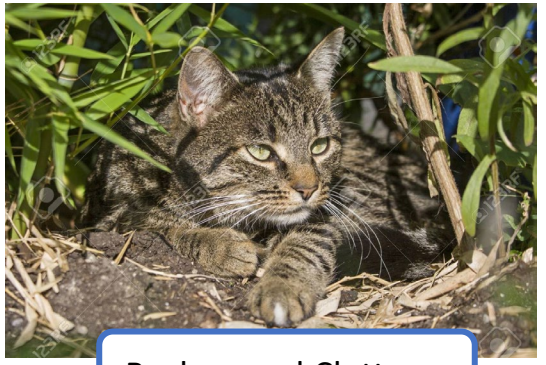
Viewpoint Variation



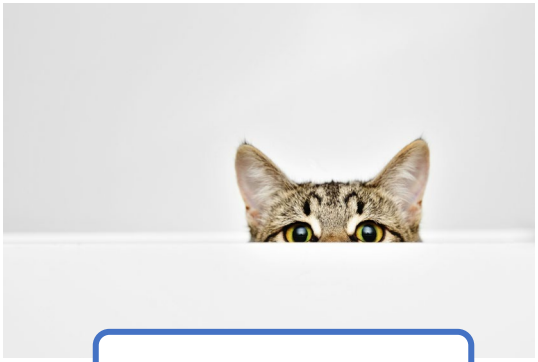
Lighting Variation



Deformation



Background Clutter



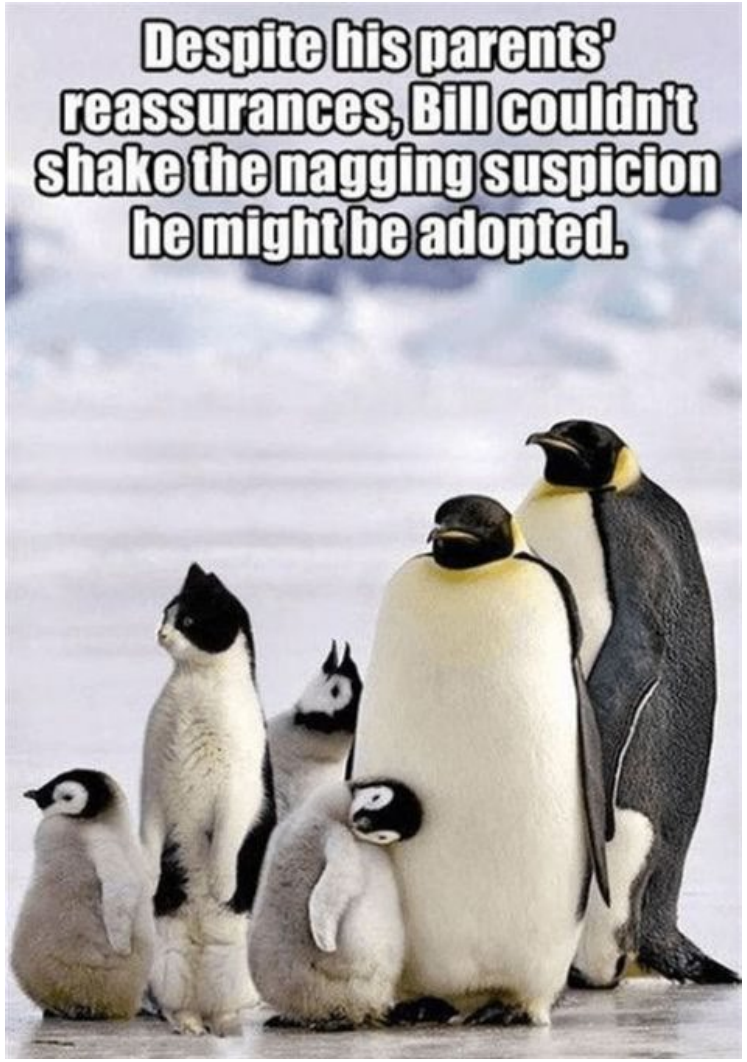
Occlusion

The Problem is Under-constrained

- Distinct realities can produce the same image…

The Problem is Under-constrained

- Distinct realities can produce the same image…
- We generally can’t compute the “right” answer, but we can compute the most likely one…
- We need some kind of prior to condition on. We can learn this prior from data.

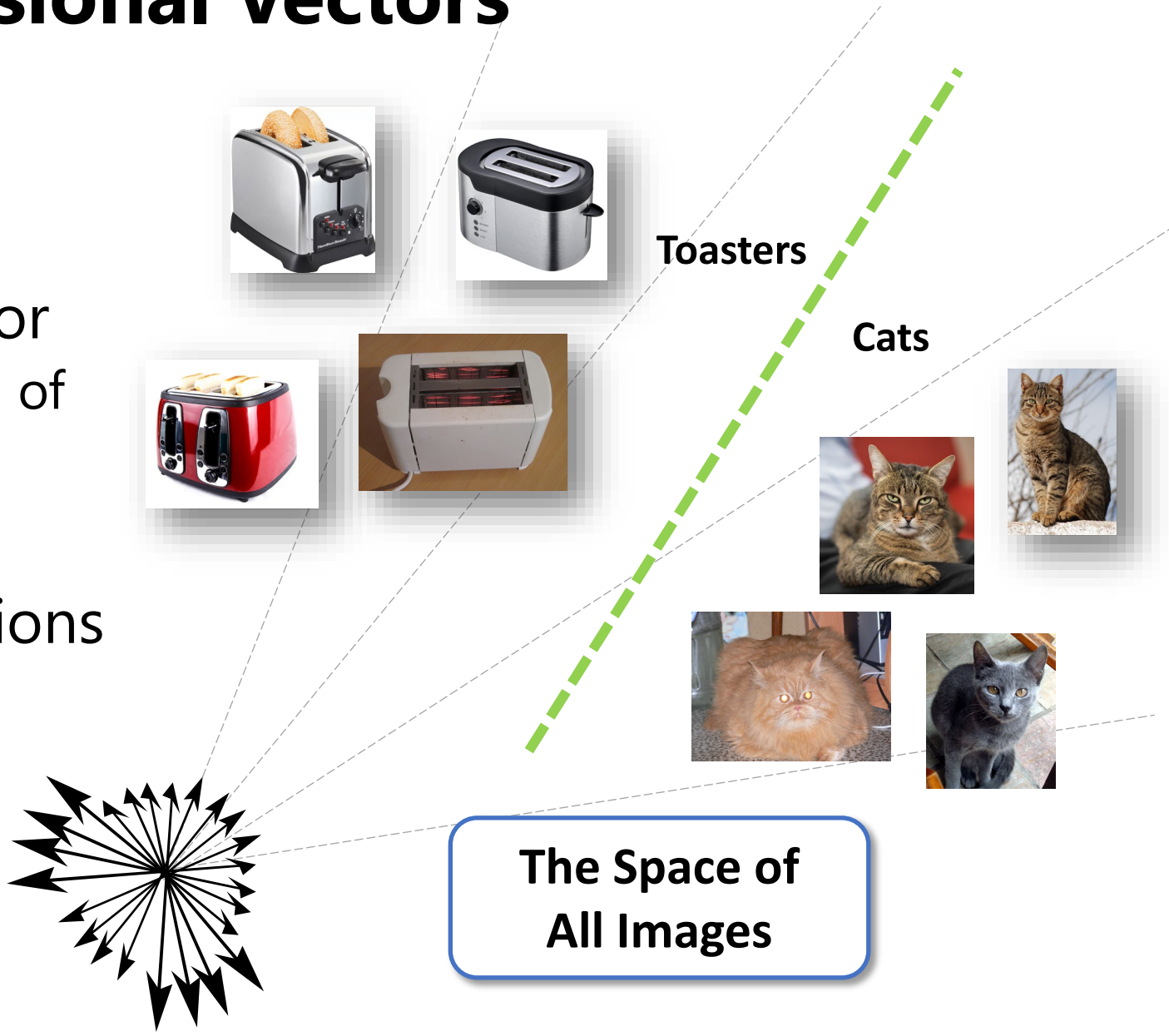


Images As High-Dimensional Vectors

- An image is just a bunch of numbers
- Let's stack them up into a vector
  - Our training data is just a bunch of high-dimensional points now

