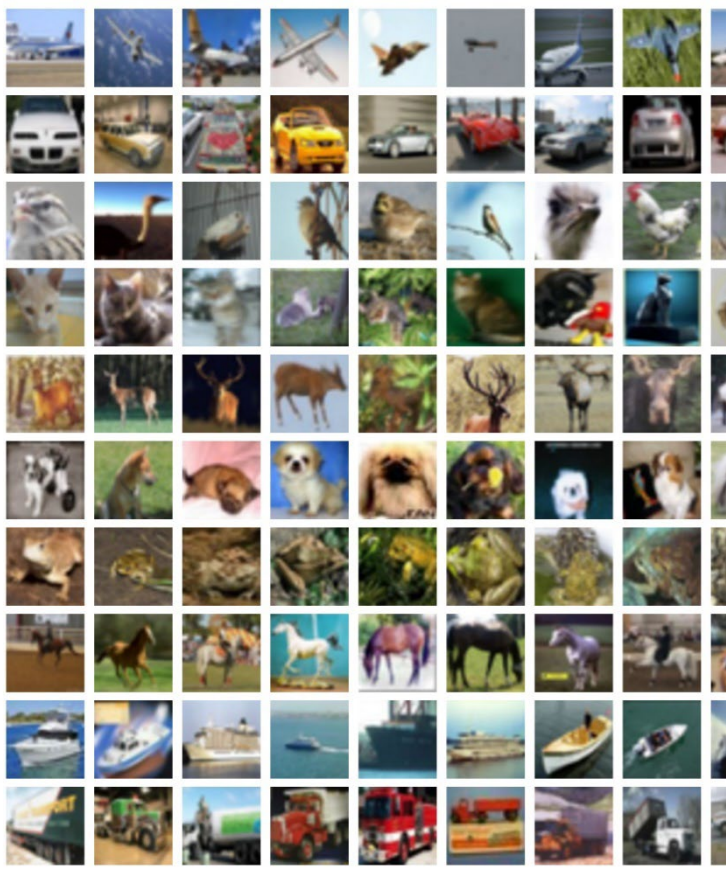
DEEP LEARNING: IMAGE CLASSIFICATION



Chih-Chung Hsu (許志仲)
Institute of Data Science
National Cheng Kung University
https://cchsu.info

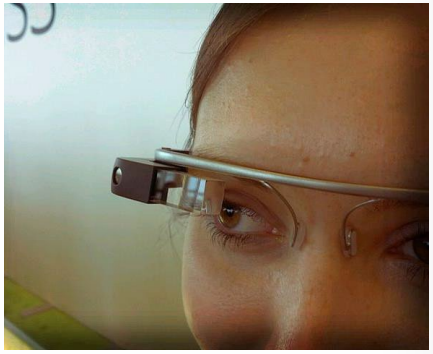








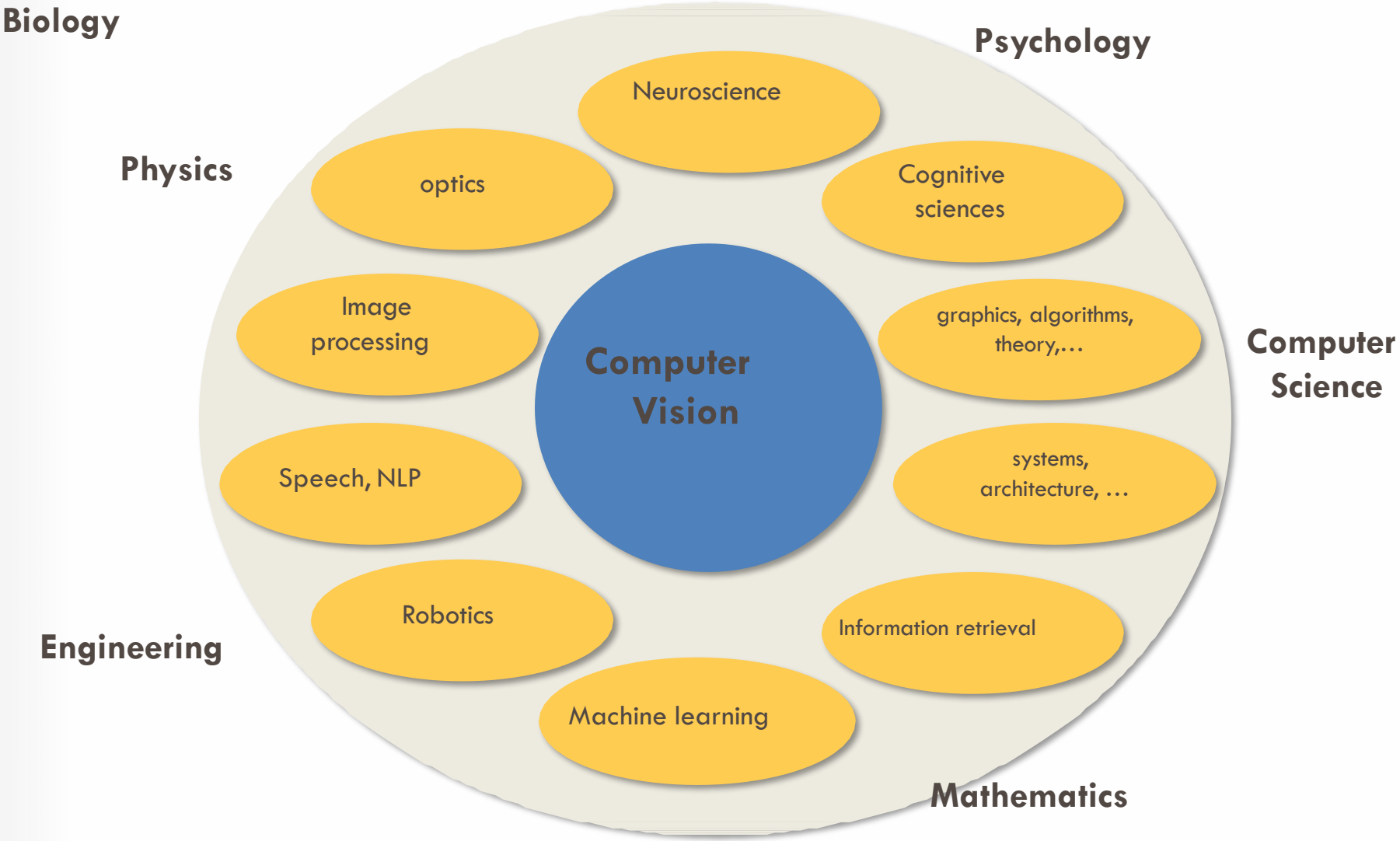


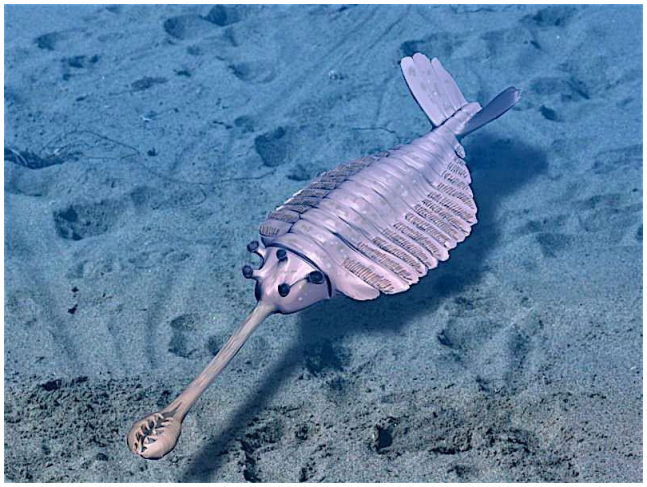
Top row, left to right:
Image by Roger H Goun is licensed under CC BY 2.0
Image is CC0 1.0 public domain
Image is CC0 1.0 public domain
Image is CC0 1.0 public domain

Middle row, left to right
Image by BGPHP Conference is licensed under CC BY 2.0; changes made
Image is CC0 1.0 public domain
Image by NASA is licensed under CC BY 2.0
Image is CC0 1.0 public domain

Bottom row, left to right Image
is CC0 1.0 public domain
Image by Derek Keats is licensed under CC BY 2.0; changes made
Image is public domain
Image is licensed under CC-BY 2.0; changes made


Biology
Psychology
Physics
Neuroscience
optics
Cognitive sciences
Image processing
Computer Vision
graphics, algorithms, theory,…
Computer Science
Speech, NLP
systems, architecture, …
Robotics
Information retrieval
Engineering
Machine learning
Mathematics

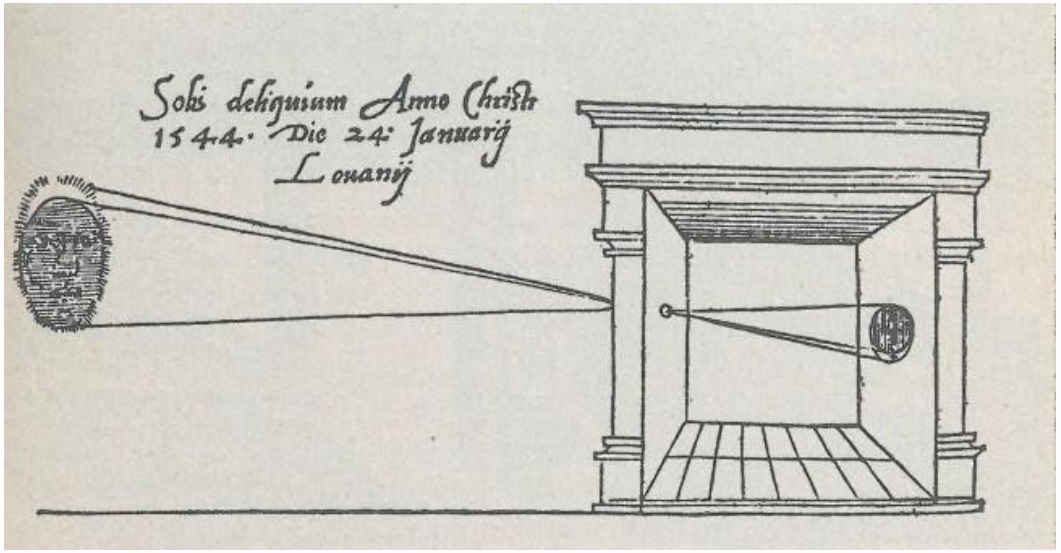
Evolution's Big Bang

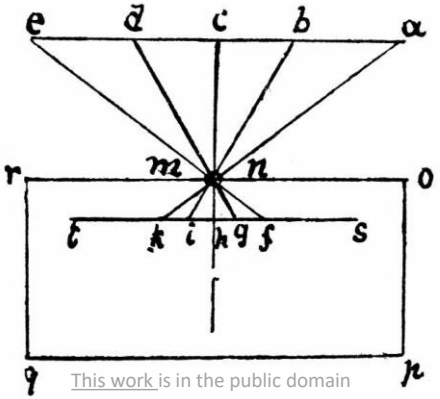


543 million years, B.C.

Camera Obscura

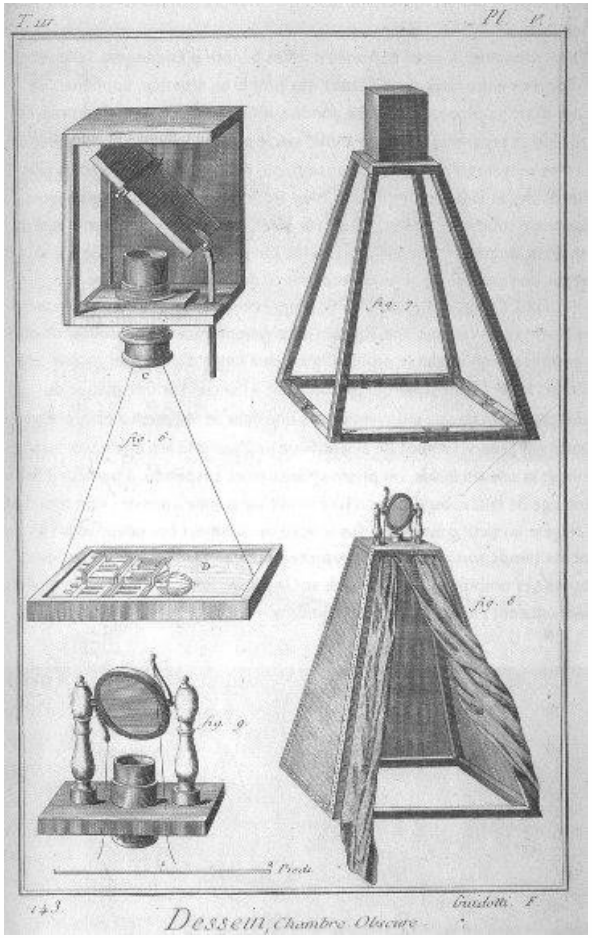
Gemma Frisius, 1545





Leonardo da Vinci,
16th Century AD

Encyclopedie, 18th Century



MASSACHUSETTS INSTITUTE OF TECHNOLOGY

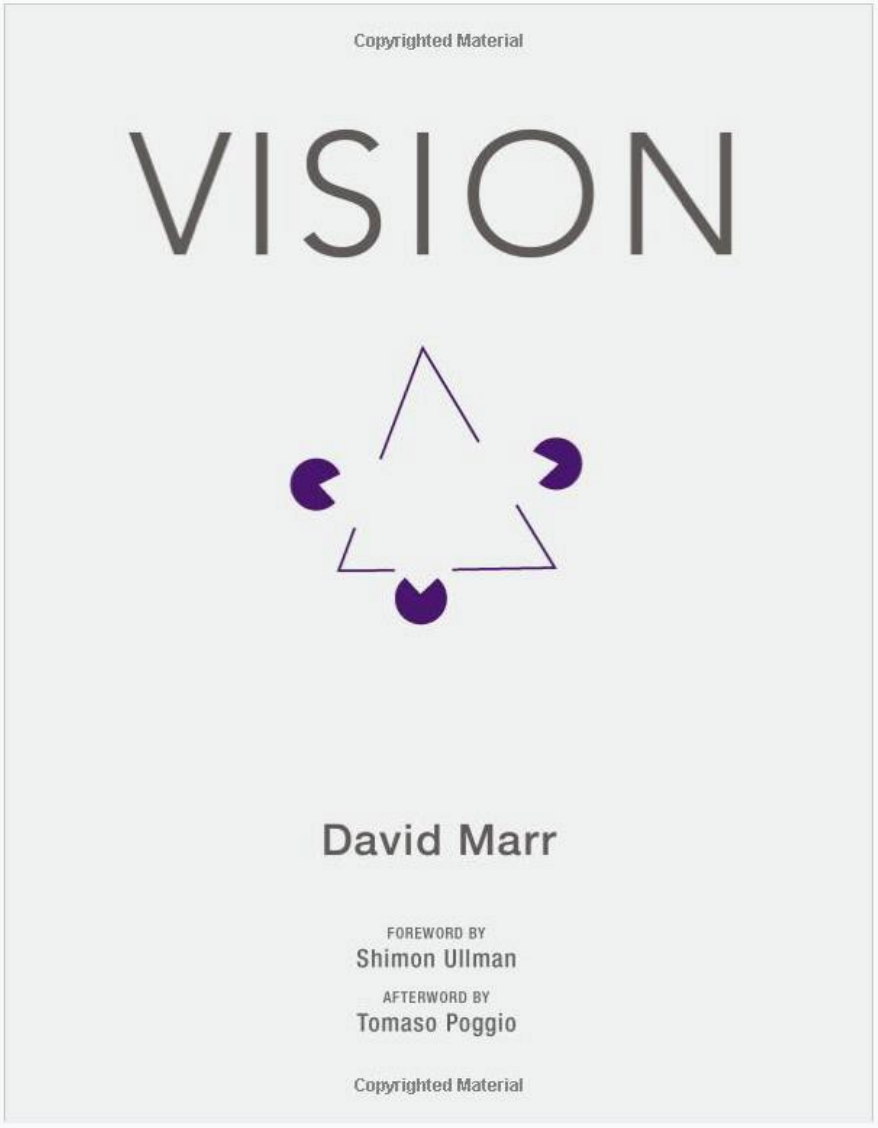
PROJECT MAC

Artificial Intelligence Group July 7, 1966
Vision Memo. No. 100.

THE SUMMER VISION PROJECT

Seymour Papert

The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

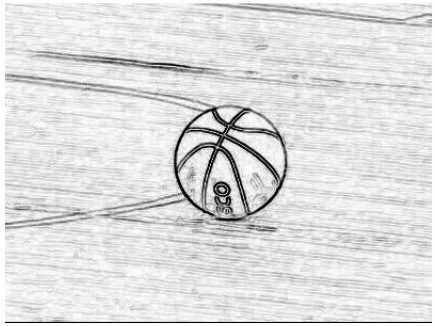


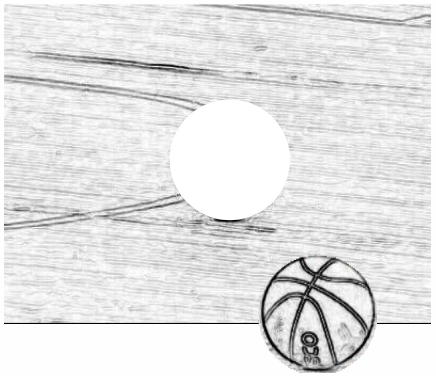
David Marr, 1970s

Input image

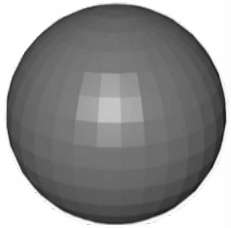
This image is CC0 1.0 public domain

Edge image





3-D model



This image is CC0 1.0 public domain

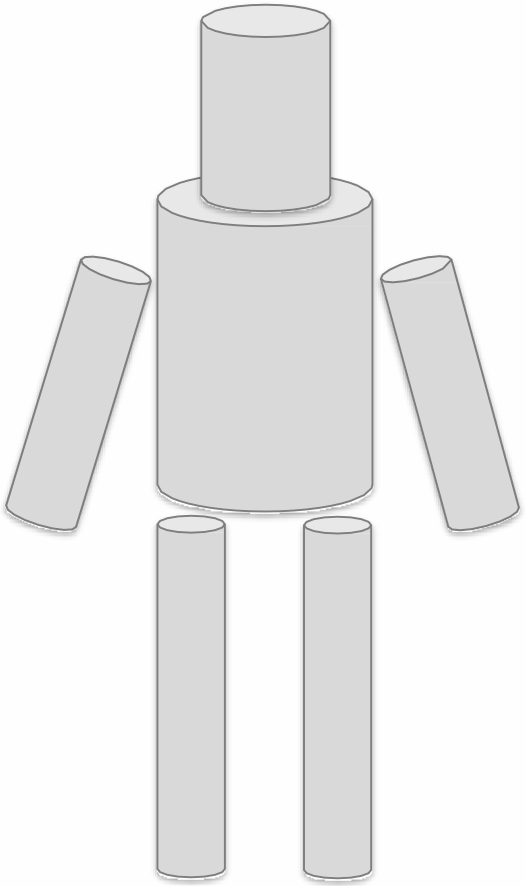
2 ½-D sketch

Input Image	Primal Sketch	2 ½-D Sketch	3-D Model Representation
Perceived intensities	Zero crossings, blobs, edges, bars, ends, virtual lines, groups, curves boundaries	Local surface orientation and discontinuities in depth and in surface orientation	3-D models hierarchically organized in terms of surface and volumetric primitives

Stages of Visual Representation, David Marr, 1970s

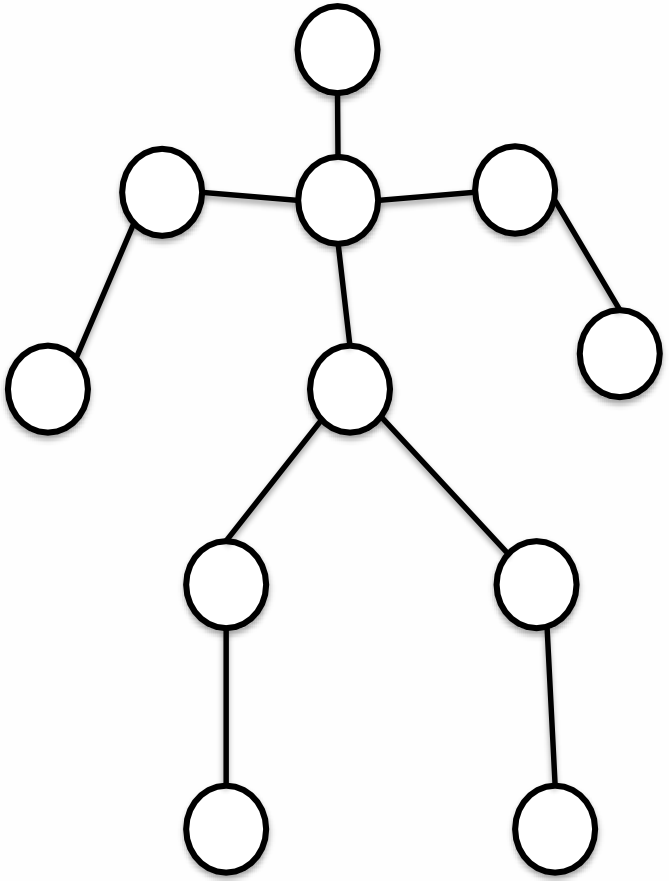
- Generalized Cylinder

  Brooks & Binford, 1979



- Pictorial Structure

  Fischler and Elschlager, 1973



Edge map to form the objects

David Lowe, 1987

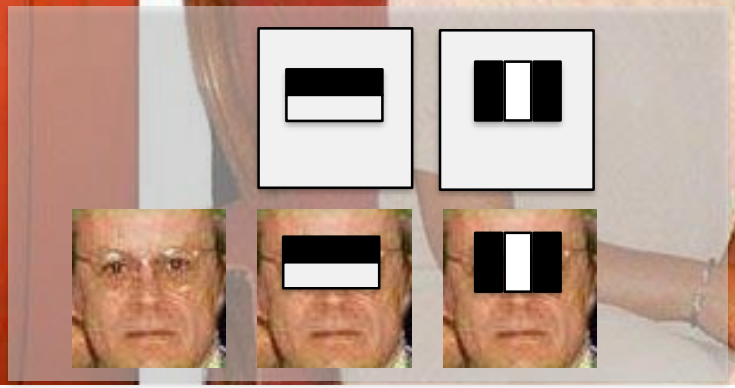
Normalized Cut (Shi & Malik, 1997)

Image is CC BY 3.0

Image is public domain

Image is CC-BY 2.0; changes made

Face Detection, Viola & Jones, 2001

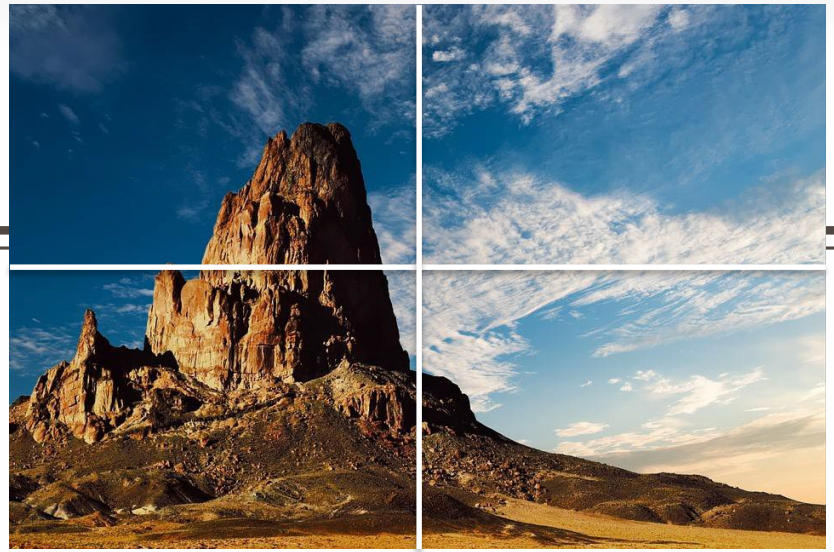


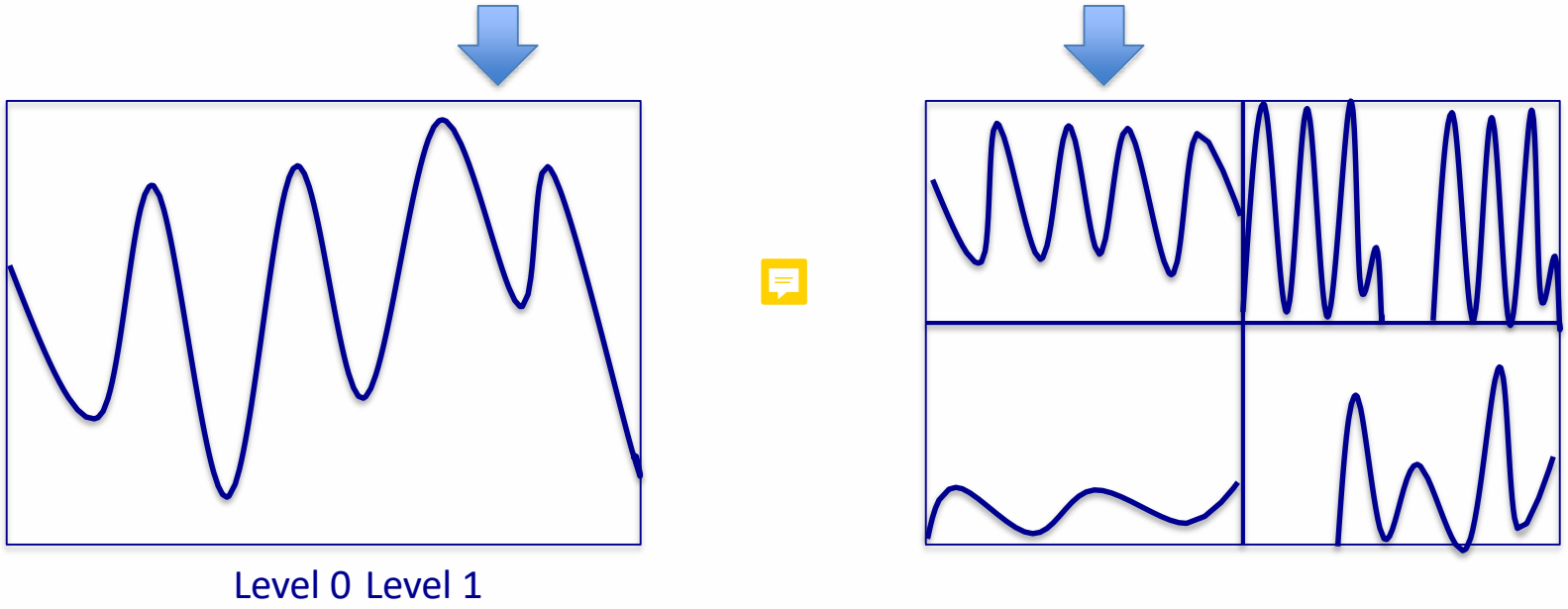
Local Feature and Matching

Image is public domain

Image is public domain

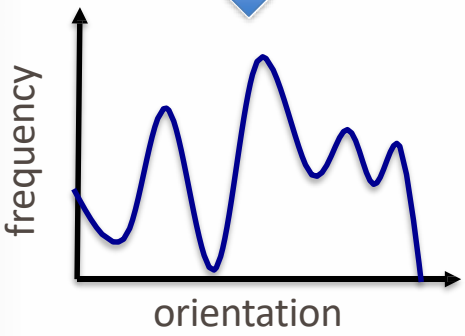
"SIFT" & Object Recognition, David Lowe, 1999



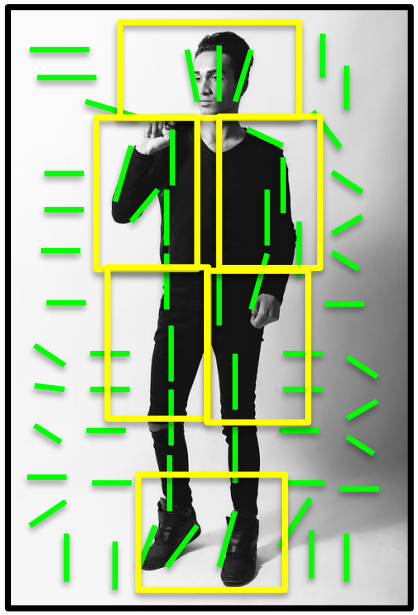


Level 0 Level 1

Spatial Pyramid Matching, Lazebnik, Schmid & Ponce, 2006



Histogram of Gradients (HoG)
Dalal & Triggs, 2005



Deformable Part Model
Felzenswalb, McAllester, Ramanan, 2009

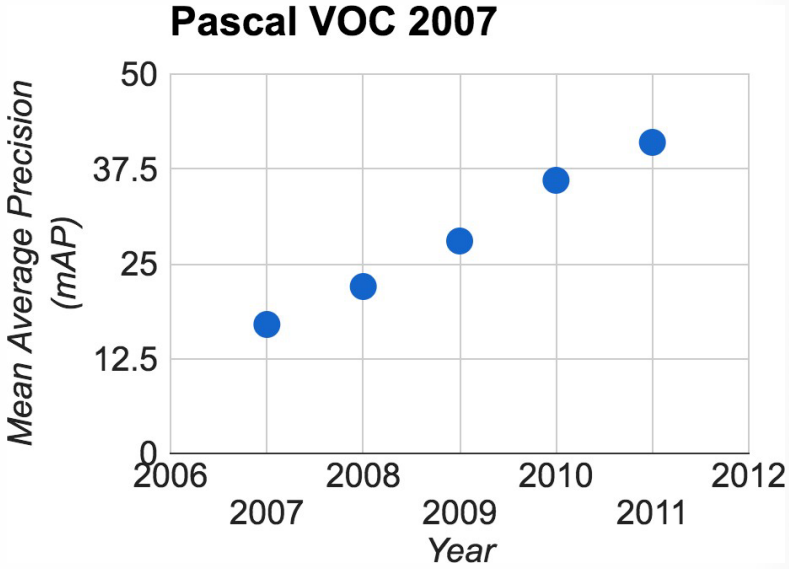
PASCAL Visual Object Challenge (20 object categories)

[Everingham et al. 2006-2012]

Image is CC0 1.0 public domain

Image is CC0 1.0 public domain

Image is CC0 1.0 public domain



IMAGENET

www.image-net.org

22K categories and **15M** images

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
- Food
- Materials
- Structures
- Artifact
  - Tools
  - Appliances
  - Structures
- Person
- Scenes
  - Indoor
  - Geological Formations
- Sport Activities

Deng, Dong, Socher, Li, Li, & Fei-Fei, 2009

IMAGENET Large Scale Visual Recognition Challenge

The Image Classification Challenge:
1,000 object classes
1,431,167 images

Output:
Scale

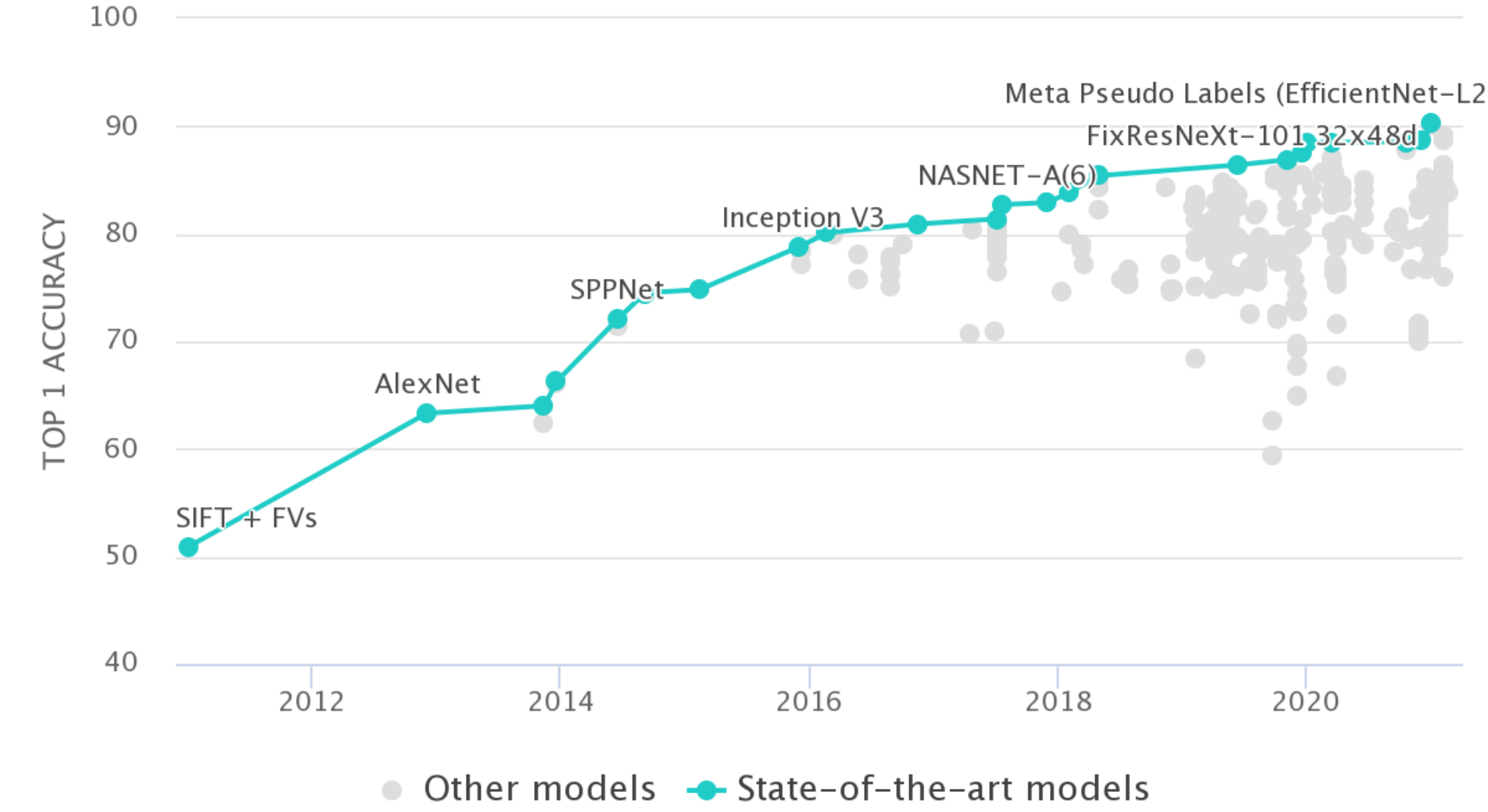
Steel drum
Drumstick
Mud turtle

Output:
Scale

Giant panda
Drumstick
Mud turtle

Russakovsky et al. IJCV 2015

IMAGENET Large Scale Visual Recognition Challenge



Russakovsky et al. IJCV 2015

Image by US Army is licensed under CC BY 2.0

Image is CC0 1.0 public domain

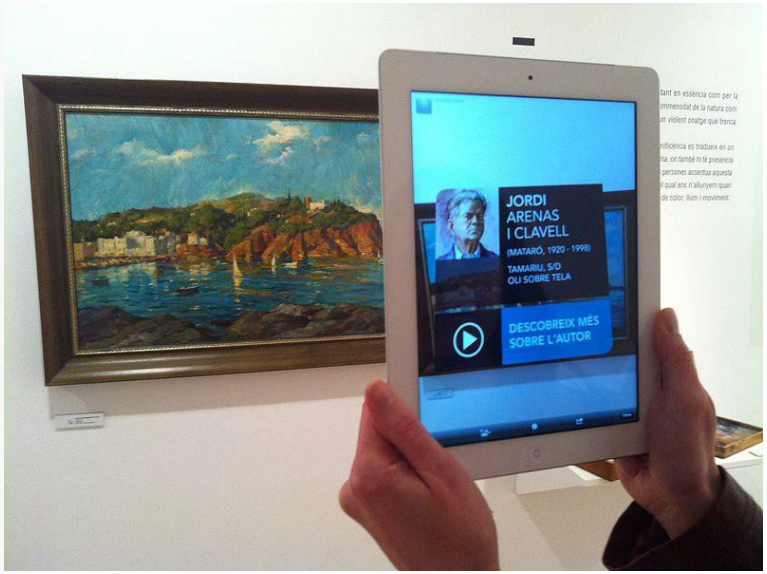
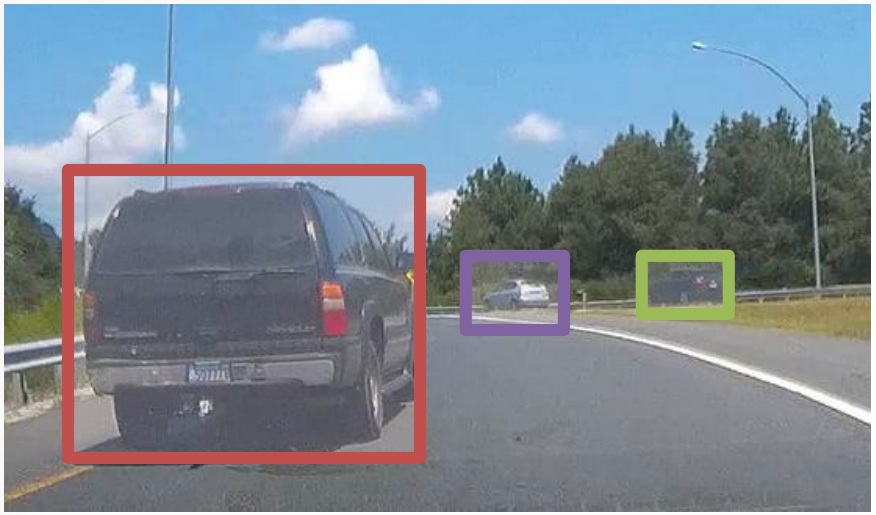


Image by Kippelboy is licensed under CC BY-SA 3.0

Image by Christina C. is licensed under CC BY-SA 4.0

There are many visual recognition problems that are related to image classification, such as

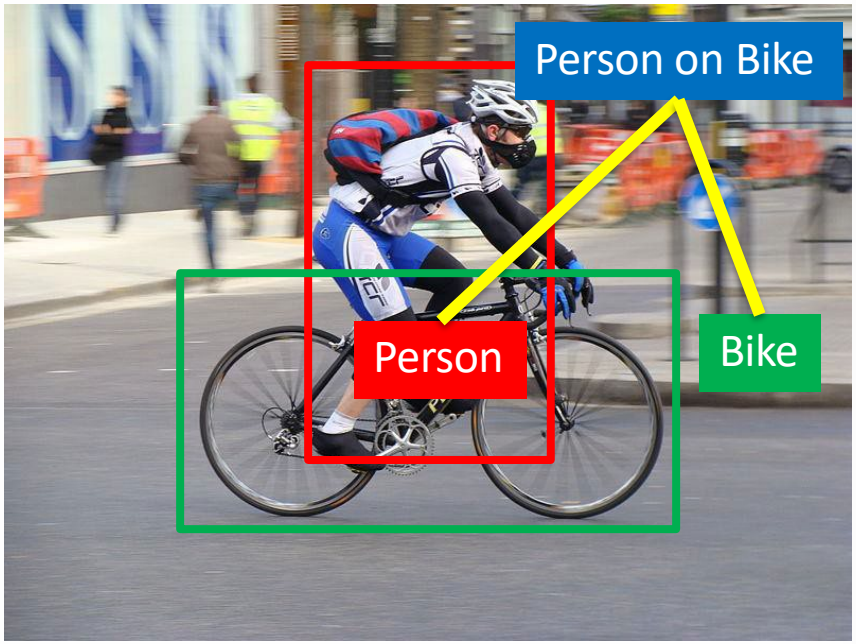
object detection, image captioning



- Object detection
- Action classification
- Image captioning
- …

Person

Hammer



Convolutional Neural Networks (CNN) have become an important tool for object recognition

IMAGENET Large Scale Visual Recognition Challenge

Year 2010

NEC-UIUC

Dense descriptor grid: HOG, LBP

Coding: local coordinate, super-vector

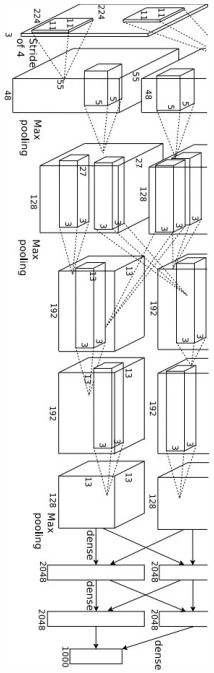
Pooling, SPM

Linear SVM

[Lin CVPR 2011]

Year 2012

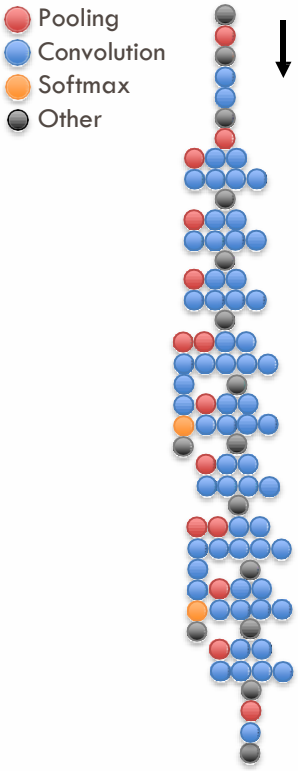
SuperVision



[Krizhevsky NIPS 2012]

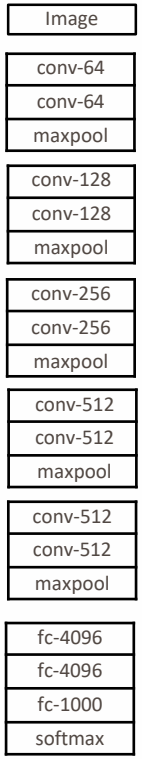
Year 2014

GoogLeNet



[Szegedy arxiv 2014]

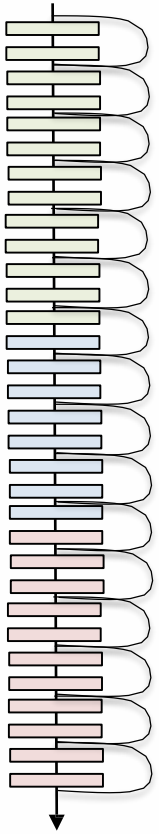
VGG



[Simonyan arxiv 2014]

Year 2015

MSRA



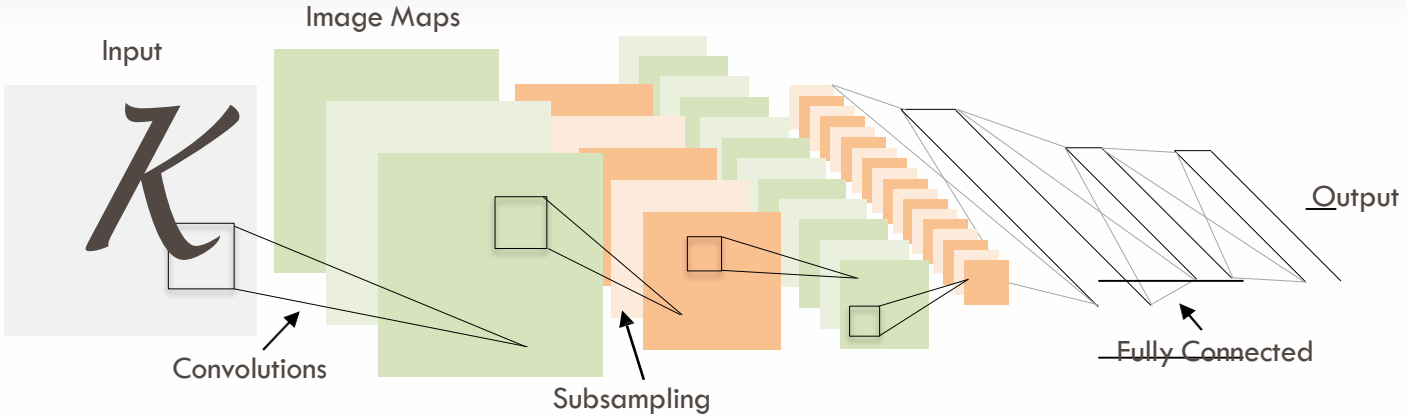
[He ICCV 2015]

Convolutional Neural Networks (CNN)

were not invented overnight

1998

LeCun et al.