Images As High-Dimensional Vectors

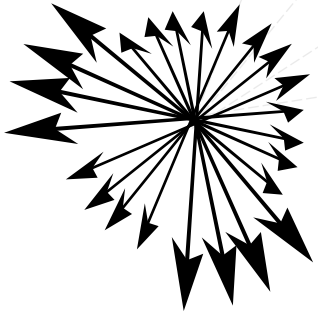
- An image is just a bunch of numbers
- Let's stack them up into a vector
  - Our training data is just a bunch of high-dimensional points now
- Divide space into different regions for different classes



Images As High-Dimensional Vectors

- An image is just a bunch of numbers
- Let's stack them up into a vector
  - Our training data is just a bunch of high-dimensional points now

- Divide space into different regions for different classes



The Space of All Images

Images As High-Dimensional Vectors

- An image is just a bunch of numbers
- Let's stack them up into a vector
  - Our training data is just a bunch of high-dimensional points now

- Divide space into different regions for different classes

or

- Define a distribution over space for each class

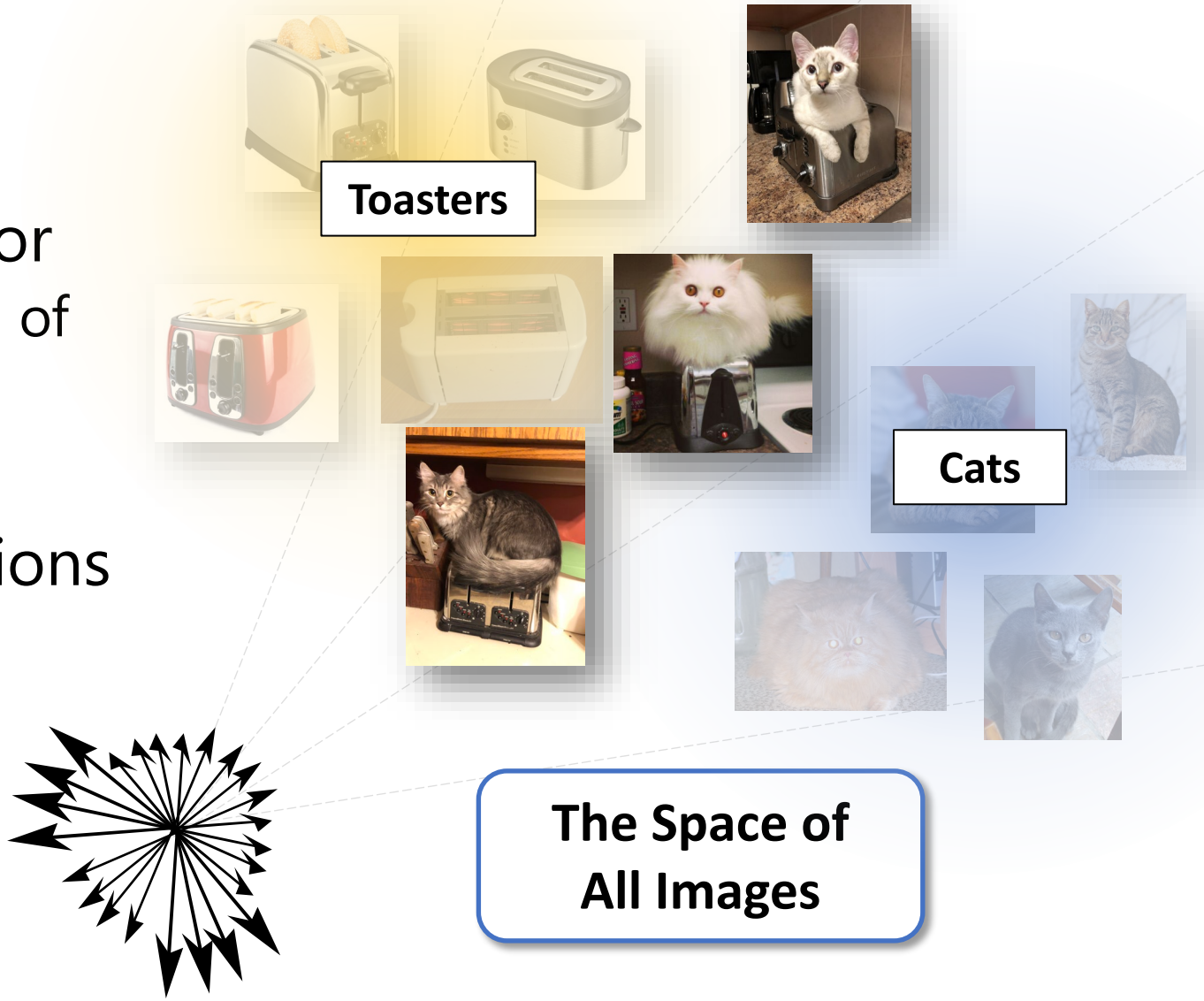
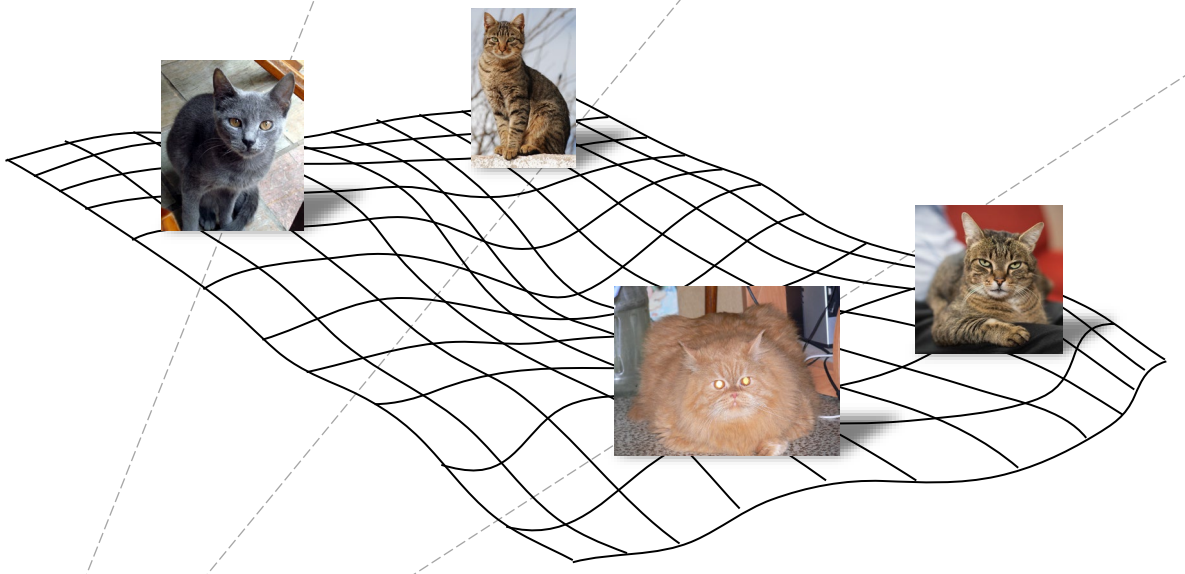
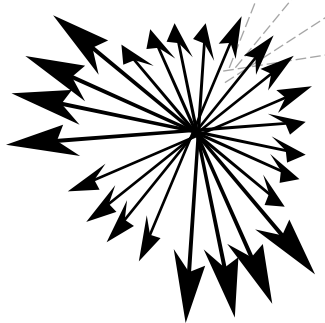


Image Features and Dimensionality Reduction

- How high-dimensional is an image?
  - Let's consider an iPhone X photo:
    - 4032 x 3024 pixels
    - Every pixel has 3 colors
    - 36,578,304 pixels (36.5 Mega pixels)
- In practice, images sit on a lower-dimensional manifold
- Think of image features and dimensionality reduction as ways to represent images by their location on such manifolds





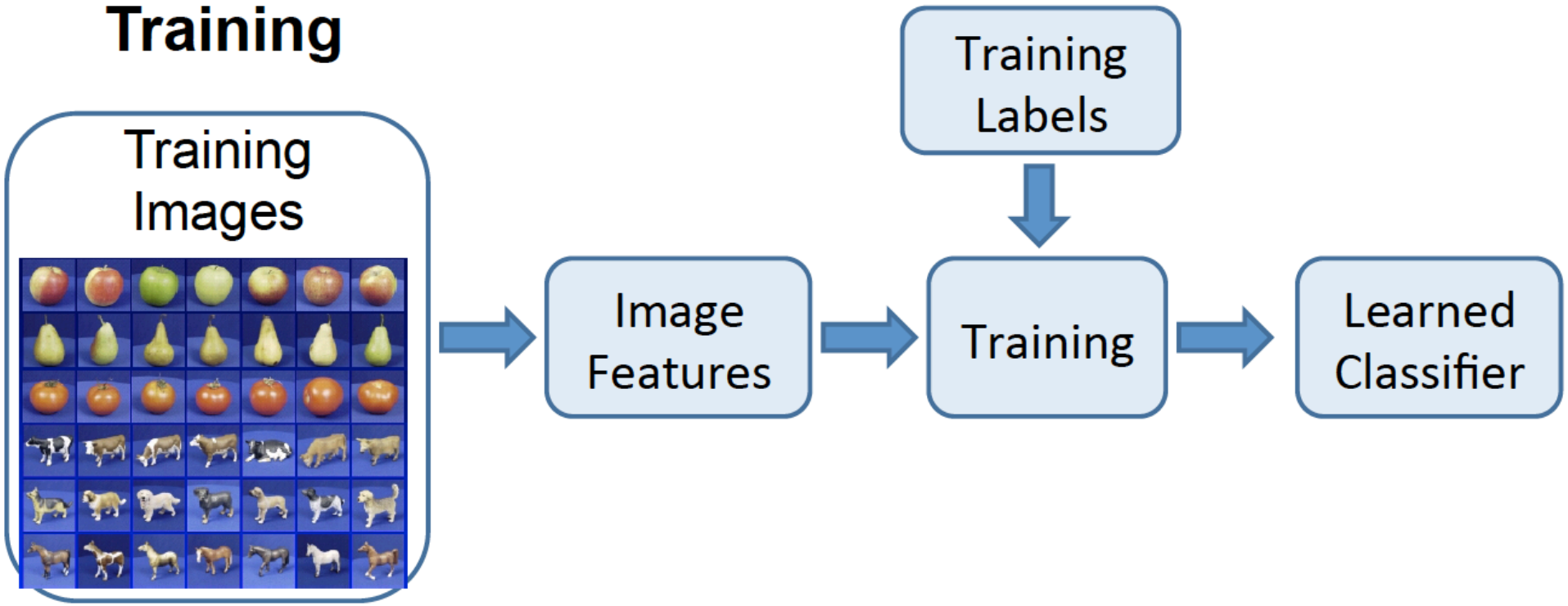
The Space of All Images

Training & Testing a Classifier

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

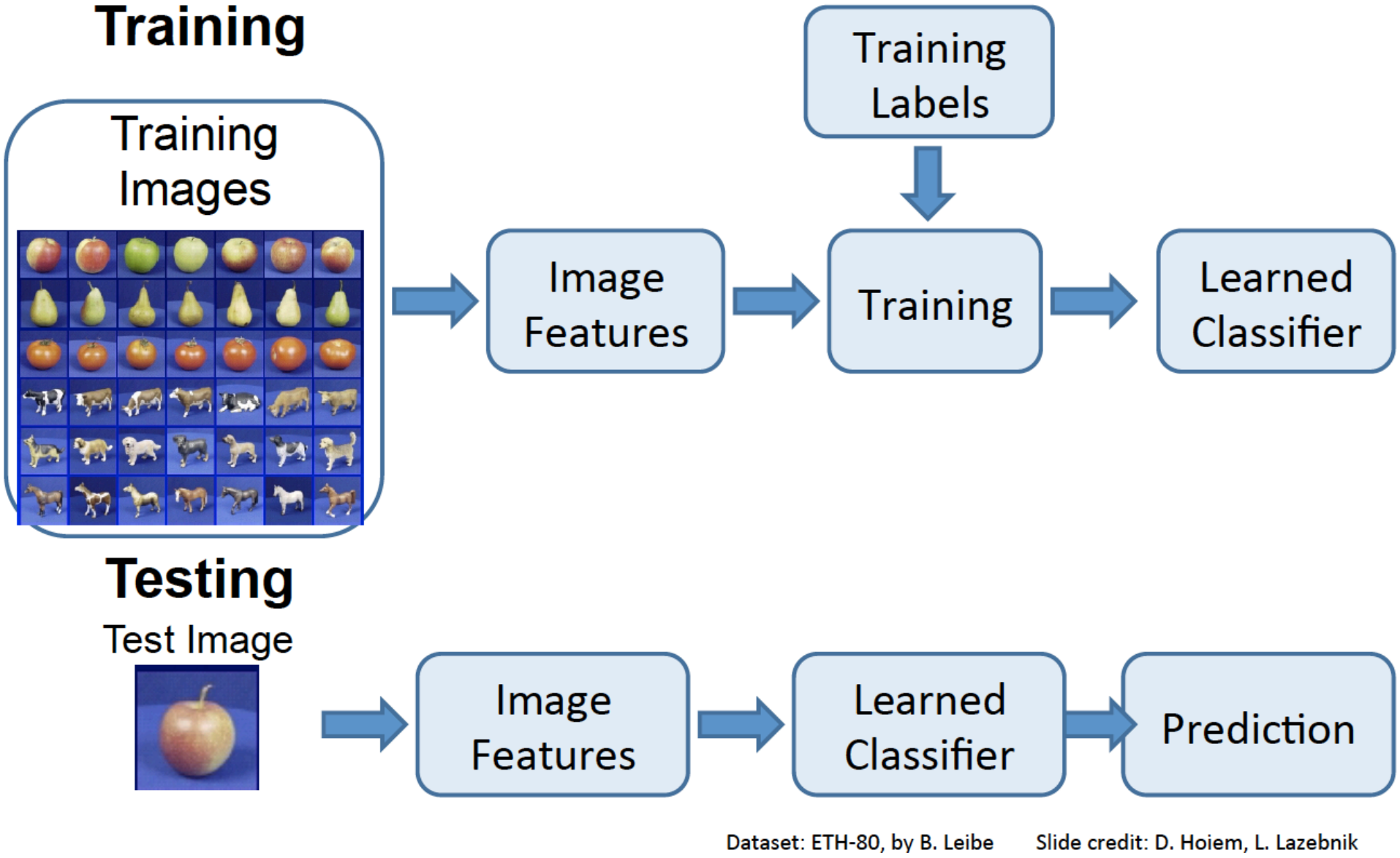
Slide from Andrej Karpathy and Fei-Fei Li http://vision.stanford.edu/teaching/cs231n/