# of transistors

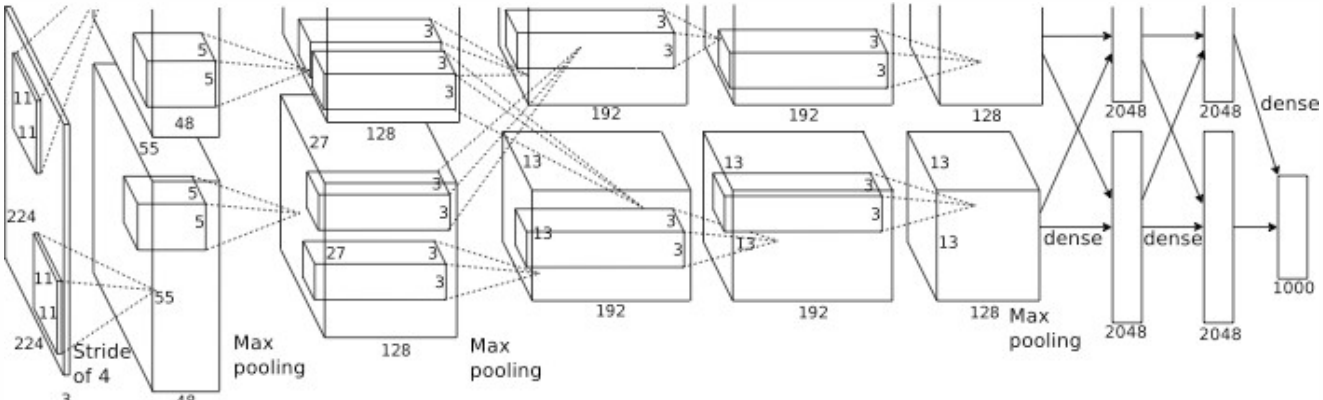
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

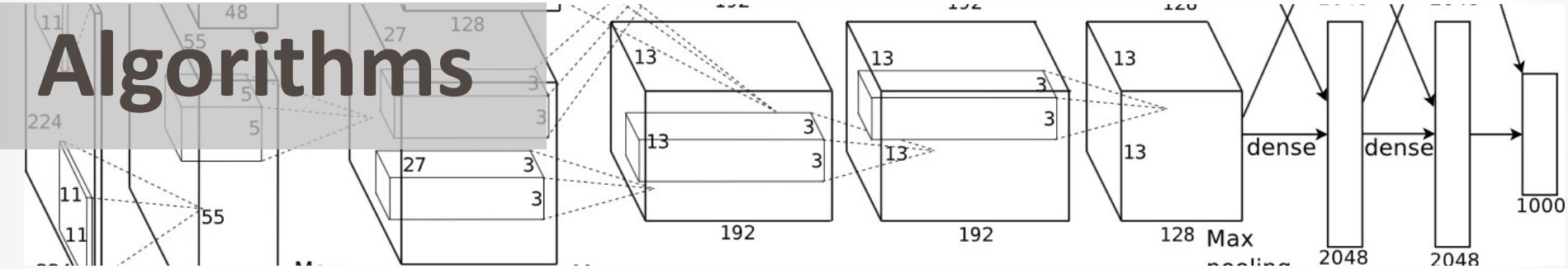
# of pixels used in training

10^{14} IMAGENET

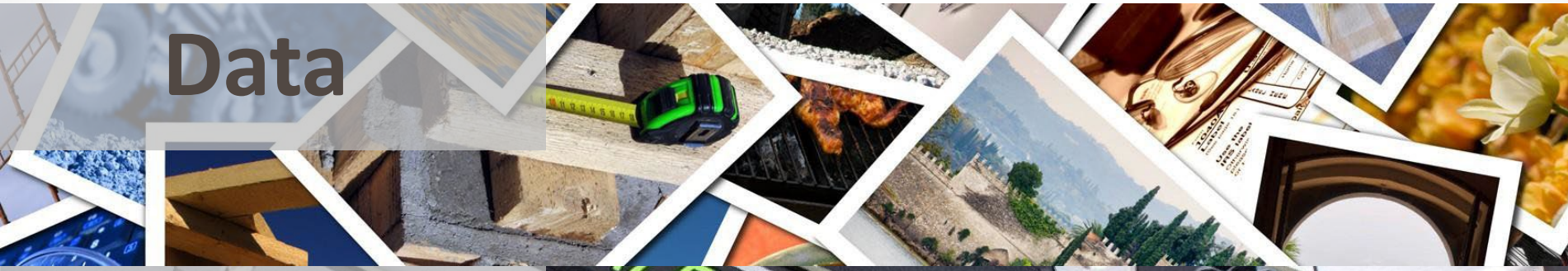
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Ingredients for Deep Learning

Algorithms



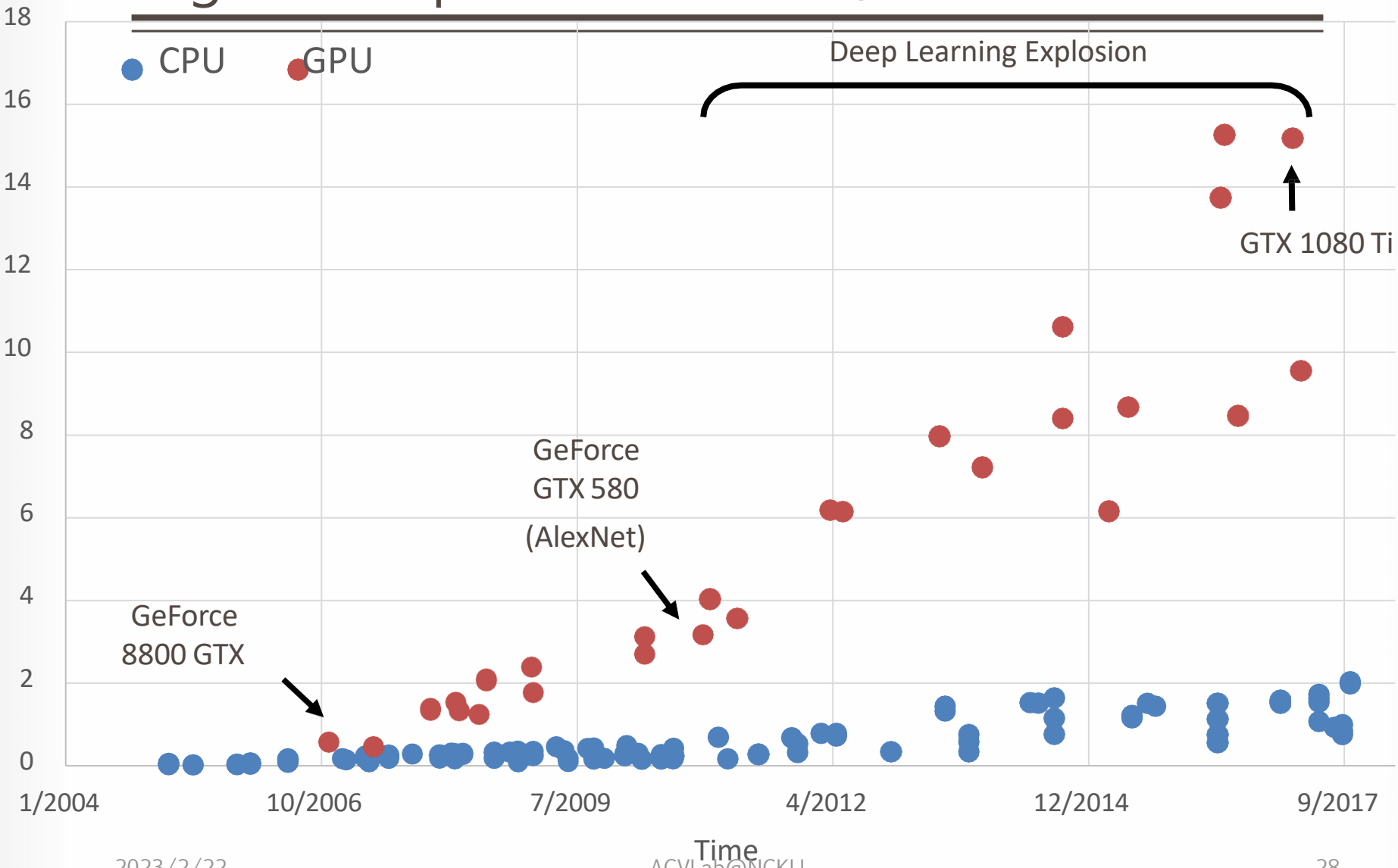
Data



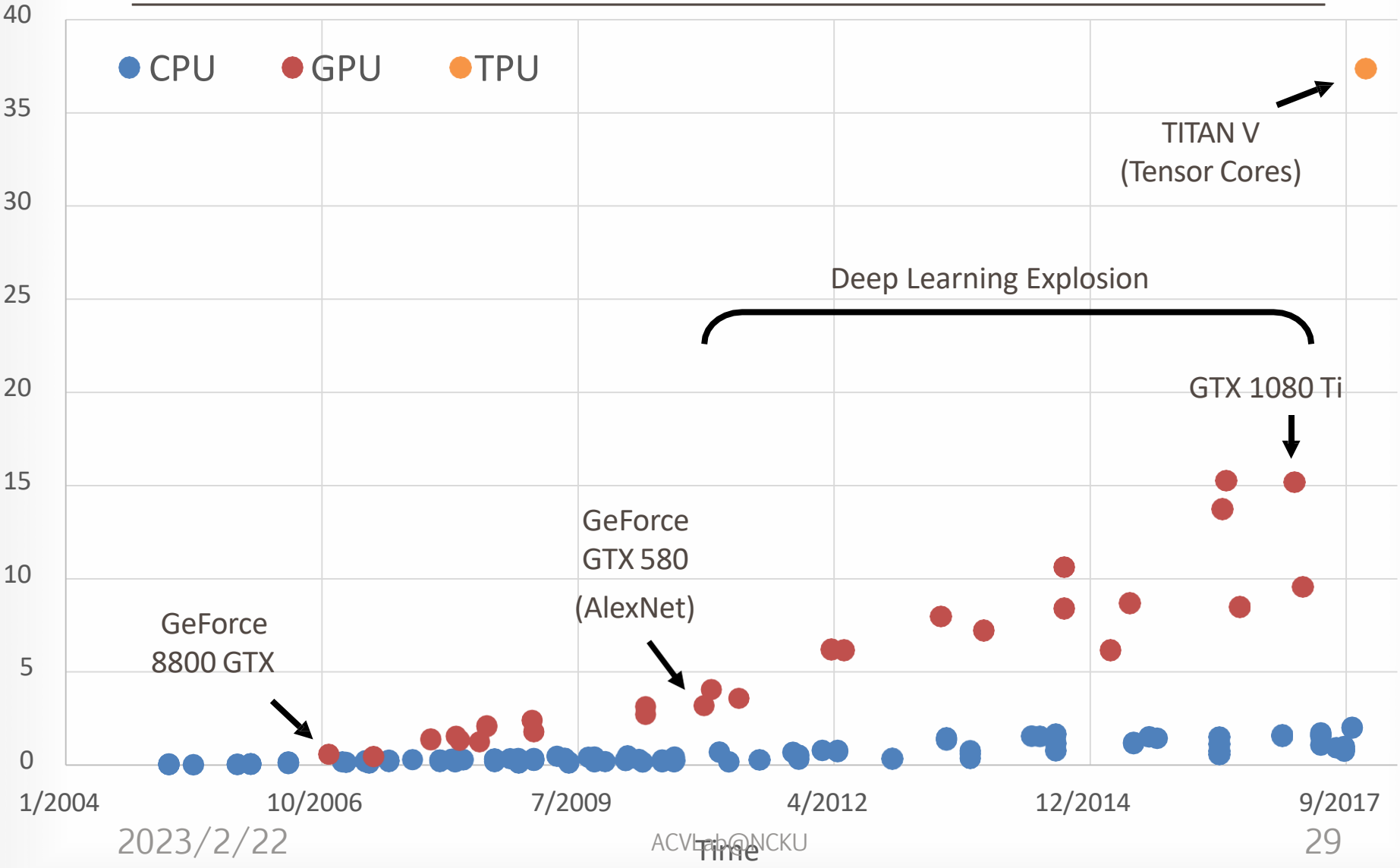
Computation

GigaFLOPs per Dollar

Floating-point operations per second

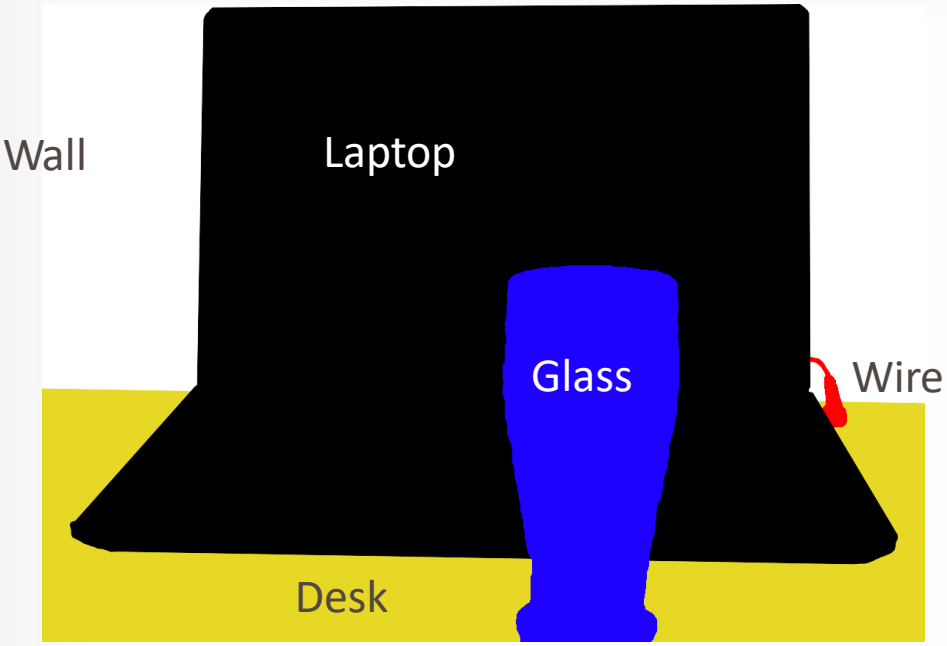


GigaFLOPs per Dollar



The quest for visual intelligence
goes far beyond object recognition…

Image is CC0 1.0 public domain



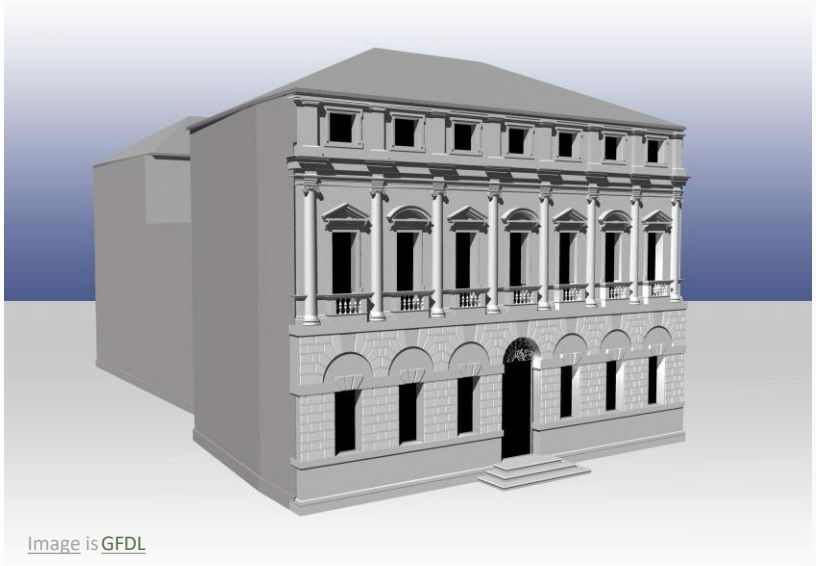


Image is GFDL

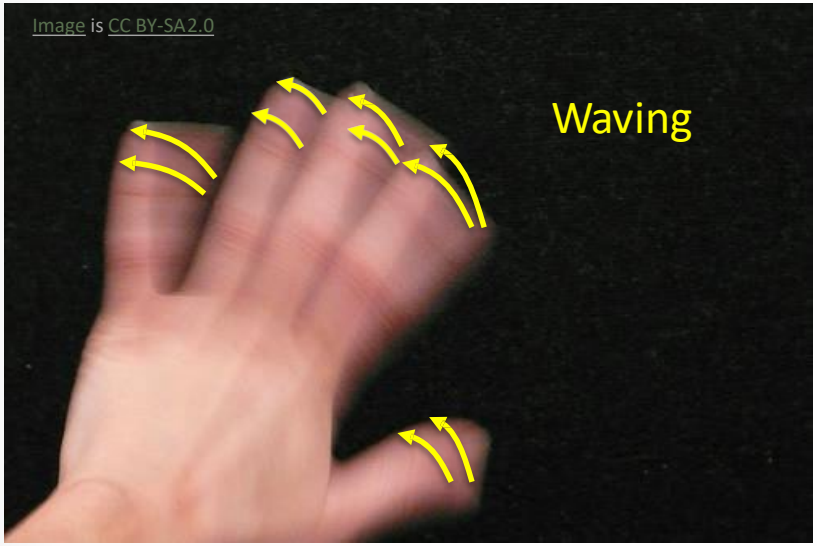
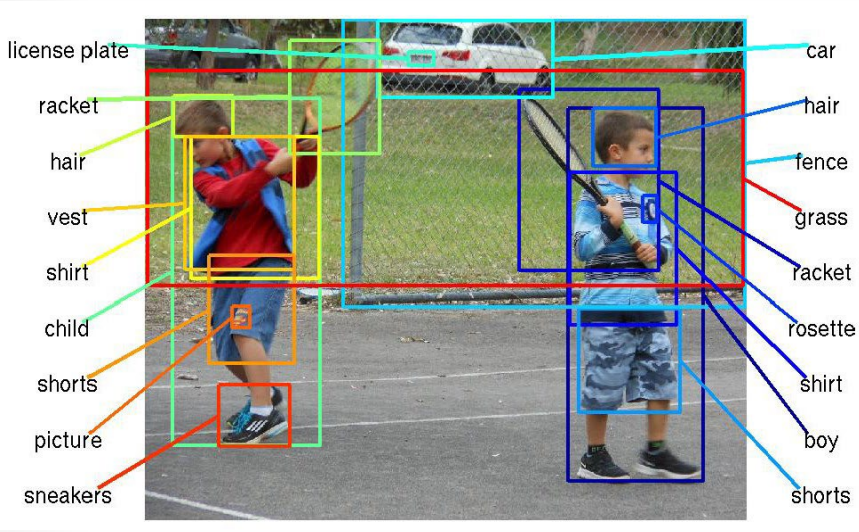
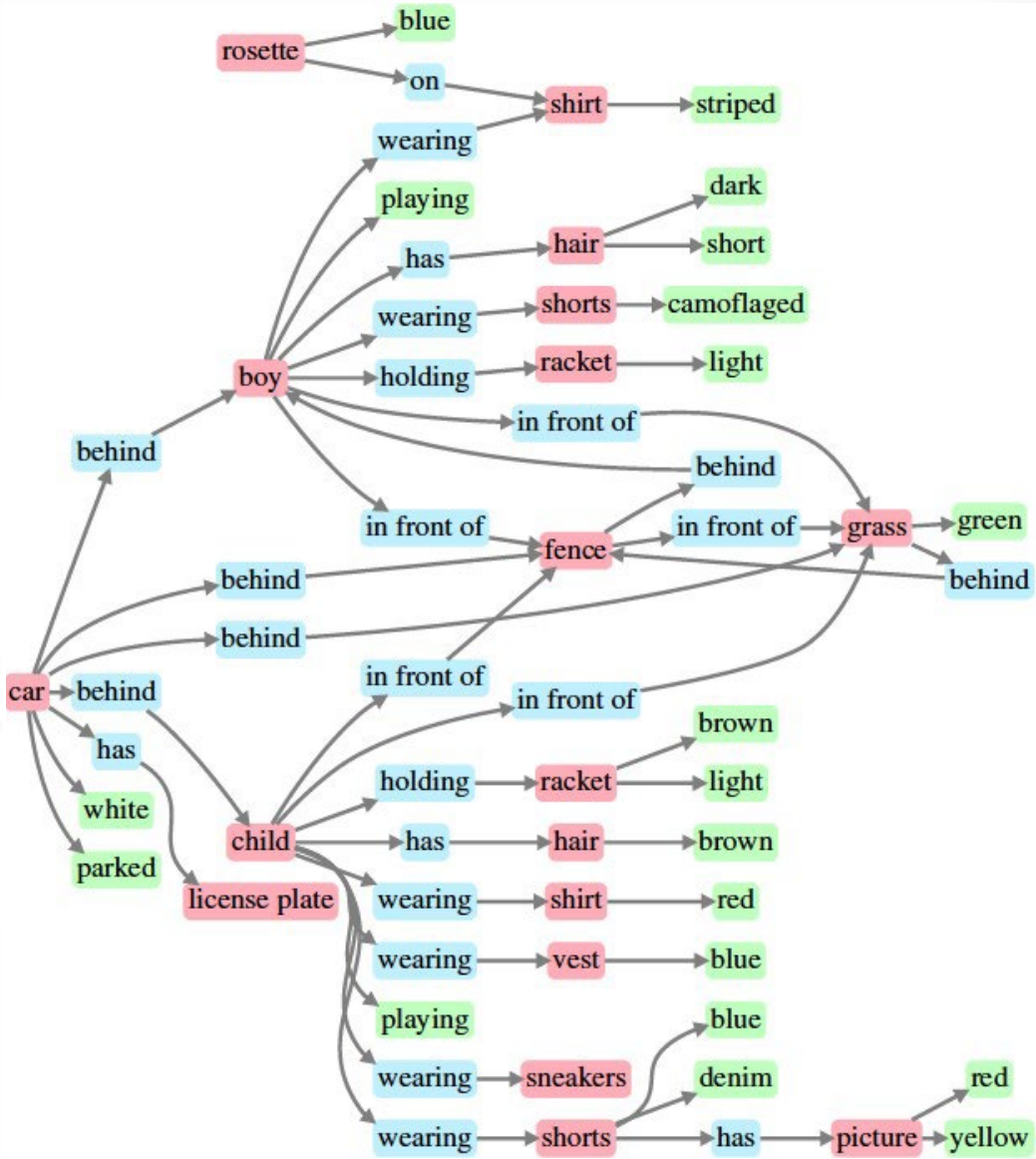


Image is CC BY-SA2.0

Image is CC BY-SA4.0

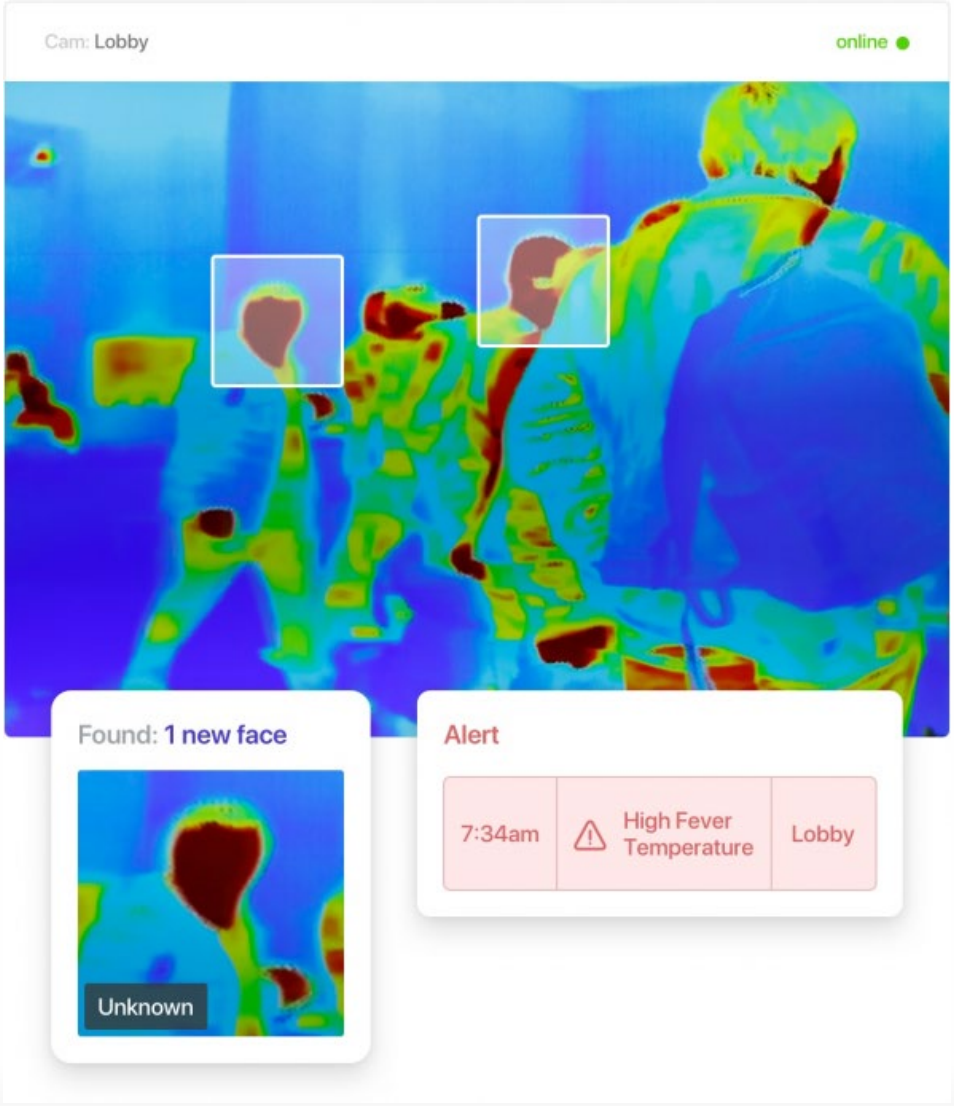




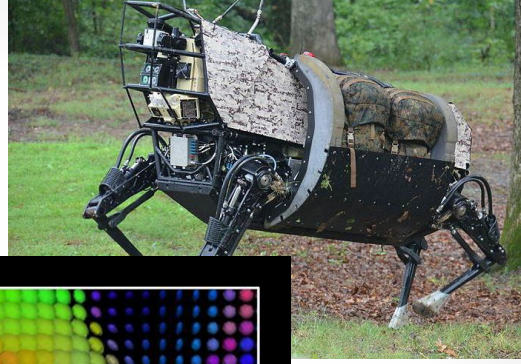
Johnson *et al.*, “Image Retrieval using Scene Graphs”, CVPR 2015

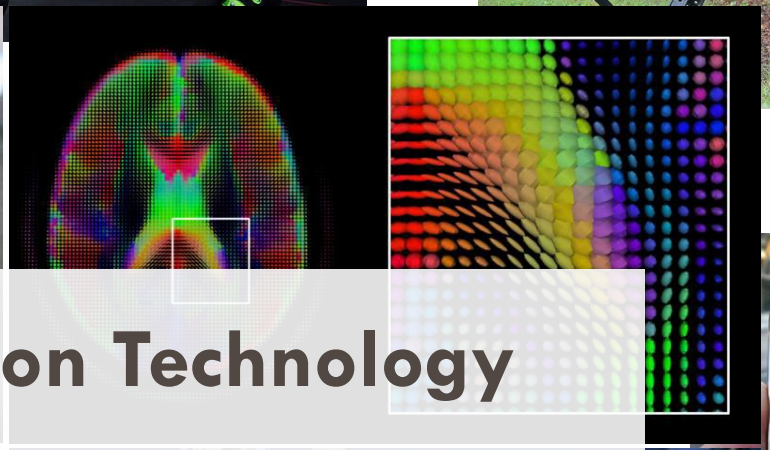
Example credit: Andrej Karpathy

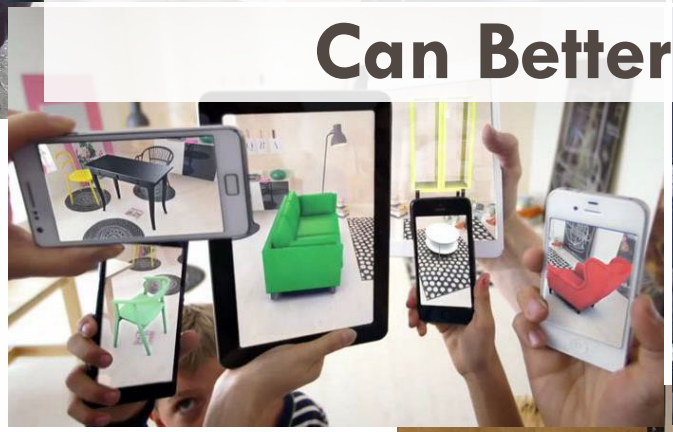
Face Recognition of NIR Images

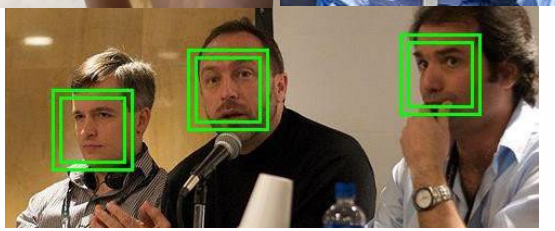










Computer Vision Technology Can Better Our Lives

HOW?

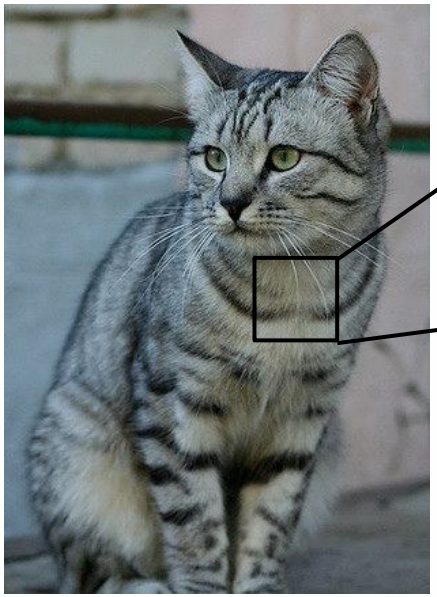
IMAGE CLASSIFICATION PIPELINE

Image Classification: A core task in Computer Vision

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

cat

The Problem: Semantic Gap



This image by Nikita is licensed under CC-BY 2.0

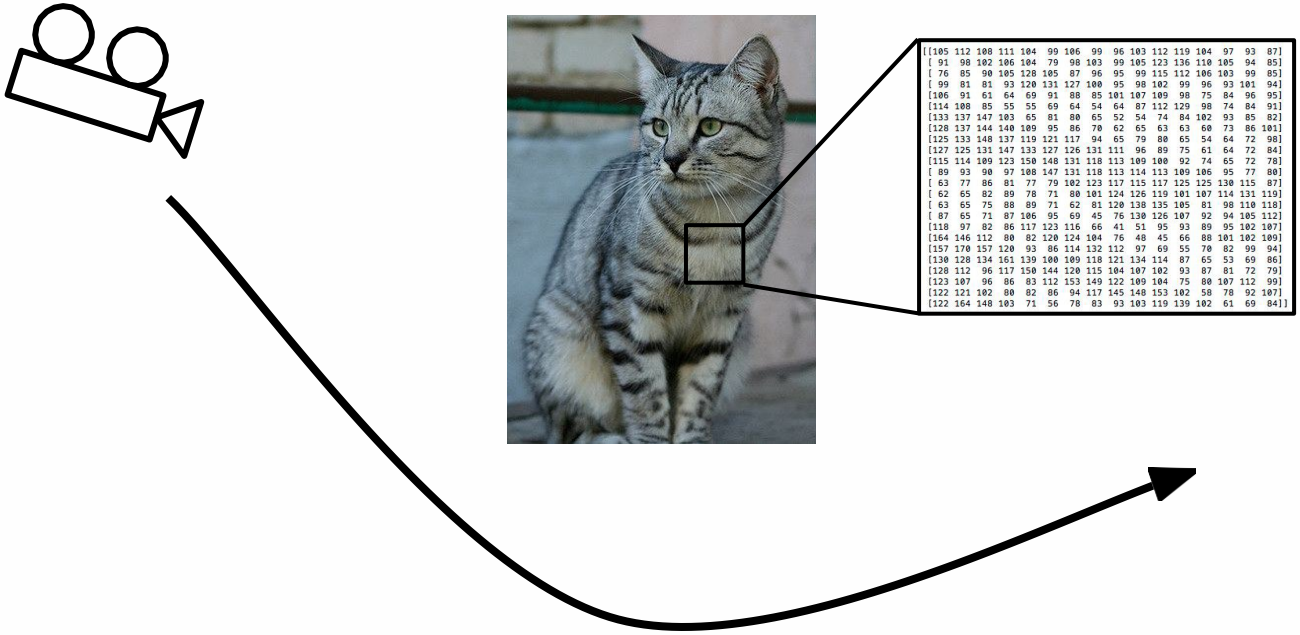
```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  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  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  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]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  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  98 110 118]
 [ 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  93  89  95 102 107]
 [164 146 112  80  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  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  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  94 117 145 148 153 102  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)

Challenges: Viewpoint variation



All pixels change when the camera moves!

Challenges: Background Clutter

This image is CC0 1.0 public domain

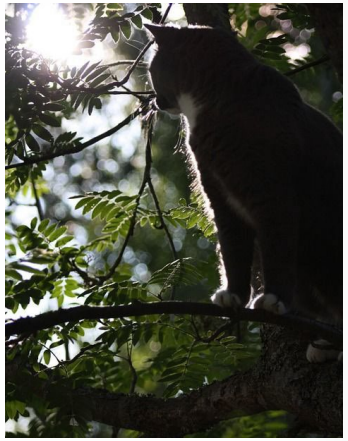
This image is CC0 1.0 public domain

Challenges: Illumination

This image is CC0 1.0 public domain

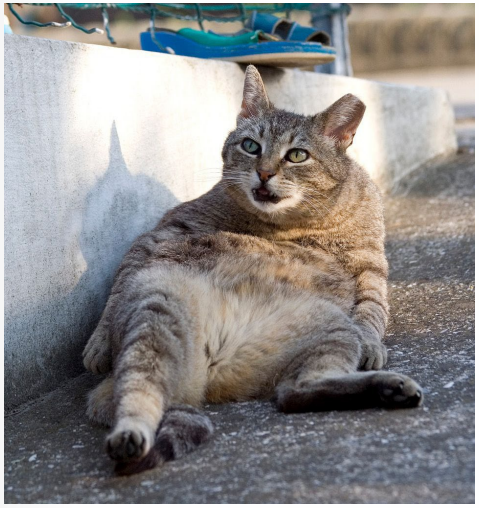
This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

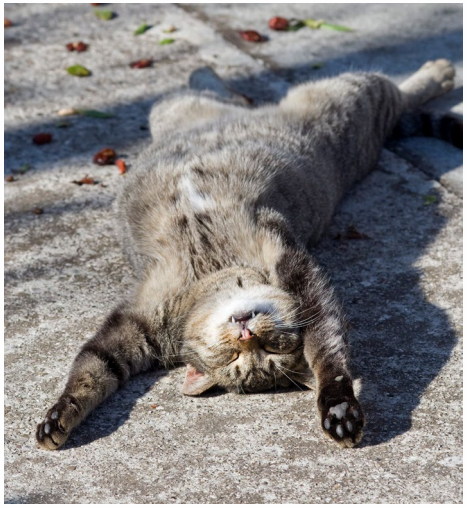


This image is CC0 1.0 public domain

Challenges: Deformation

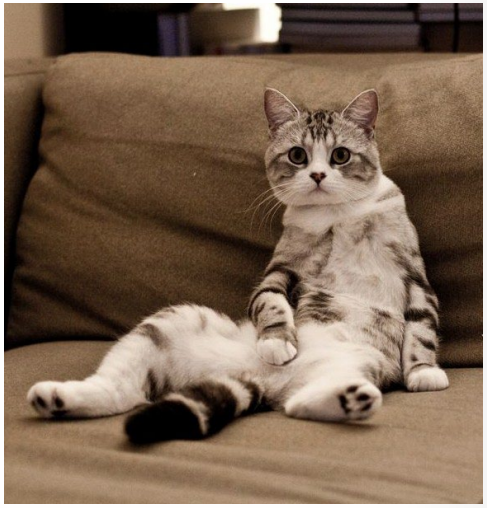


This image by Umberto Salvagnin is licensed under CC-BY 2.0



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 Thai is licensed under CC-BY 2.0

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

Challenges: Intraclass variation

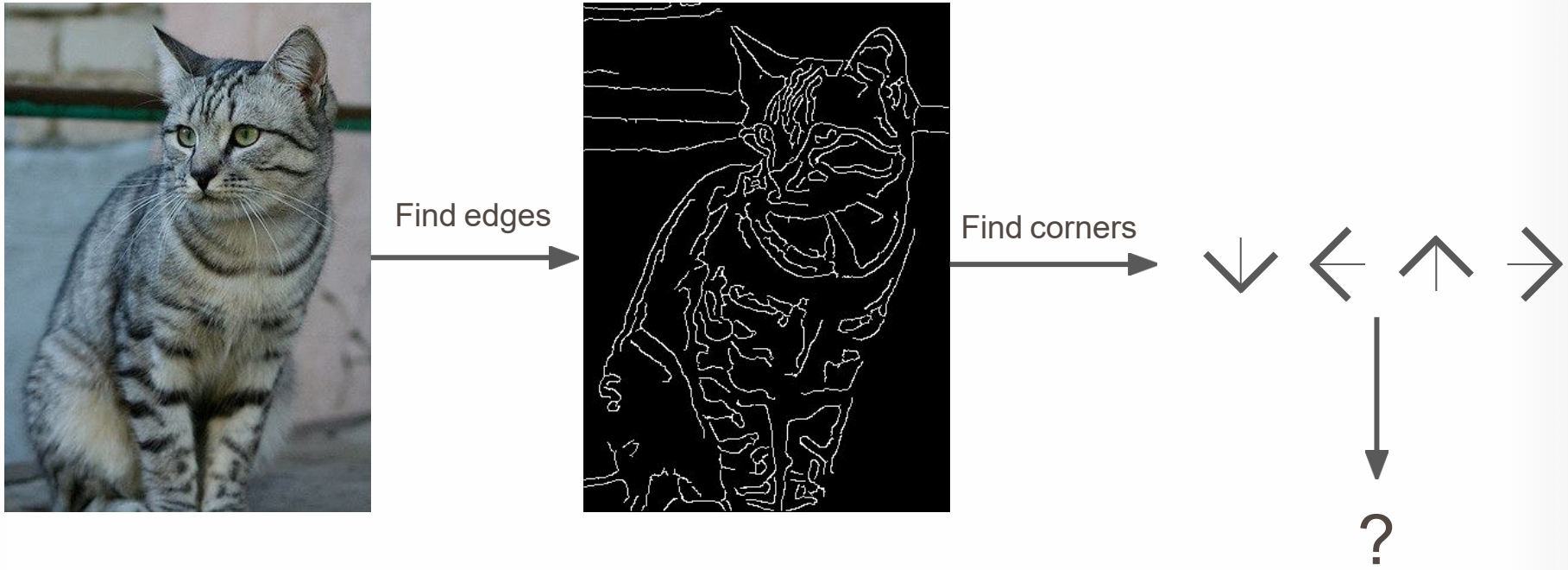
An image classifier

```
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.