Training & Testing a Classifier



Dataset: ETH-80, by B. Leibe Slide credit: D. Hoiem, L. Lazebnik

Training & Testing a Classifier



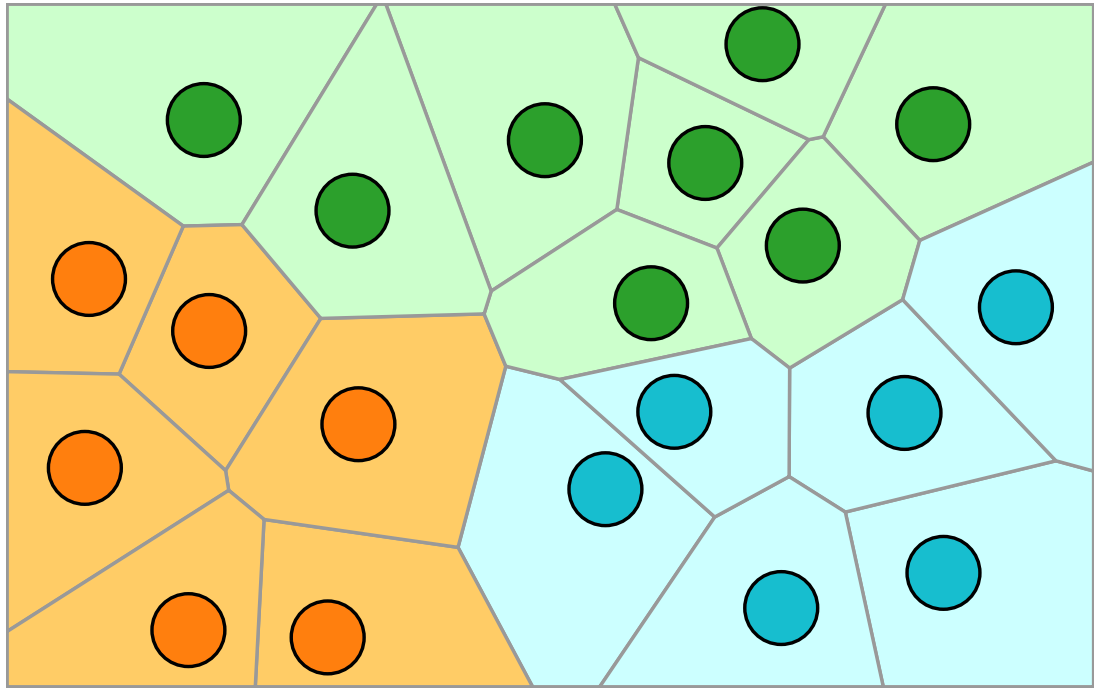
Dataset: ETH-80, by B. Leibe Slide credit: D. Hoiem, L. Lazebnik

Classifiers

- Nearest Neighbor
- kNN (“k-Nearest Neighbors”)
- Linear Classifier
- Neural Network
- Deep Neural Network
- …