Attempts have been made



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

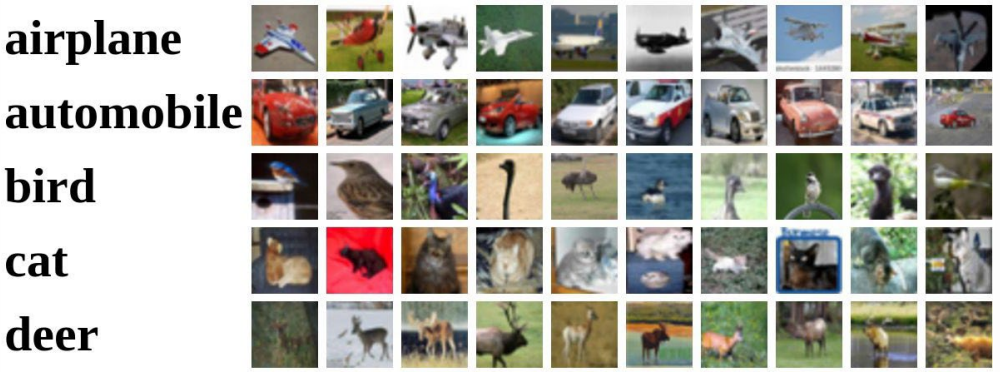
Machine Learning: Data-Driven Approach

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

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

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

Example training set



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

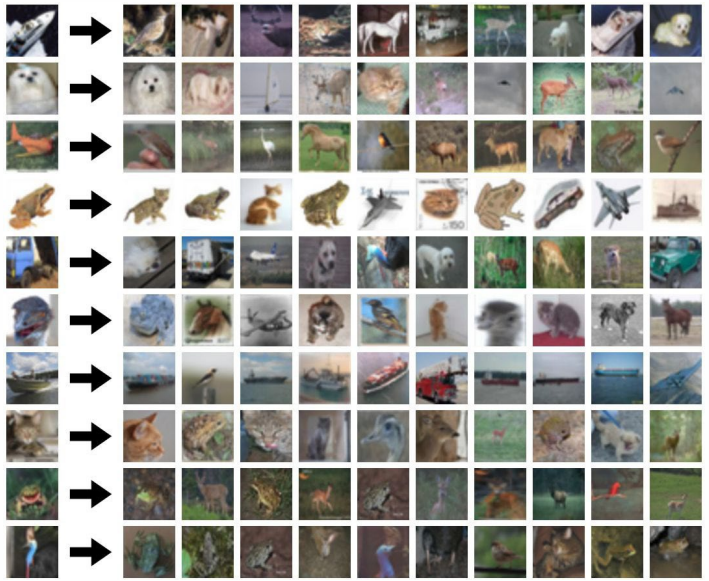
10 classes
50,000 training images
10,000 testing images

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

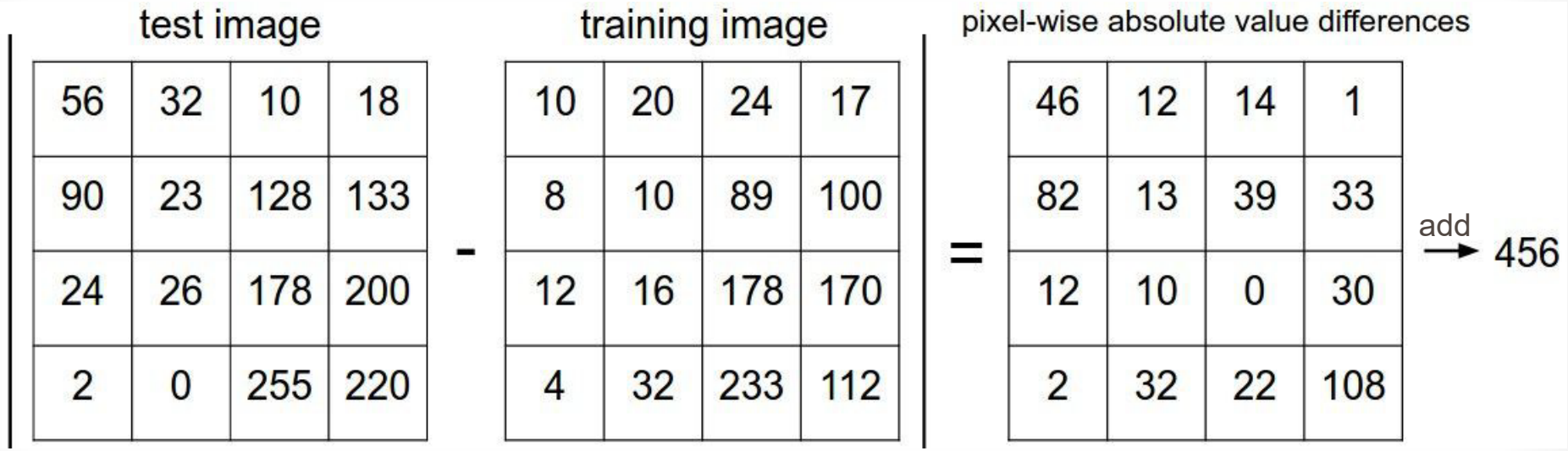
Test images and nearest neighbors



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

Distance Metric to compare images

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



Nearest Neighbor classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

Memorize training data

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

For each test image:
Find closest train image
Predict label of nearest image

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

A: Train O(1),
predict O(N)

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

Q: With N examples, how fast are training and prediction?

A: Train O(1),
predict O(N)

This is bad: we want classifiers that are **fast** at prediction; **slow** for training is ok

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

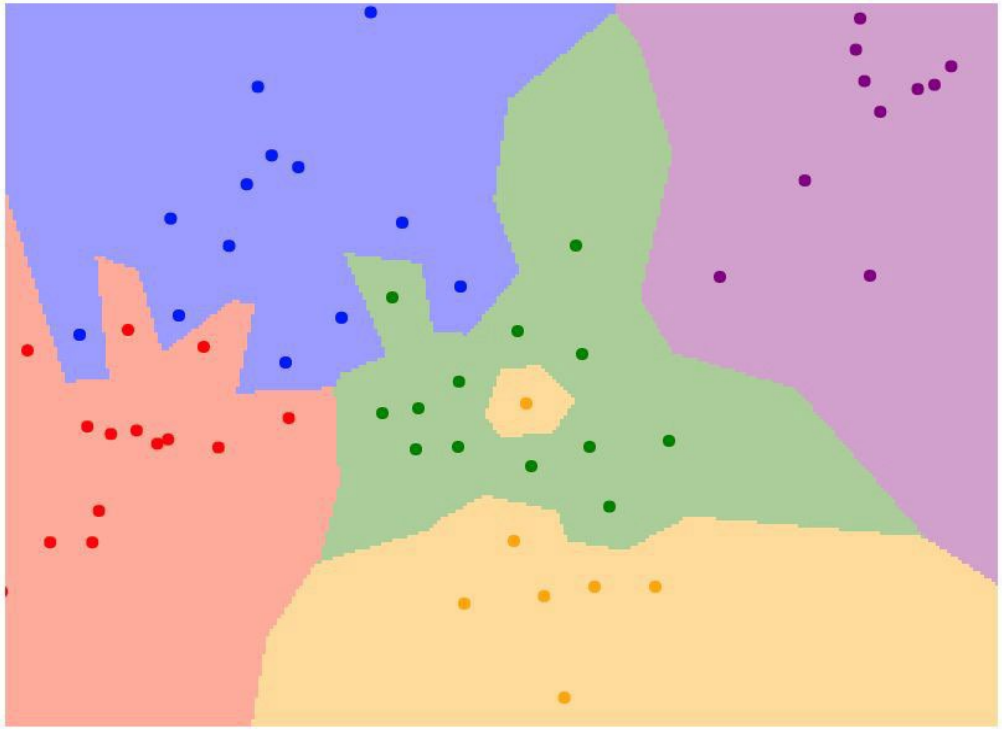
Nearest Neighbor classifier

Many methods exist for fast / approximate nearest neighbor

A good implementation:
https://github.com/facebookresearch/faiss

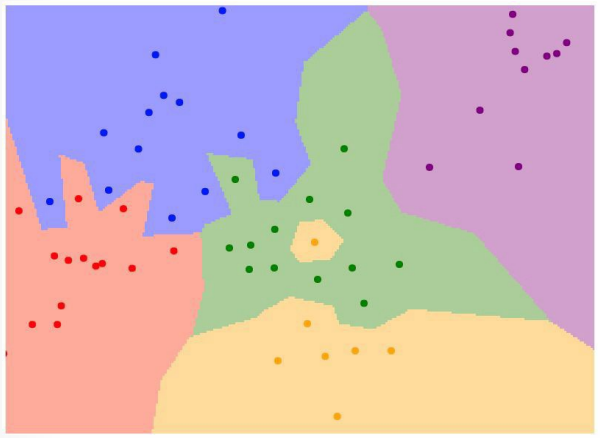
Johnson et al, “Billion-scale similarity search with GPUs”, arXiv 2017

What does this look like?

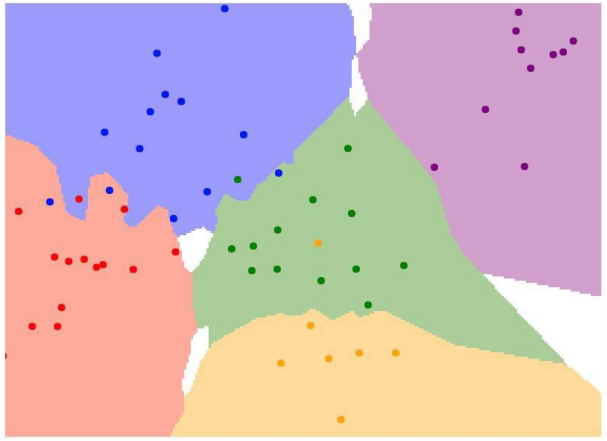


K-Nearest Neighbors

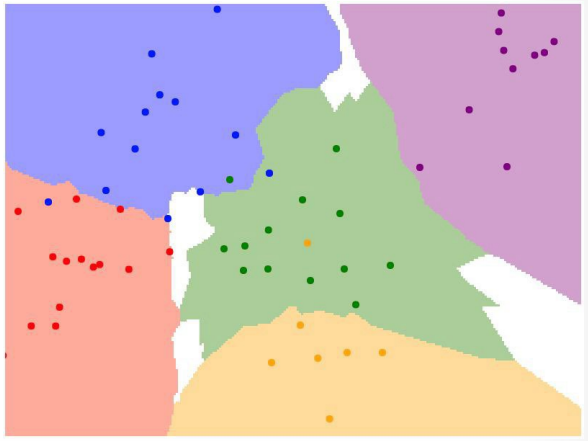
Instead of copying label from nearest neighbor,
take **majority vote** from K closest points



K = 1

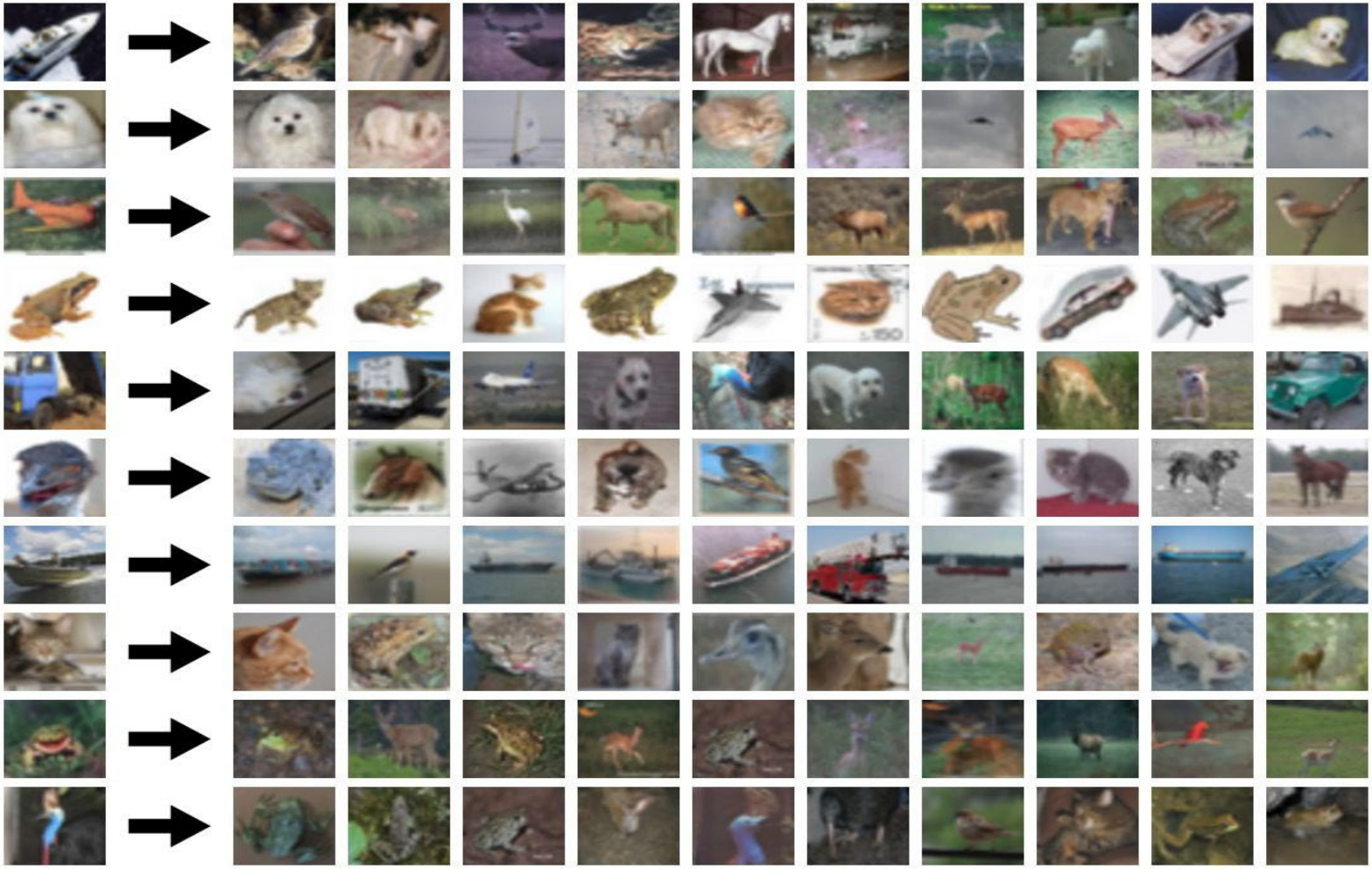


K = 3

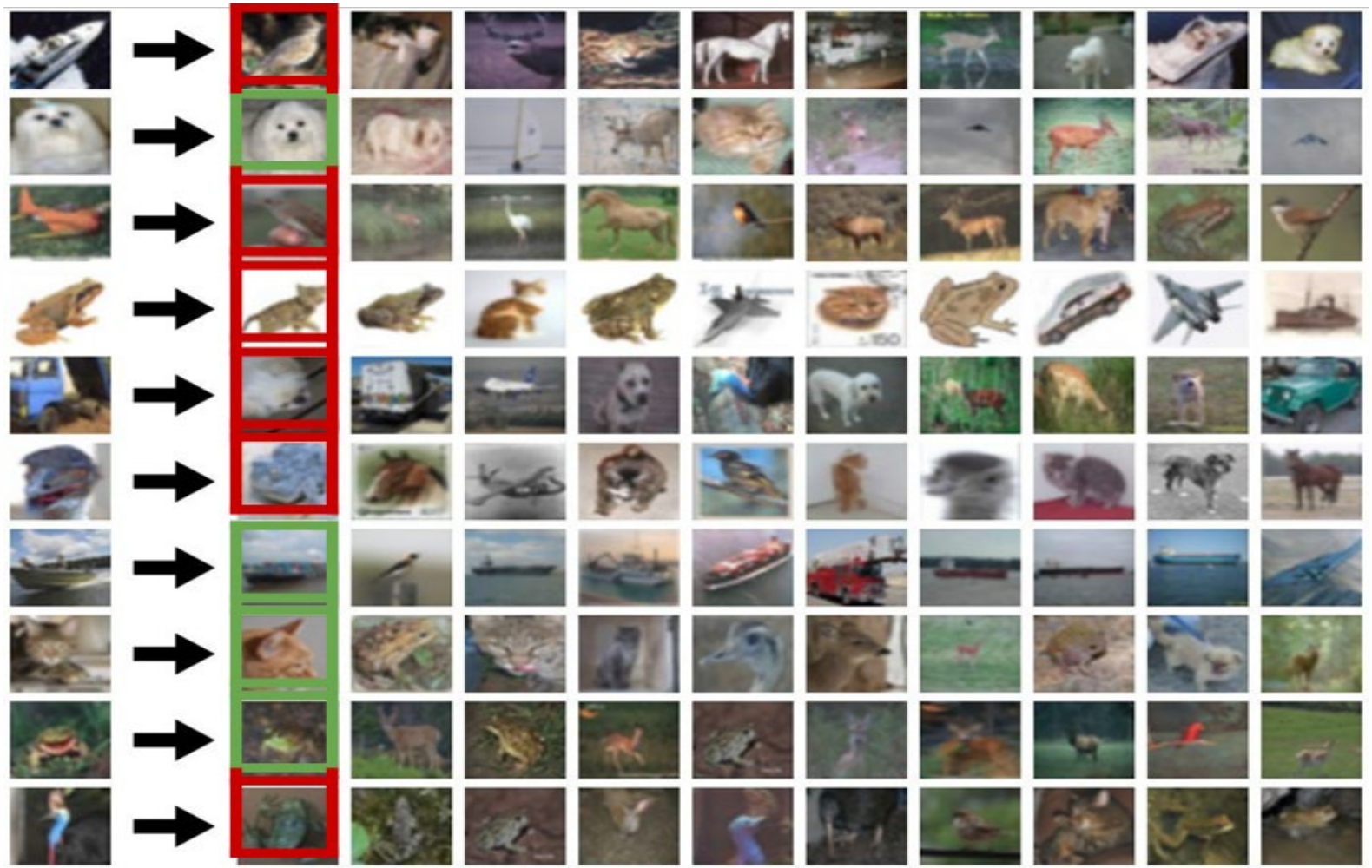


K = 5

What does this look like?



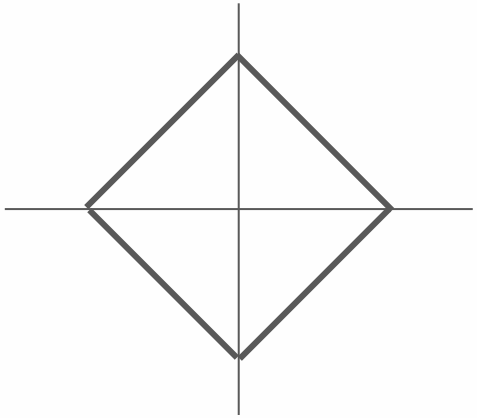
What does this look like?



K-Nearest Neighbors: Distance Metric

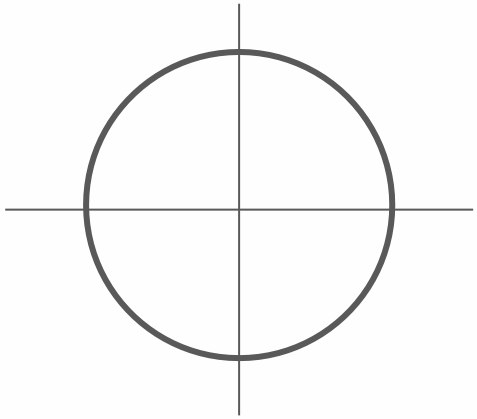
L1 (Manhattan) distance

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



L2 (Euclidean) distance

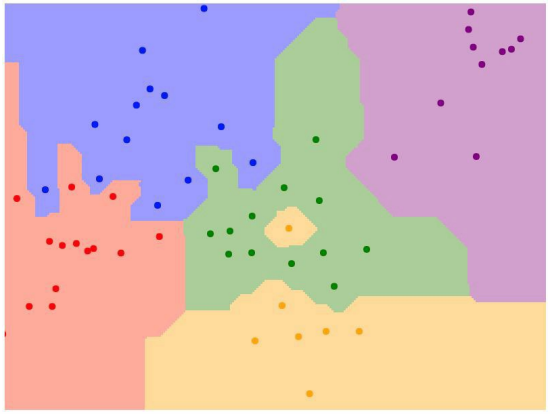
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$



K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

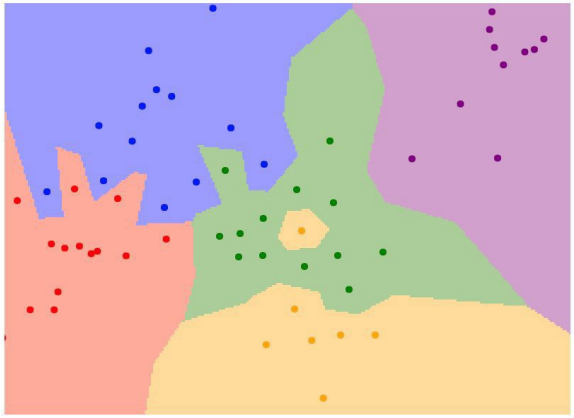
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

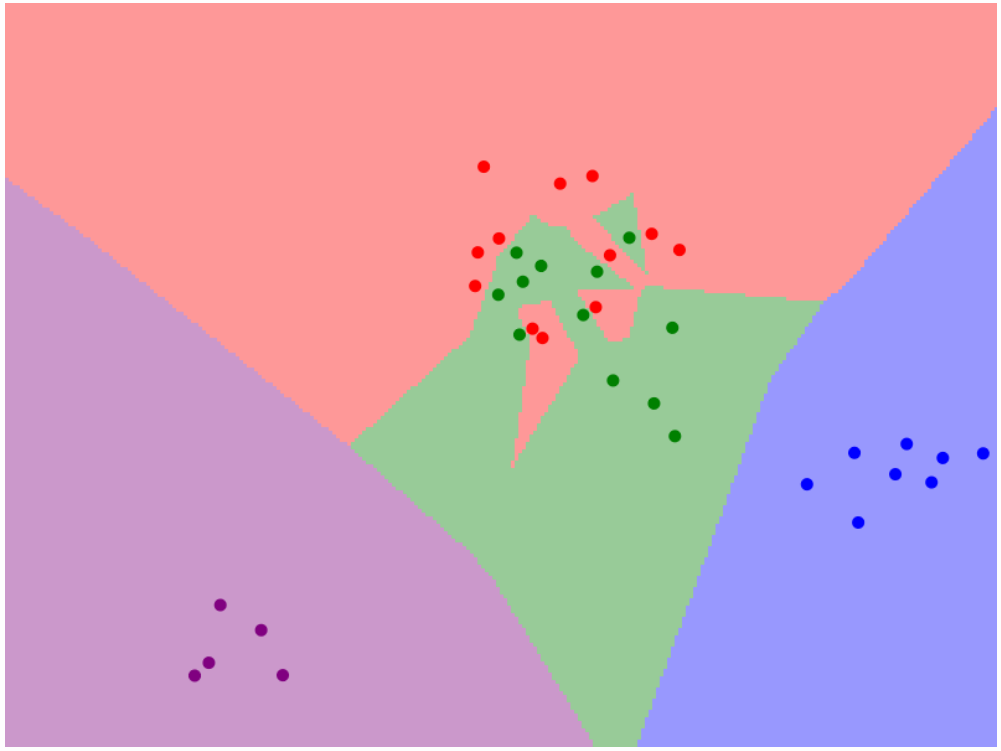
L2 (Euclidean) distance

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



K = 1

K-Nearest Neighbors: Demo Time



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

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

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: Evaluation

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

Your Dataset

Setting Hyperparameters: Evaluation

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

BAD: K = 1 always works perfectly on training data

Setting Hyperparameters: Evaluation

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

Setting Hyperparameters: Evaluation

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

Setting Hyperparameters: Evaluation

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!

Setting Hyperparameters: Evaluation

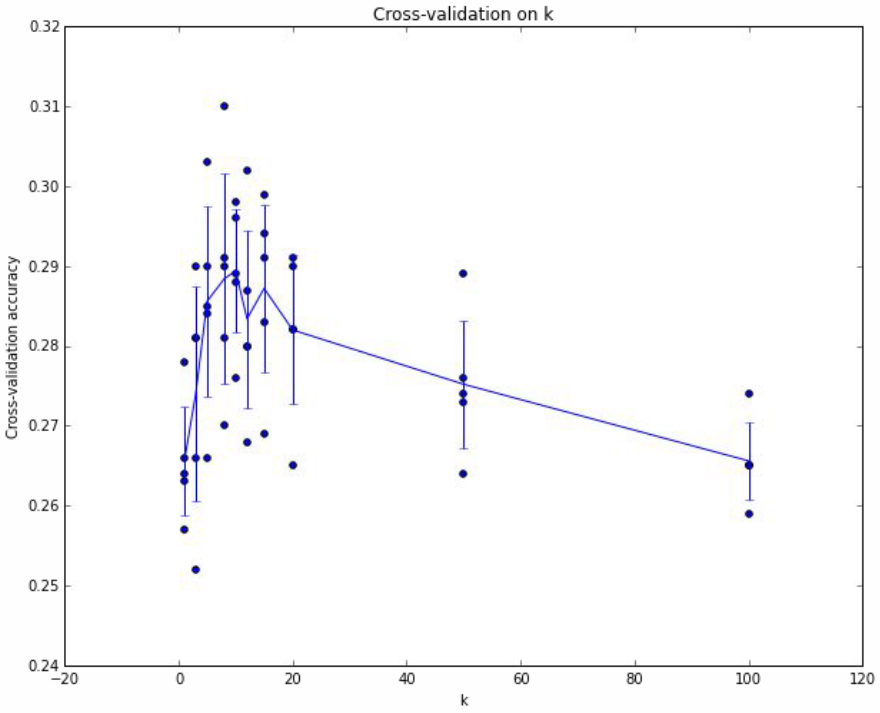
Your Dataset

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: Evaluation



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

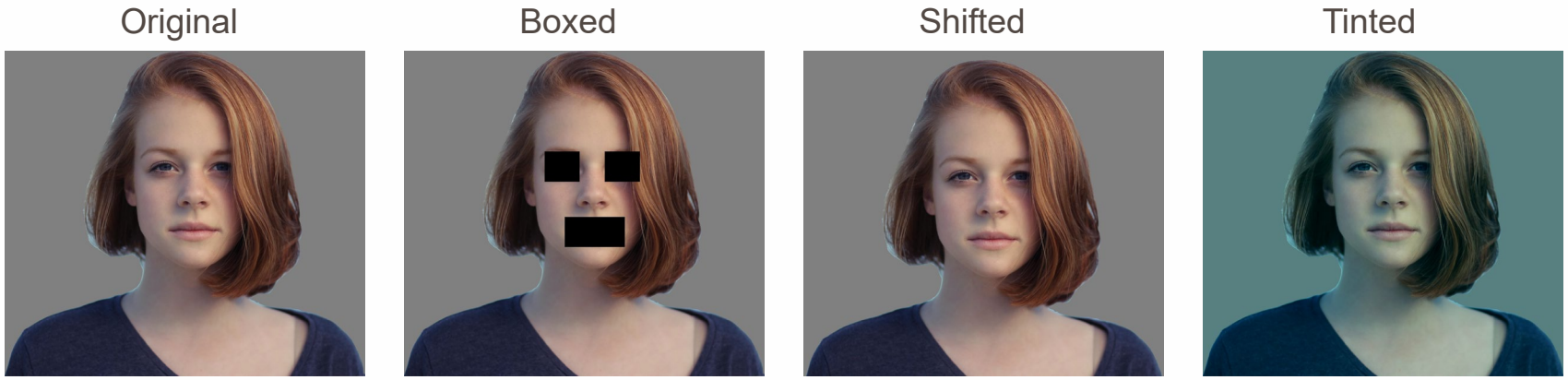
Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that k ~= 7 works best for this data)

k-Nearest Neighbor on images never used.

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



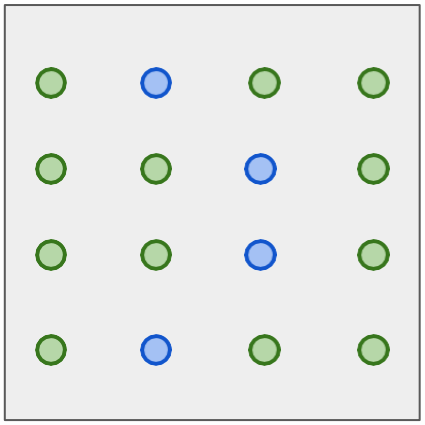
(all 3 images have same L2 distance to the one on the left)

k-Nearest Neighbor on images never used.

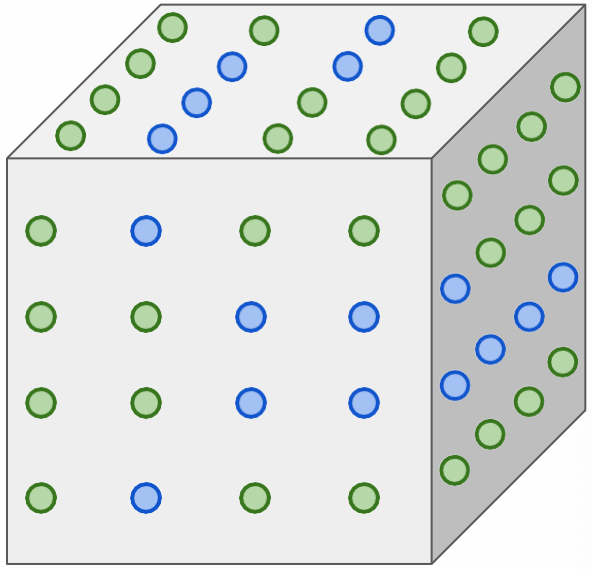
- Curse of dimensionality

Dimensions = 1
Points = 4

Dimensions = 2
Points = 4^2



Dimensions = 3
Points = 4^3



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

Neural Network

Linear classifiers

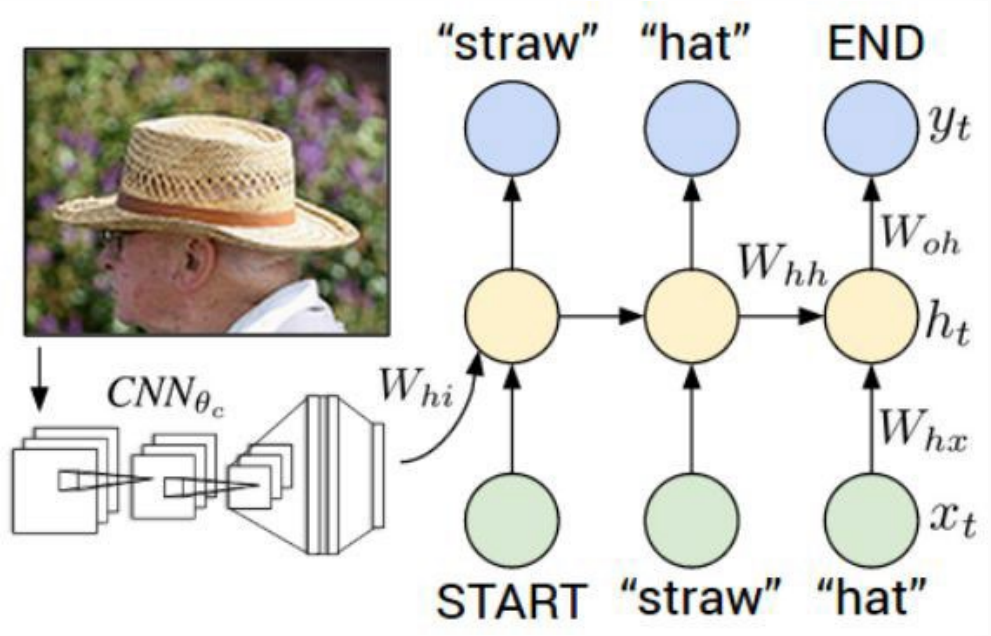
This image is CC0 1.0 public domain

Two young girls are playing with lego toy.

Boy is doing backflip on wakeboard

Man in black shirt is playing guitar.

Construction worker in orange safety vest is working on road.



Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

Recall CIFAR10

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

10,000 test images.

Parametric Approach

Image

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

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

10 numbers giving
class scores

W

parameters
or weights

Parametric Approach: Linear Classifier

Image

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

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

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

10 numbers giving
class scores

W

parameters
or weights

Parametric Approach: Linear Classifier

Image

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

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

3072x1

10x1 **10x3072**

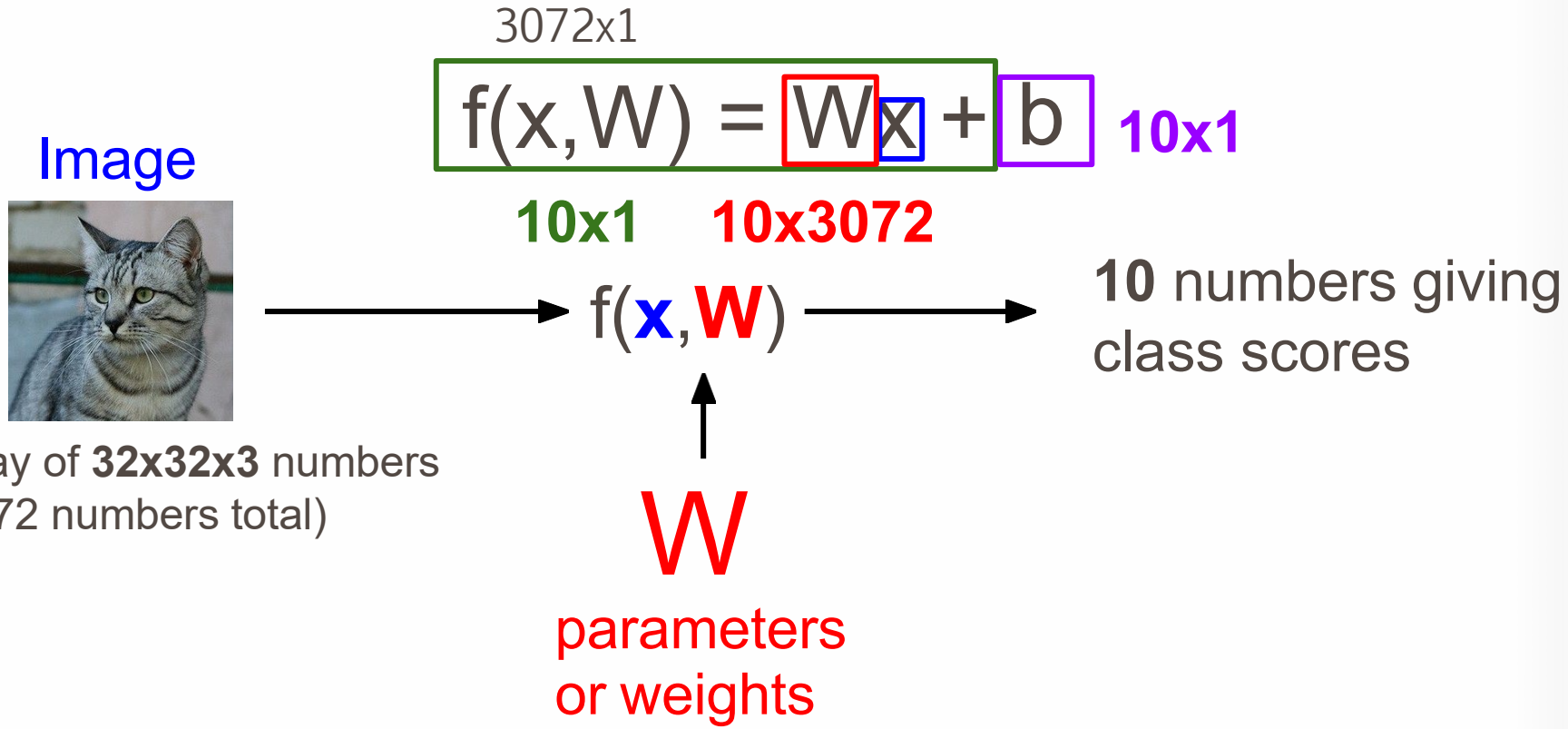
$f(\mathbf{x}, \mathbf{W})$

10 numbers giving class scores

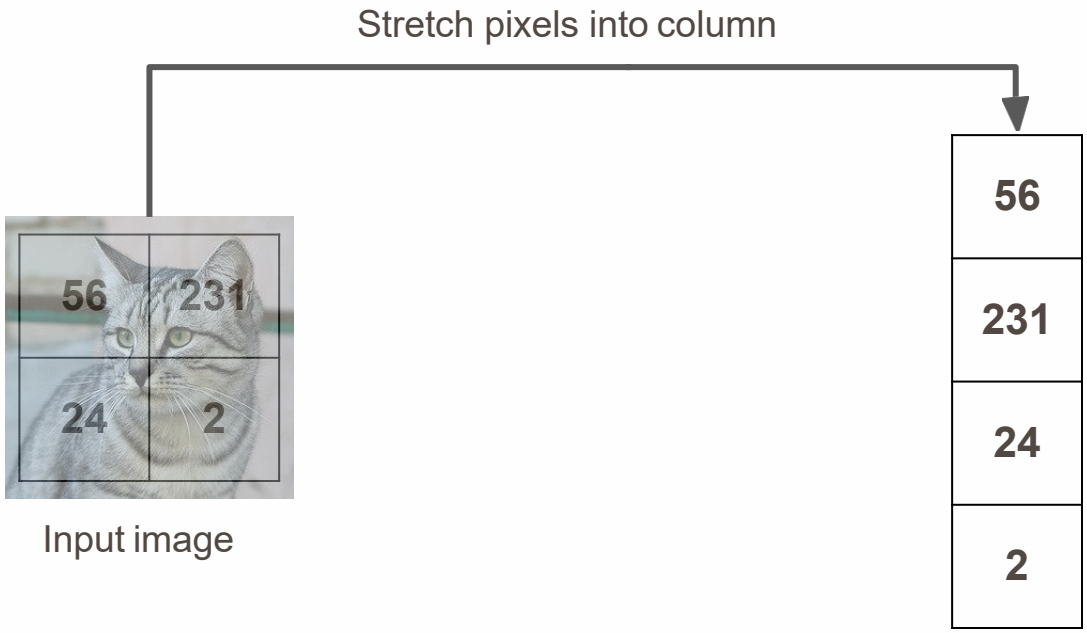
W

parameters or weights

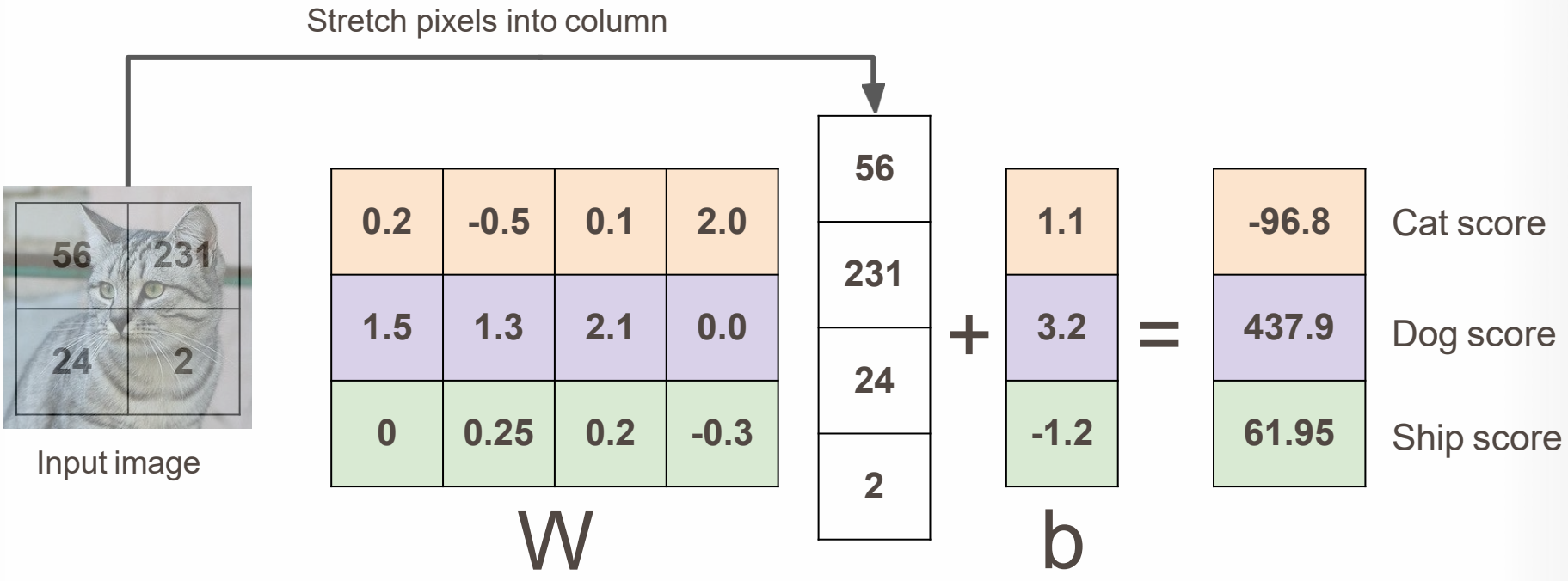
Parametric Approach: Linear Classifier



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



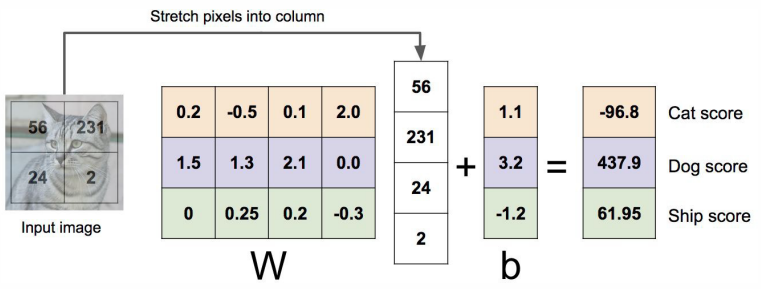
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