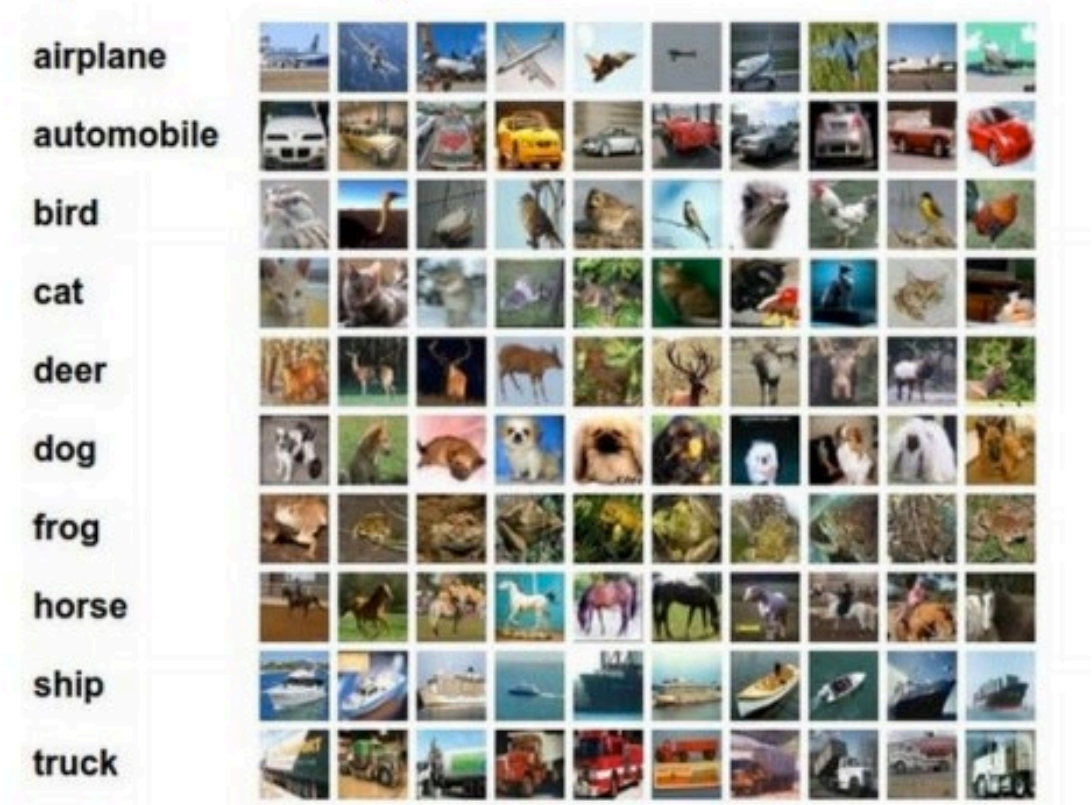
First: Nearest Neighbor (NN) Classifier

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



CIFAR-10 and NN results

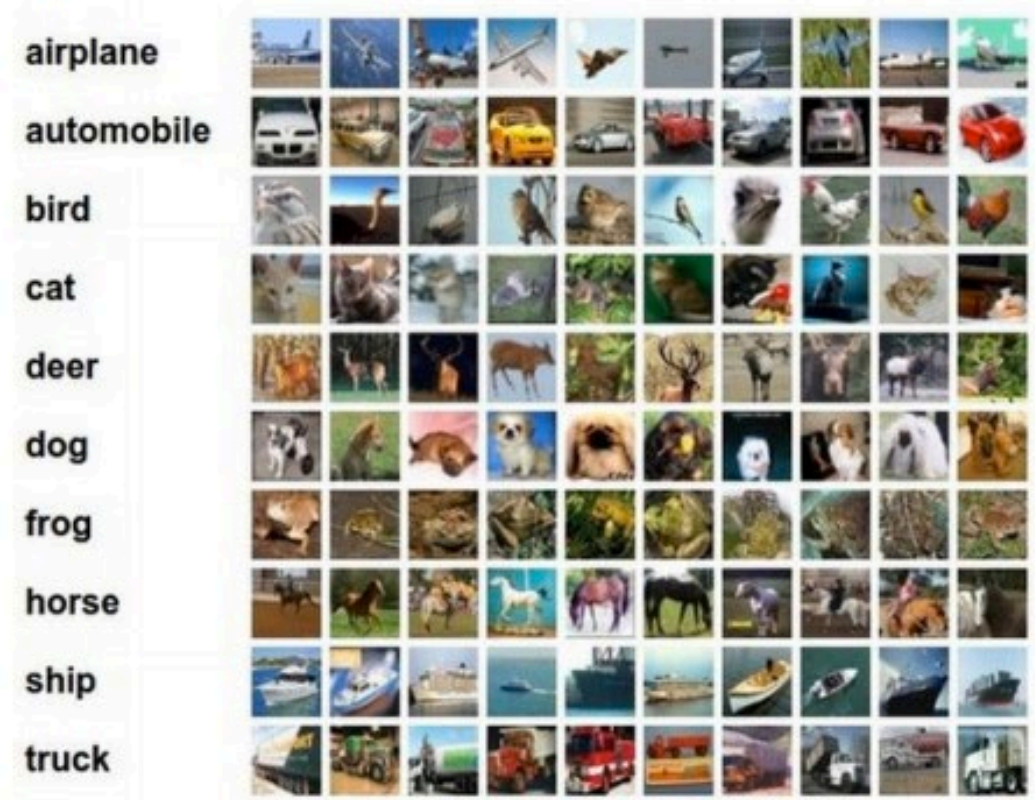
Example dataset: **CIFAR-10**
10 labels
50,000 training images
10,000 test images.



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

CIFAR-10 and NN results

Example dataset: **CIFAR-10**
10 labels
50,000 training images
10,000 test images.

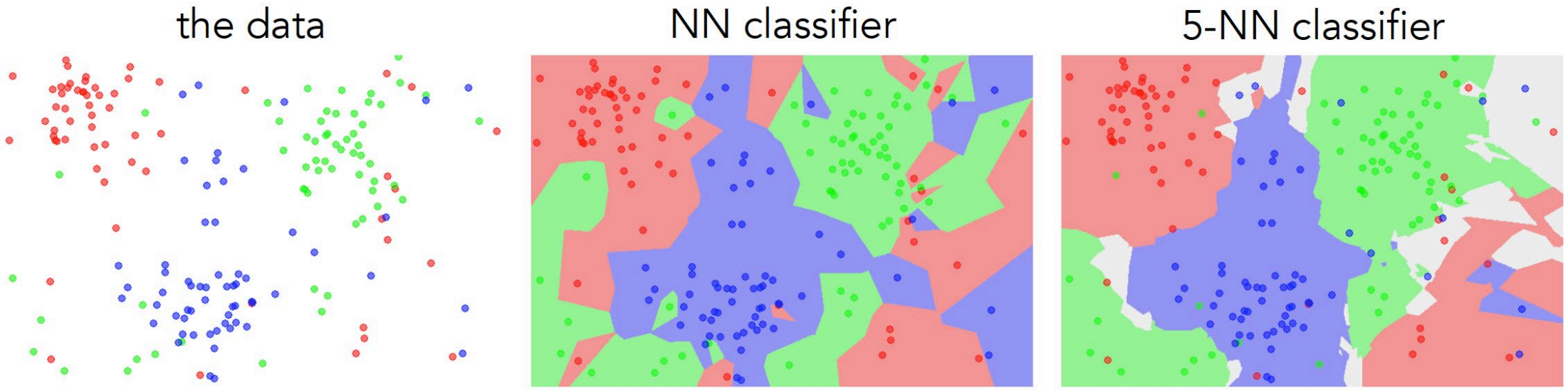


For every test image (first column), examples of nearest neighbors in rows

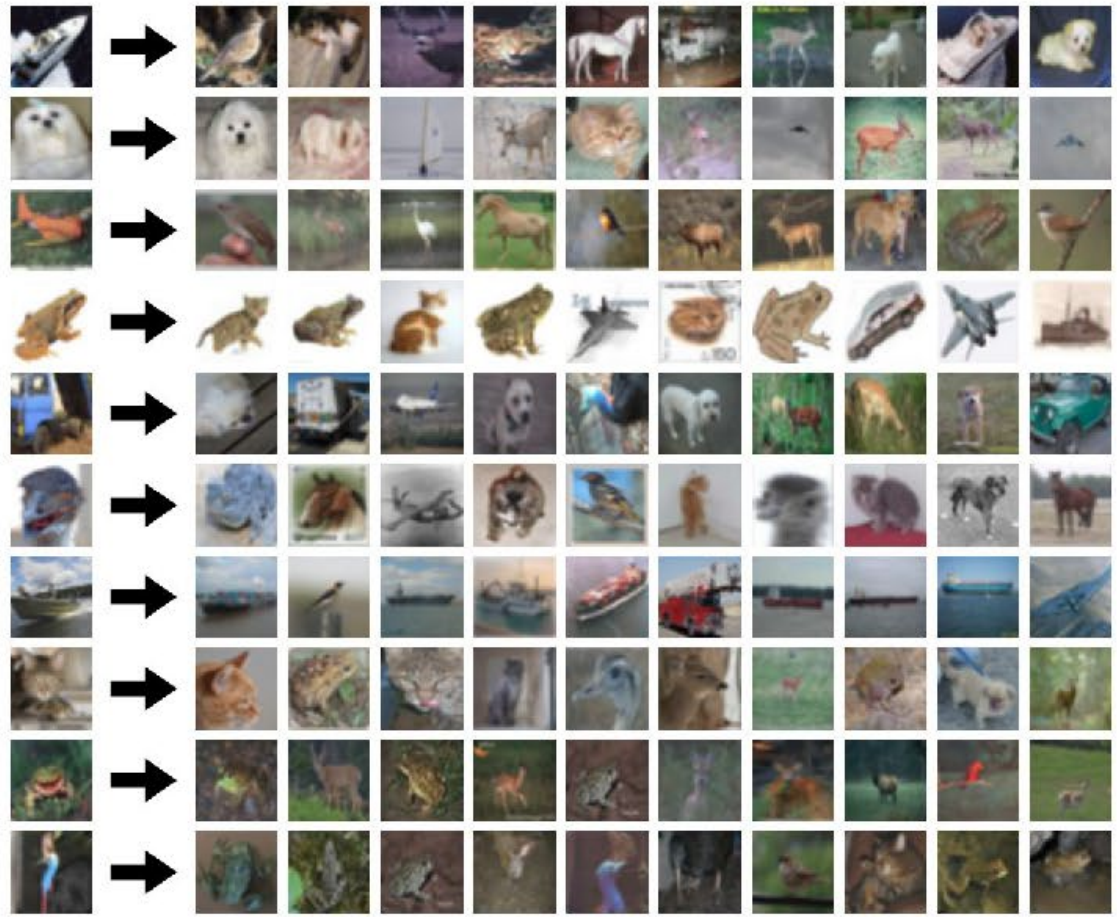
Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

k-nearest neighbor

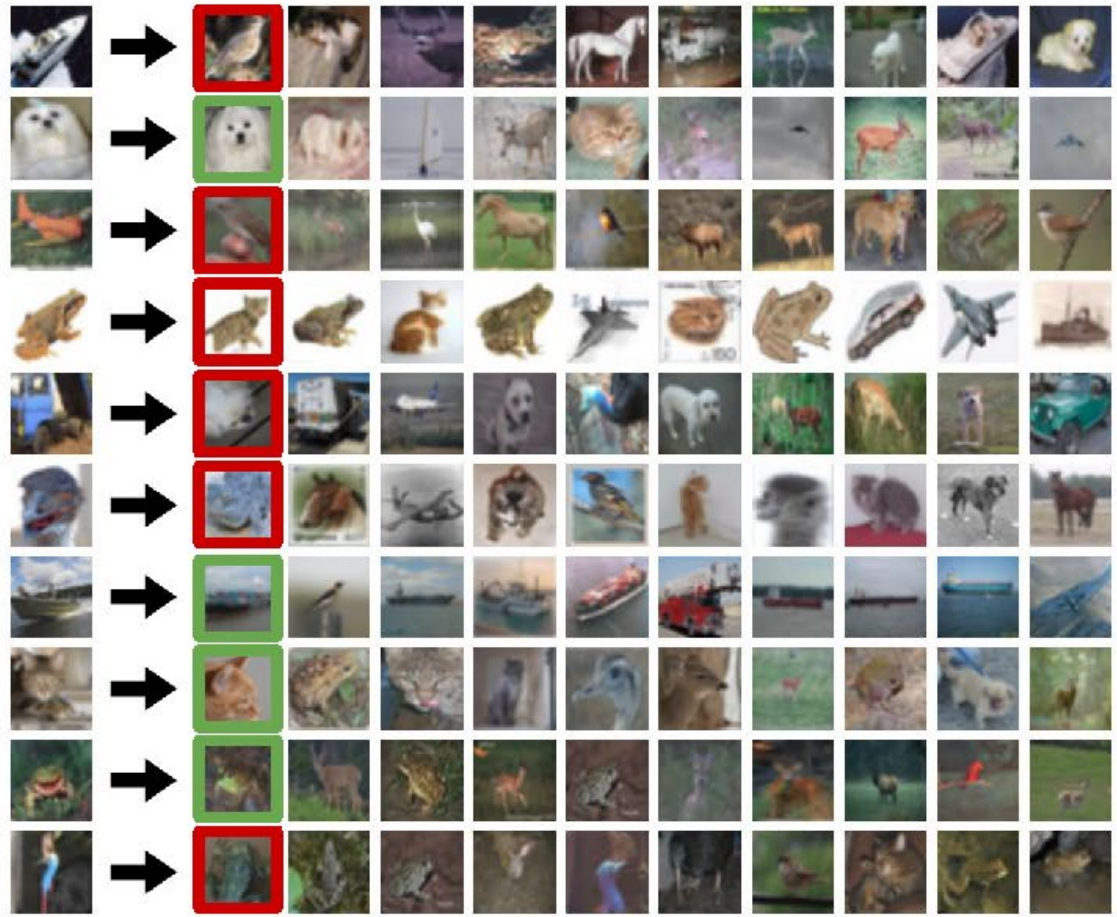
- Find the k closest points from training data
- Take **majority vote** from K closest points



What does this look like?



What does this look like?



How to Define Distance Between Images

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			
56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

\-

training image			
10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences			
46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

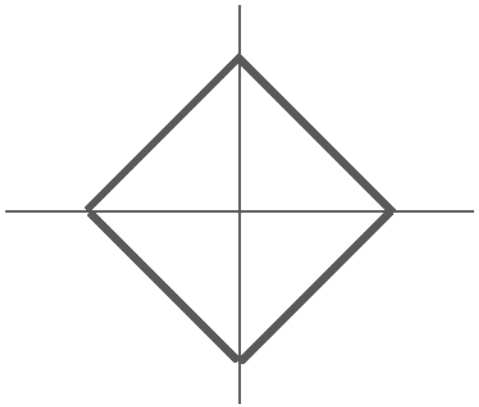
→ 456

Choice of distance metric

- Hyperparameter

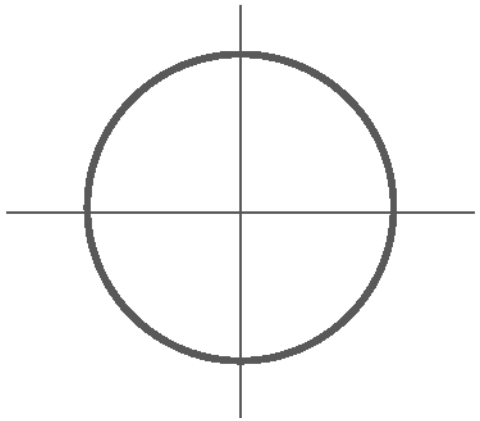
L1 (Manhattan) distance

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



L2 (Euclidean) distance

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



- Two most commonly used special cases of p-norm

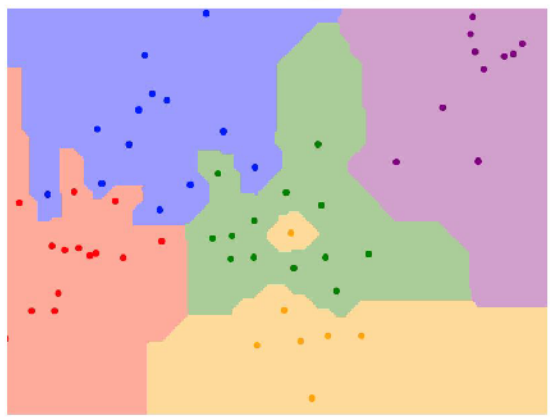
$$||x||_p = \left(|x_1|^p + \cdots + |x_n|^p\right)^{\frac{1}{p}} \qquad p \geq 1, x \in \mathbb{R}^n$$

Slide composited from Andrej Karpathy and Fei-Fei Li http://vision.stanford.edu/teaching/cs231n/

K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

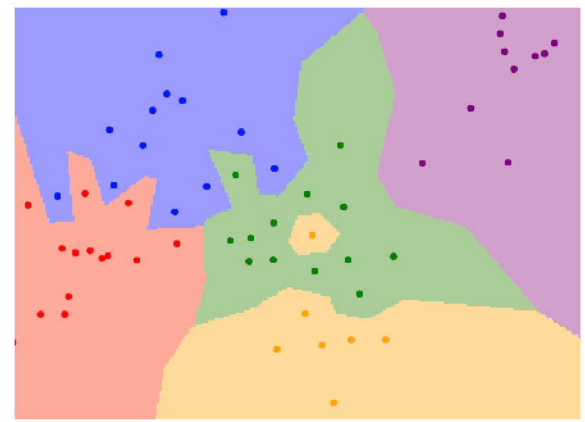
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