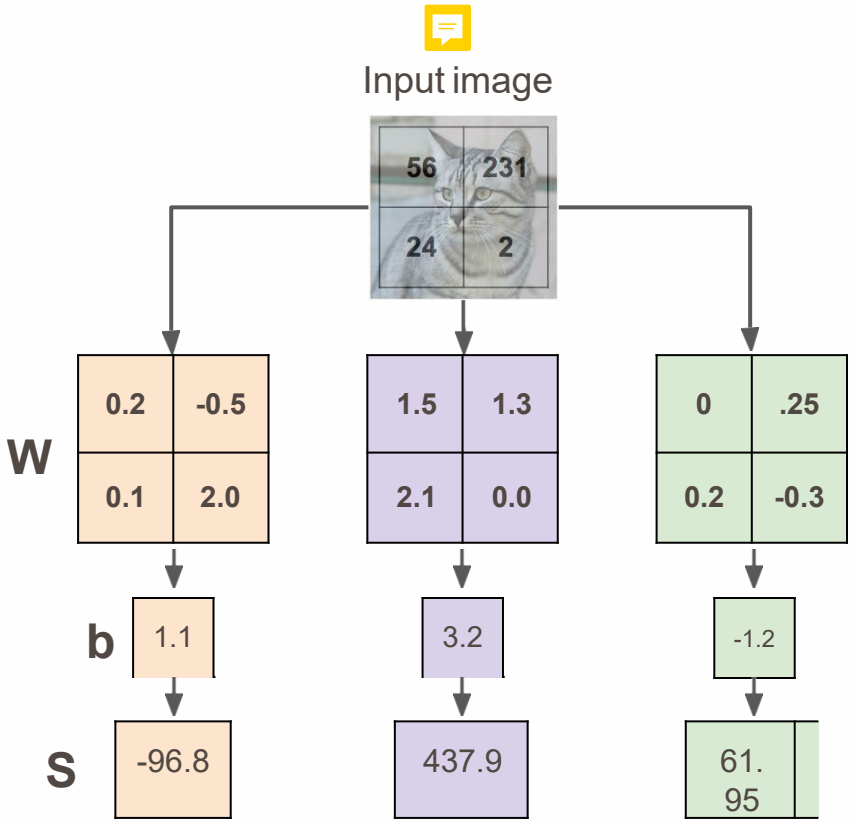


Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

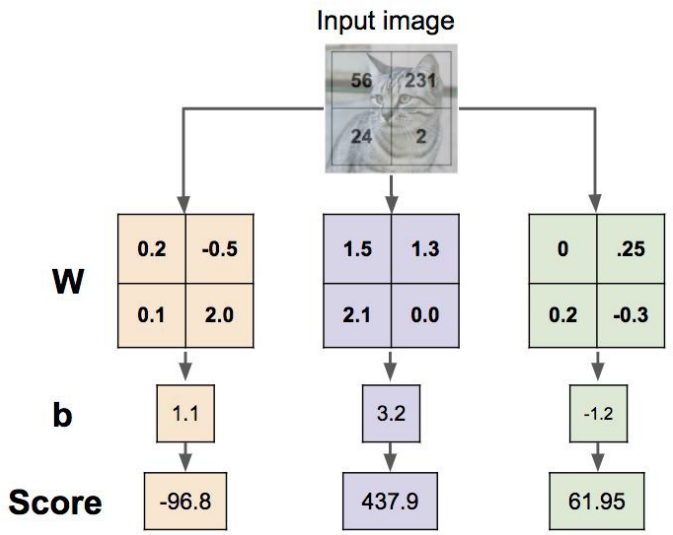
Algebraic Viewpoint

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

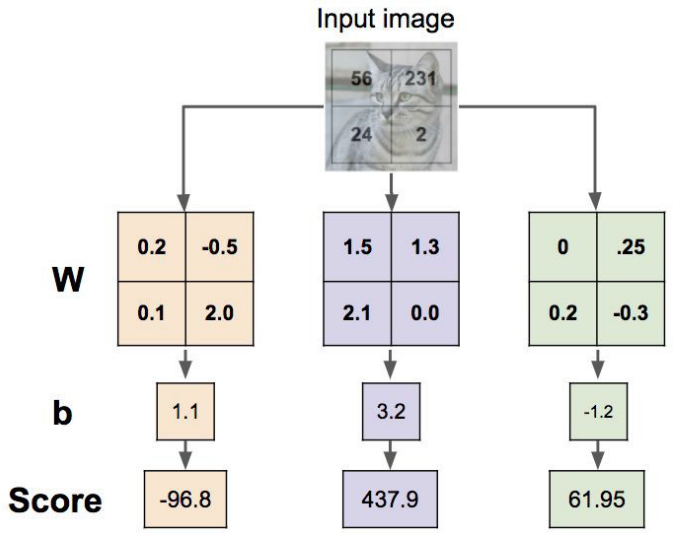




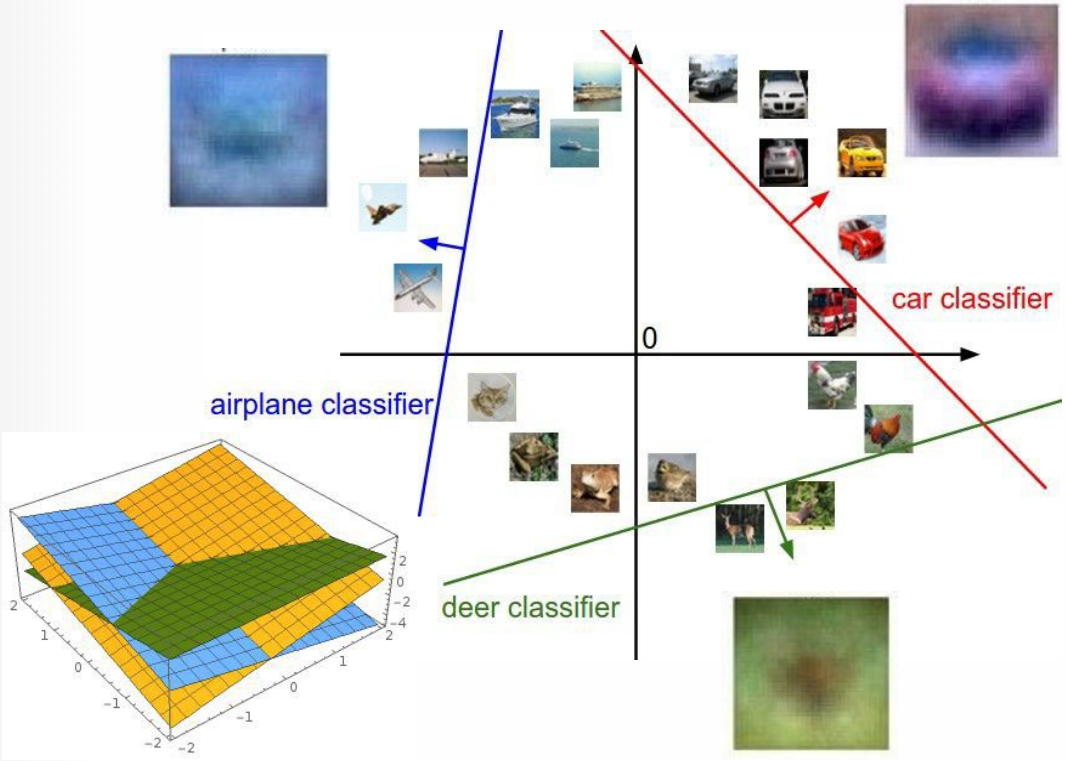
Interpreting a Linear Classifier



Interpreting a Linear Classifier: Visual Viewpoint



Interpreting a Linear Classifier: Geometric Viewpoint



Plot created using Wolfram Cloud

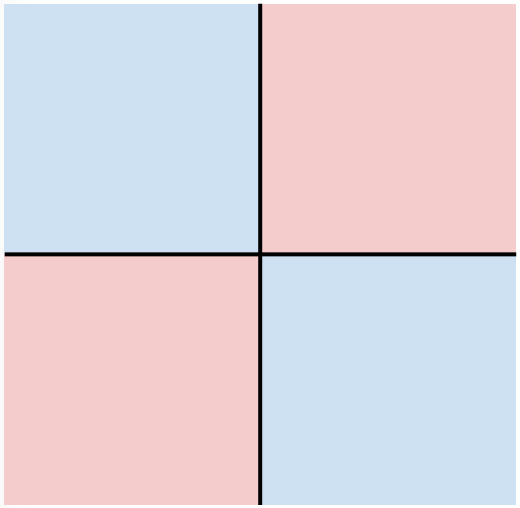
$$f(x,W) = Wx + b$$

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

Hard cases for a linear classifier

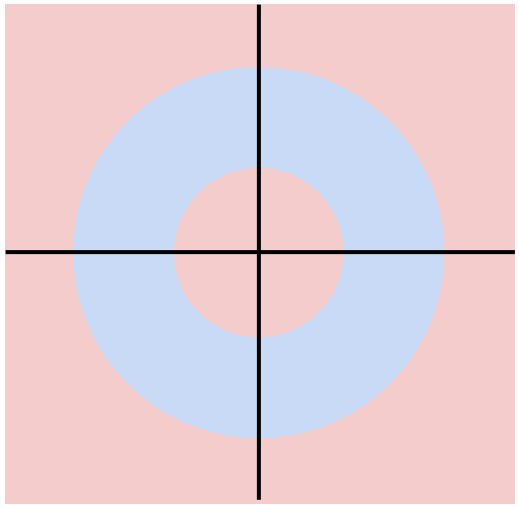
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



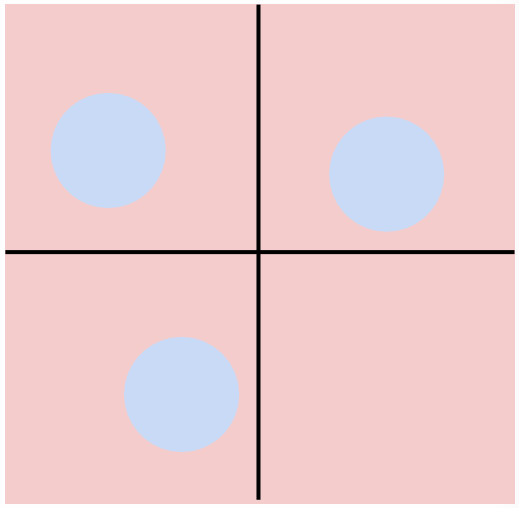
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

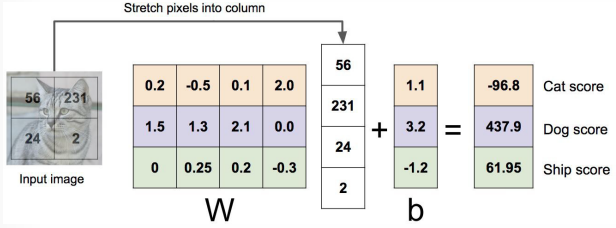
Class 2:
Everything else



Linear Classifier: Three Viewpoints

Algebraic Viewpoint

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

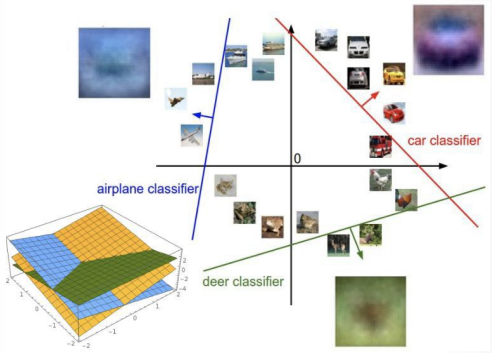


Visual Viewpoint

One template per class

Geometric Viewpoint

Hyperplanes cutting up space



Linear Classifier?

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?



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

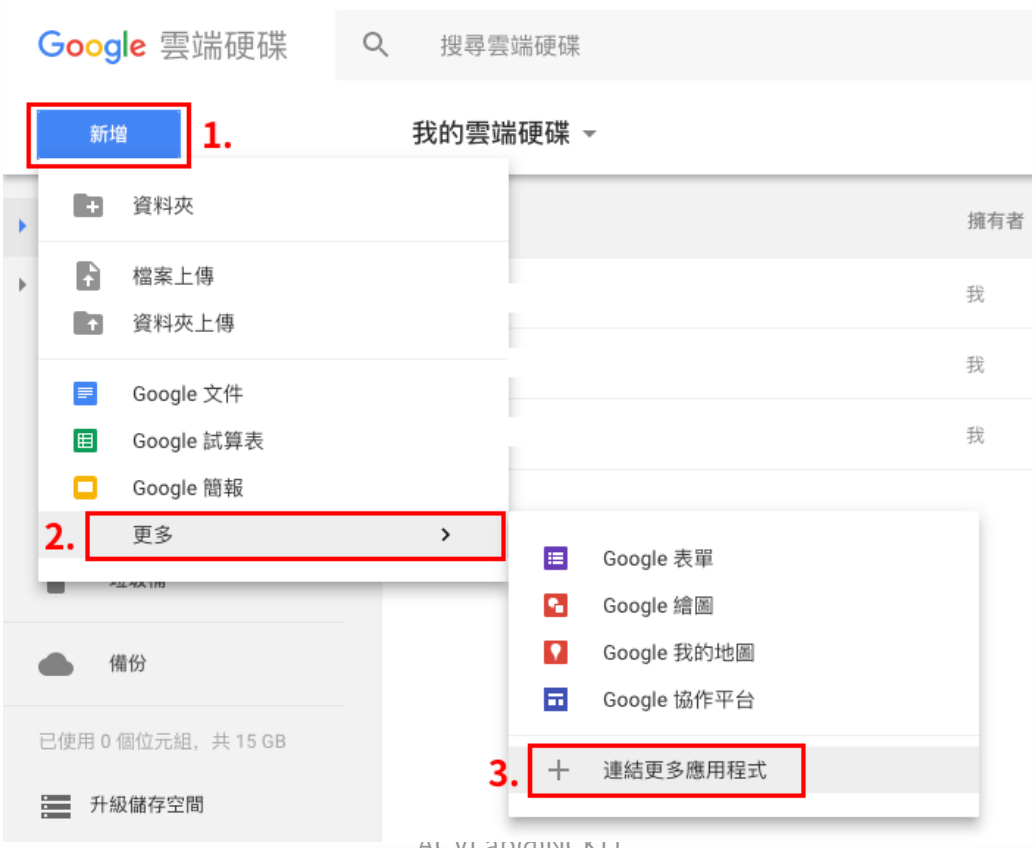
Linear Classifier: f(x,W) = Wx + b

- Next:
  - Loss function
    - quantifying what it means to have a good **W**
  - Optimization
    - start with random W and find a W that minimizes the loss
  - CNNs!
    - tweak the functional form off

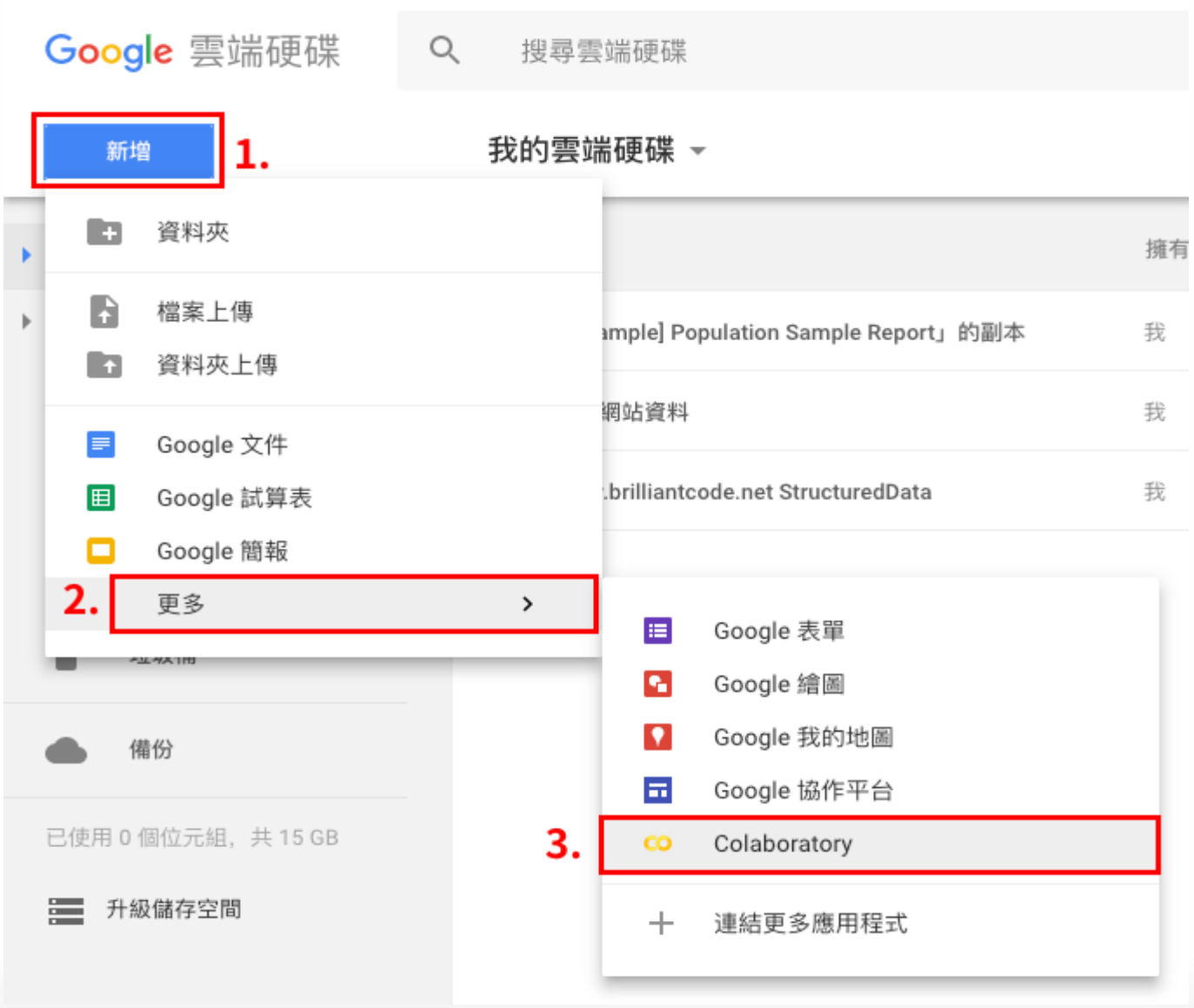
GOOGLE CLOUD RESOURCE: COLAB

Run TF without GPU?