L2 (Euclidean) distance

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



K = 1

Demo: http://vision.stanford.edu/teaching/cs231n-demos/knn/

Hyperparameters

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

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

- How do we set them?
  - One option: try them all and see what works best

Setting Hyperparameters

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

Setting Hyperparameters

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

BAD: K = 1 always works perfectly on training data

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

Slide composited from Andrej Karpathy and Fei-Fei Li http://vision.stanford.edu/teaching/cs231n/

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

Slide composited from Andrej Karpathy and Fei-Fei Li http://vision.stanford.edu/teaching/cs231n/

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

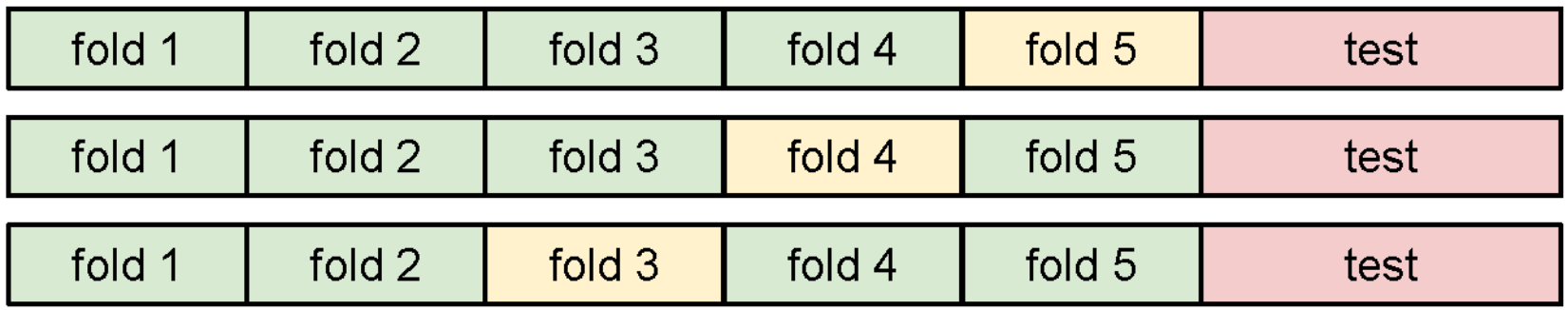
Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

Slide composited from Andrej Karpathy and Fei-Fei Li http://vision.stanford.edu/teaching/cs231n/

Setting Hyperparameters

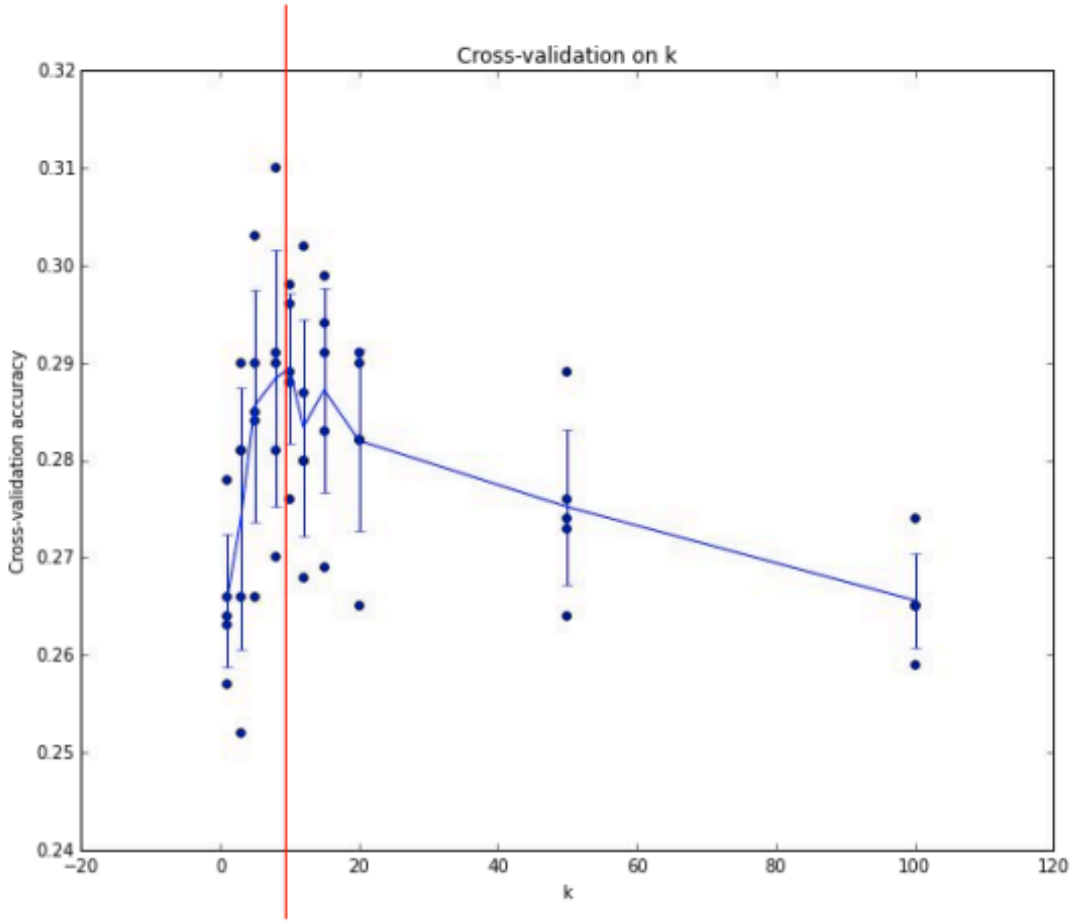
Idea #4: **Cross-Validation**: Split data into **folds**,
try each fold as validation and average the results



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

Slide composited from Andrej Karpathy and Fei-Fei Li http://vision.stanford.edu/teaching/cs231n/

Hyperparameter Tuning



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)

Recap: How to pick hyperparameters?

- Methodology
  - Train and test
  - Train, validate, test

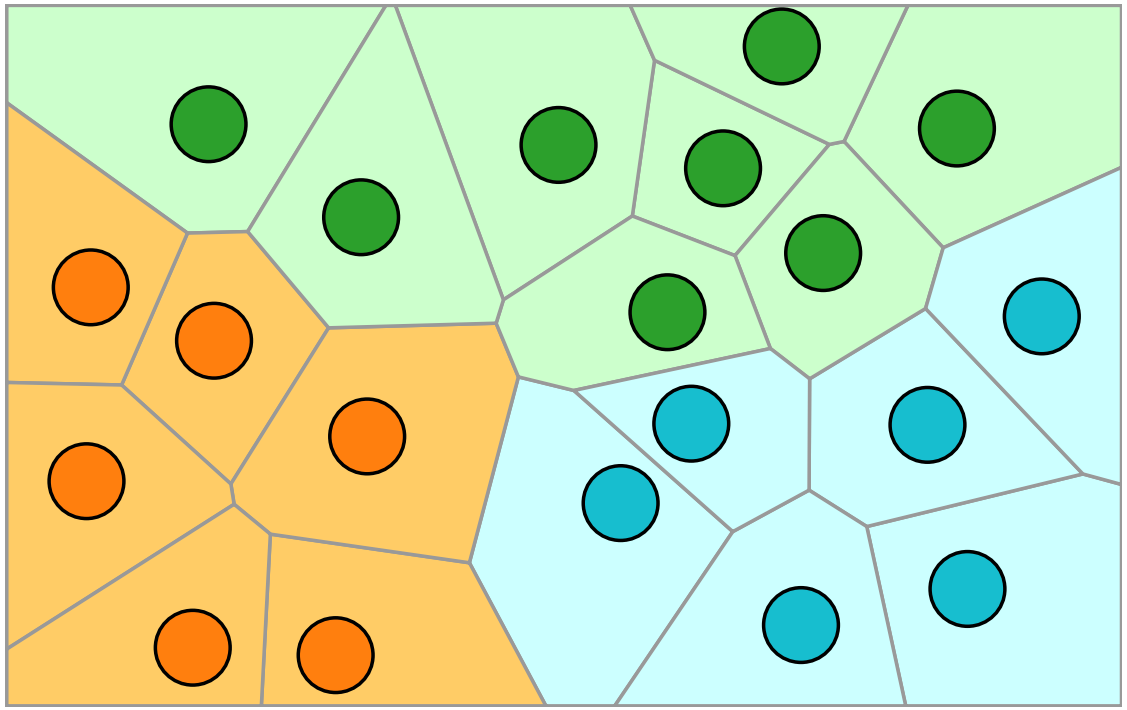
- Train for original model
- Validate to find hyperparameters
- Test to understand generalizability

kNN -- Complexity and Storage

- N training images, M test images

- Training: O(1)
- Testing: O(MN)

- We often need the opposite:
  - Slow training is ok
  - Fast testing is necessary

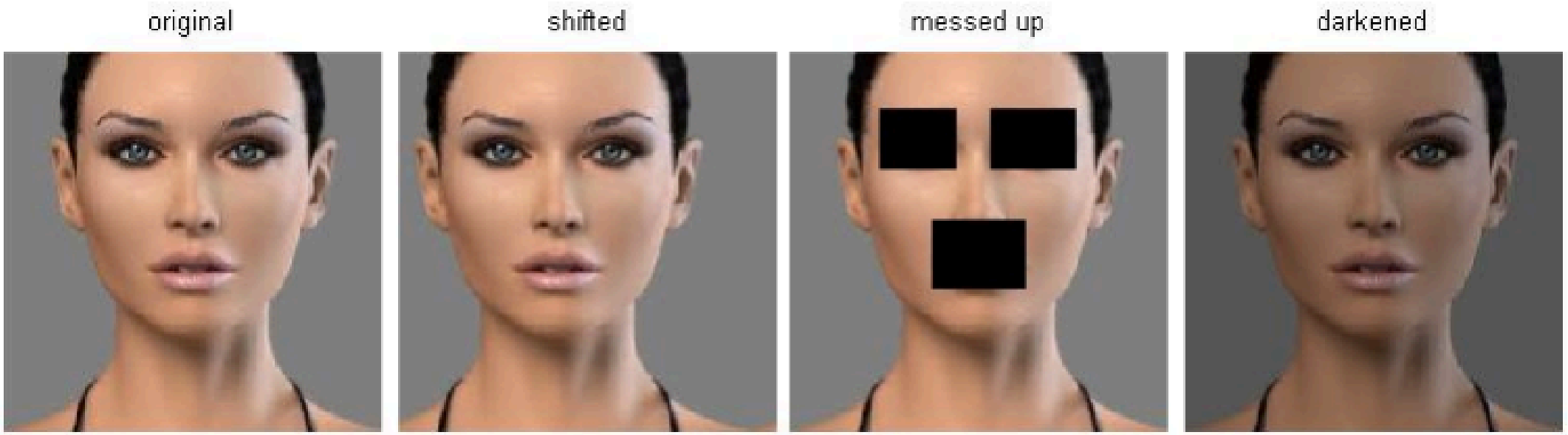


k-Nearest Neighbors: Summary

- 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!

Problems with KNN: Distance Metrics

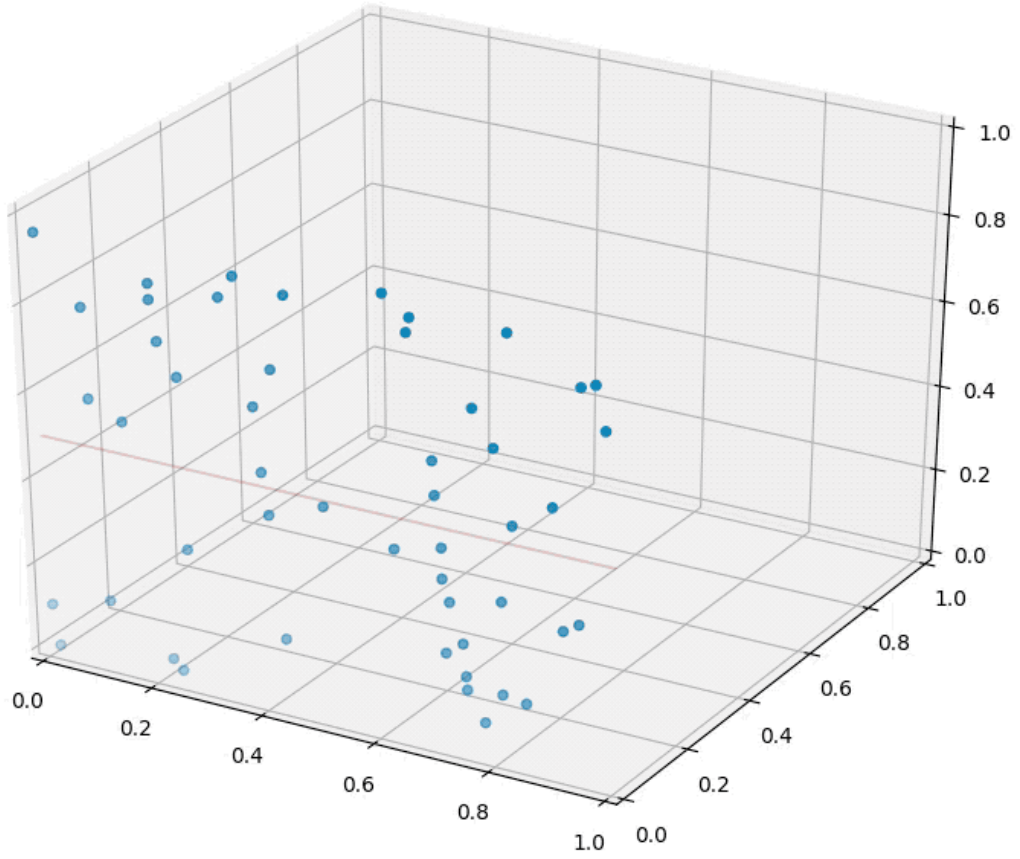
- terrible performance at test time
- distance metrics on level of whole images can be very unintuitive



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

Problems with KNN: The Curse of Dimensionality

- As the number of dimensions increases, the same amount of data becomes more sparse.
- Amount of data we need ends up being exponential in the number of dimensions



Animation from https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html