- Google COLAB (Colaboratory)
  - Only three steps
    - Having free GPU/TPU resource if you have a google account



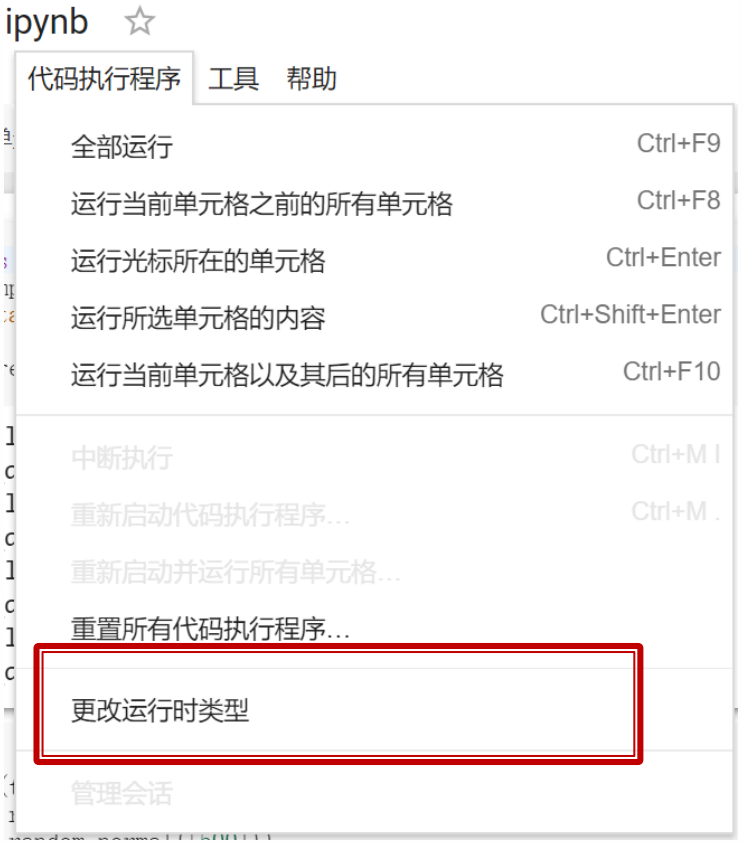
Click right key on ipynb and find colab

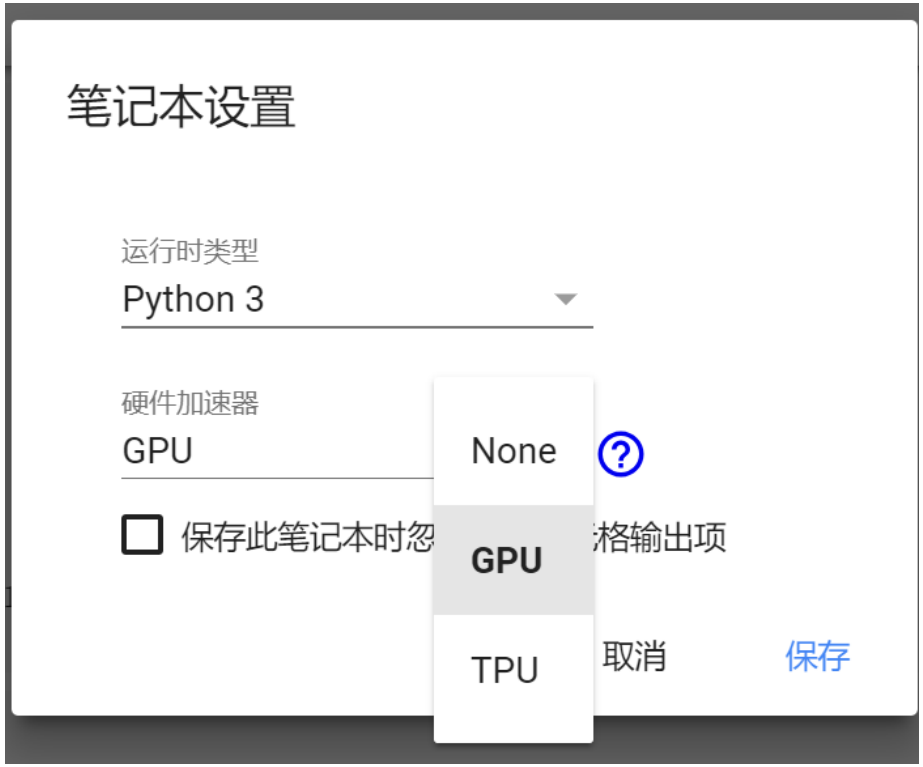


COLAB

- Free GPU (K80/V100) and TPU resource
  - 24G RAM (very large, compared to standard GPU)
    - RTX 2080 Ti
      - NTD 35,000
    - K80
      - NTD 180, 000
- You only can run a program on the free GPU 12 hours per day
  - Wait another 12 hours to access the free resource
- Runtime environment
  - Can be CPU/GPU/TPU

GPU Resource Usage





GPU vs TPU? TPU 為 Google 自家提出的加速硬體，照道理應該會比較快! 大家亦可試試看。

Check Whether The GPU Resource is Enabled

```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

- 若有使用到GPU，則會顯示

COLAB with Your Own Files

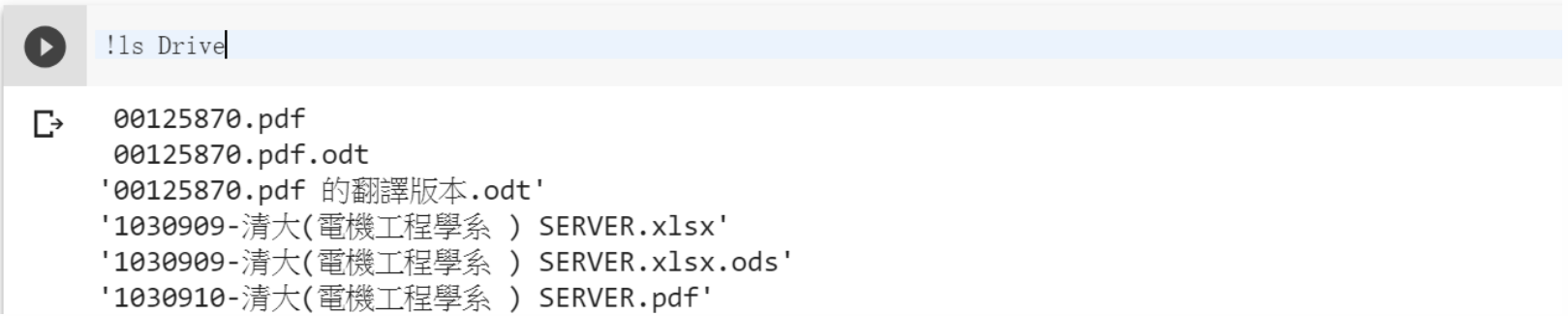
```
from google.colab import drive
import os

drive.mount('/content/gdrive’ )
os.chdir("/content/gdrive/My Drive") #更改路徑
os.getcwd() #查看當前路徑

Use ! To run bash shell in colab
!ls
## show your files
```

- 將上述程式碼貼到 COLAB 中，並根據【畫面指示】操作，就可以有與 GDRIVE連線的權限

Connect Google Drive with COLAB

```
!mkdir -p Drive
!google-drive-ocamlfuse Drive
```

- 上述指令在於
  - 在 COLAB 中建立一個資料夾 Drive (遠端中)
  - 利用指令!google-drive-ocamlfuse，把你連結到的雲端硬碟，連結到 Drive 資料夾中
  - 換句話說，你存取 Drive 資料夾，等於存取你的雲端硬碟

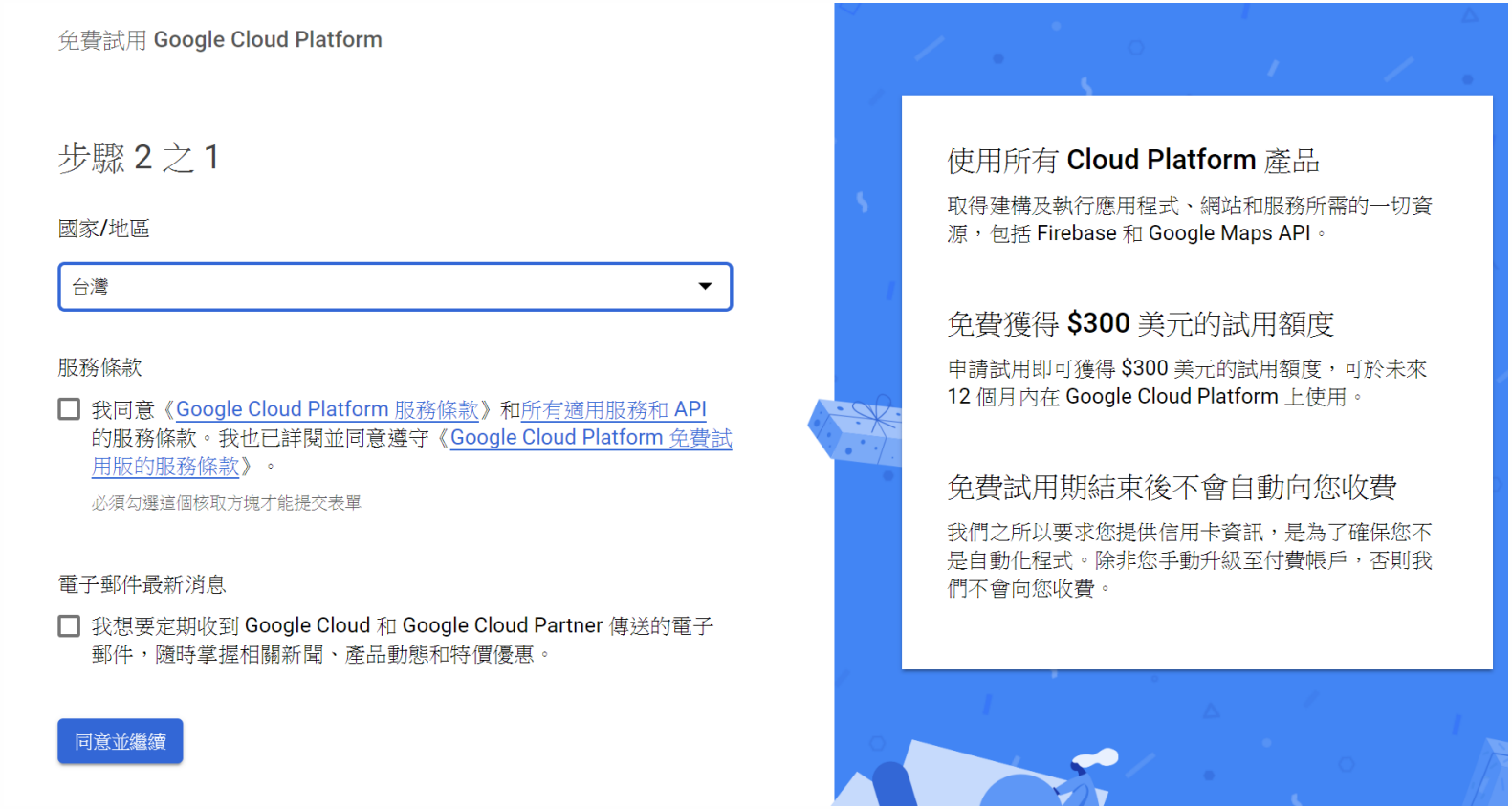


GOOGLE CLOUD SERVICE: VISION (AUTOML)

線上免費模型 (12個月免費)

- https://cloud.google.com/vision

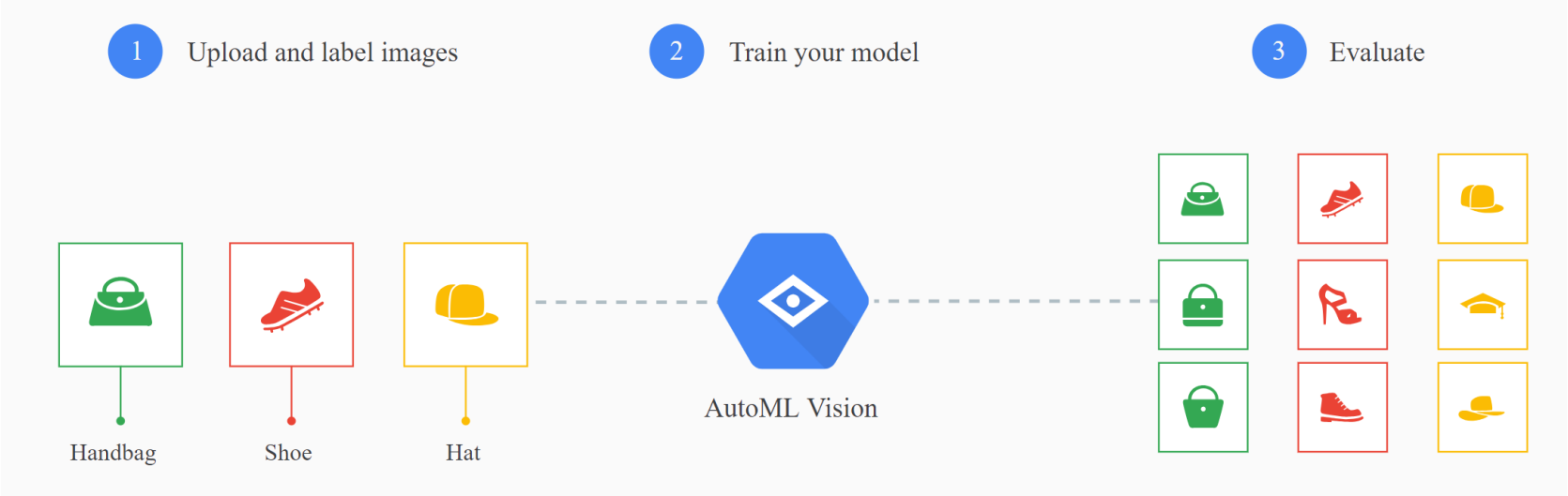
AutoML 授權申請 (尚未有付費帳號)



AutoML 授權申請 (尚未有付費帳號)

- 必須有信用卡
  - 不會偷偷被扣款
- 一般銀行的卡?
  - 台灣尚不支援
- 其他方法?
  - 請自行Google

Cloud Vision (AutoML) 使用方法



- 首先建立資料庫 (資料夾: 類別名稱)

名稱	修改日期	類型
5498592	2019/3/15 上午 0...	檔案資料夾
5513628	2019/3/15 上午 0...	檔案資料夾
5540978	2019/3/15 上午 0...	檔案資料夾

AutoML訓練自己資料庫方法

- 建立 Cloud Storage Bucket (CSB)
- 建立 CSB 的權限
- 上傳資料到自己的CSB
  - Including image files and their labels (.csv)
- 建立自己 CSB 的 Training dataset
- 訓練模型 & Testing

AutoML: 建立 CSB

- 首先開啟 Cloud AutoML 以及 Storage 的 API

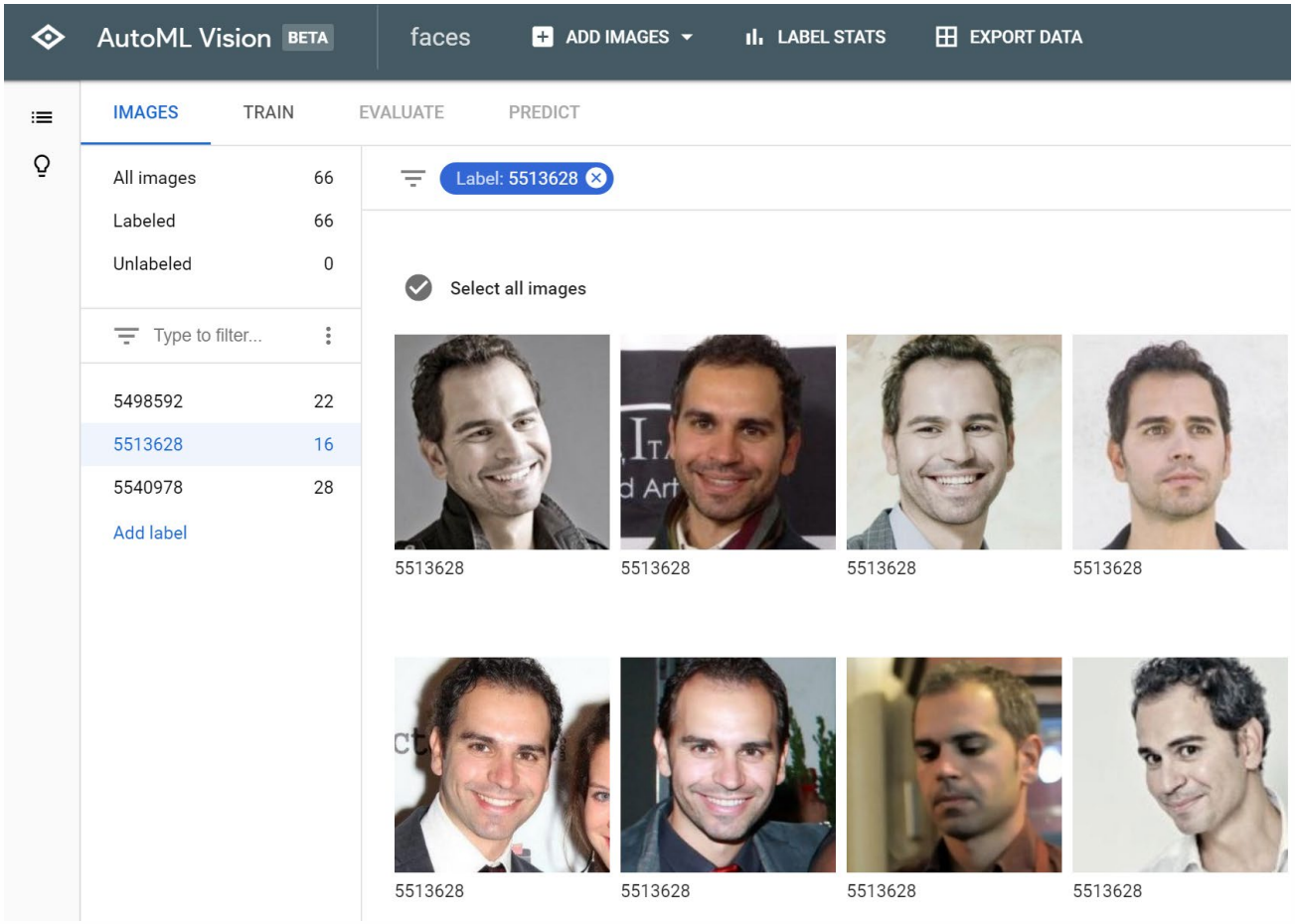
  http://console.cloud.google.com/?cloudshell=true

- 首先連結到 Google cloud 主控台 Console
  - PROJECT=$(gcloud config get-value project) && BUCKET="${PROJECT}-vcm"
  - gsutil mb -p ${PROJECT} -c regional -l us-central1 gs://${BUCKET}
- 建立 Service

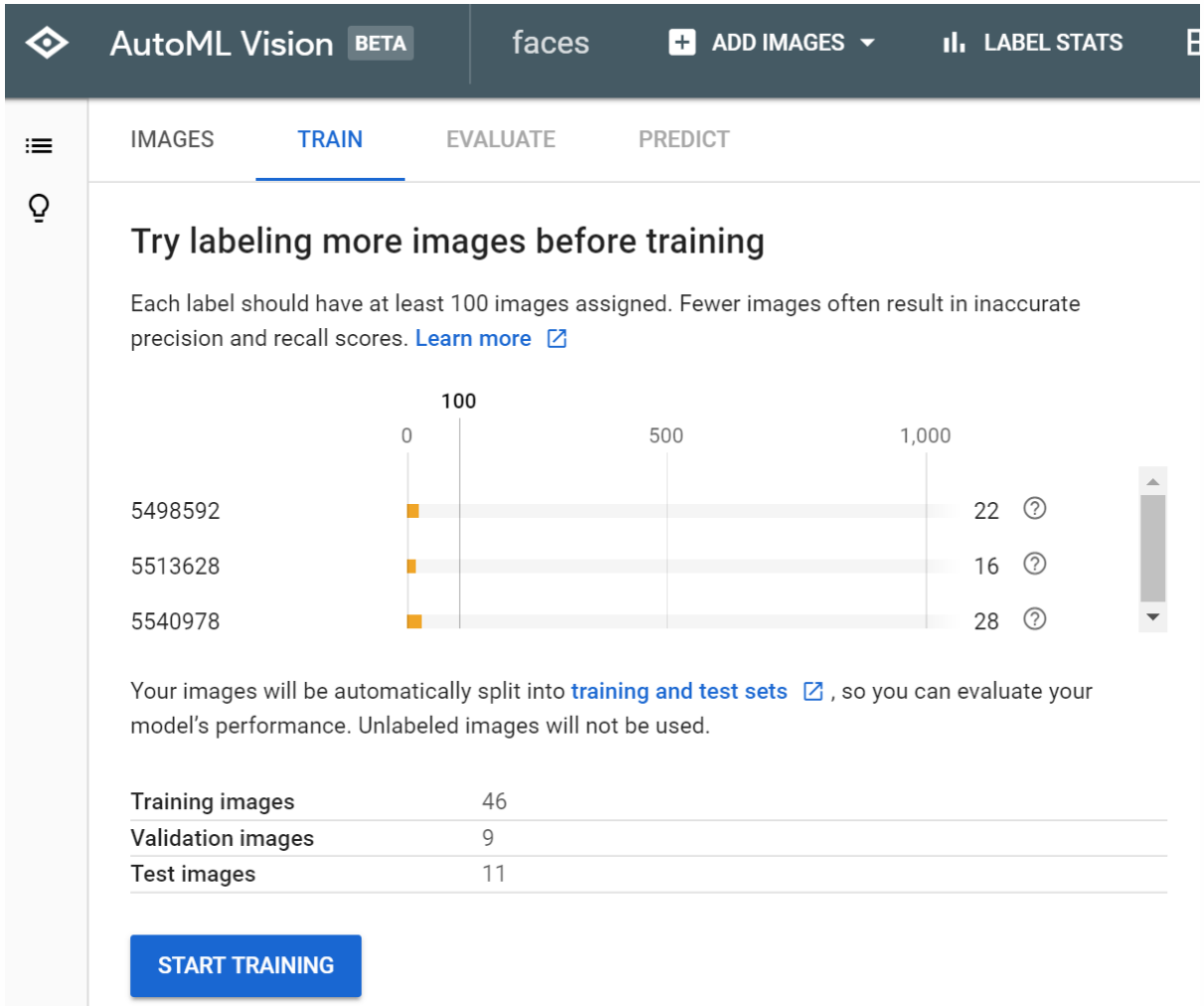
```
PROJECT=$(gcloud config get-value project)
gcloud projects add-iam-policy-binding $PROJECT \ --member="serviceAccount:custom-vision@appspot.gserviceaccount.com" \ --role="roles/ml.admin"
gcloud projects add-iam-policy-binding $PROJECT \ --member="serviceAccount:custom-vision@appspot.gserviceaccount.com" \ --role="roles/storage.admin"
```

建立自己的資料庫

- 進入到 AutoML Vision 的頁面

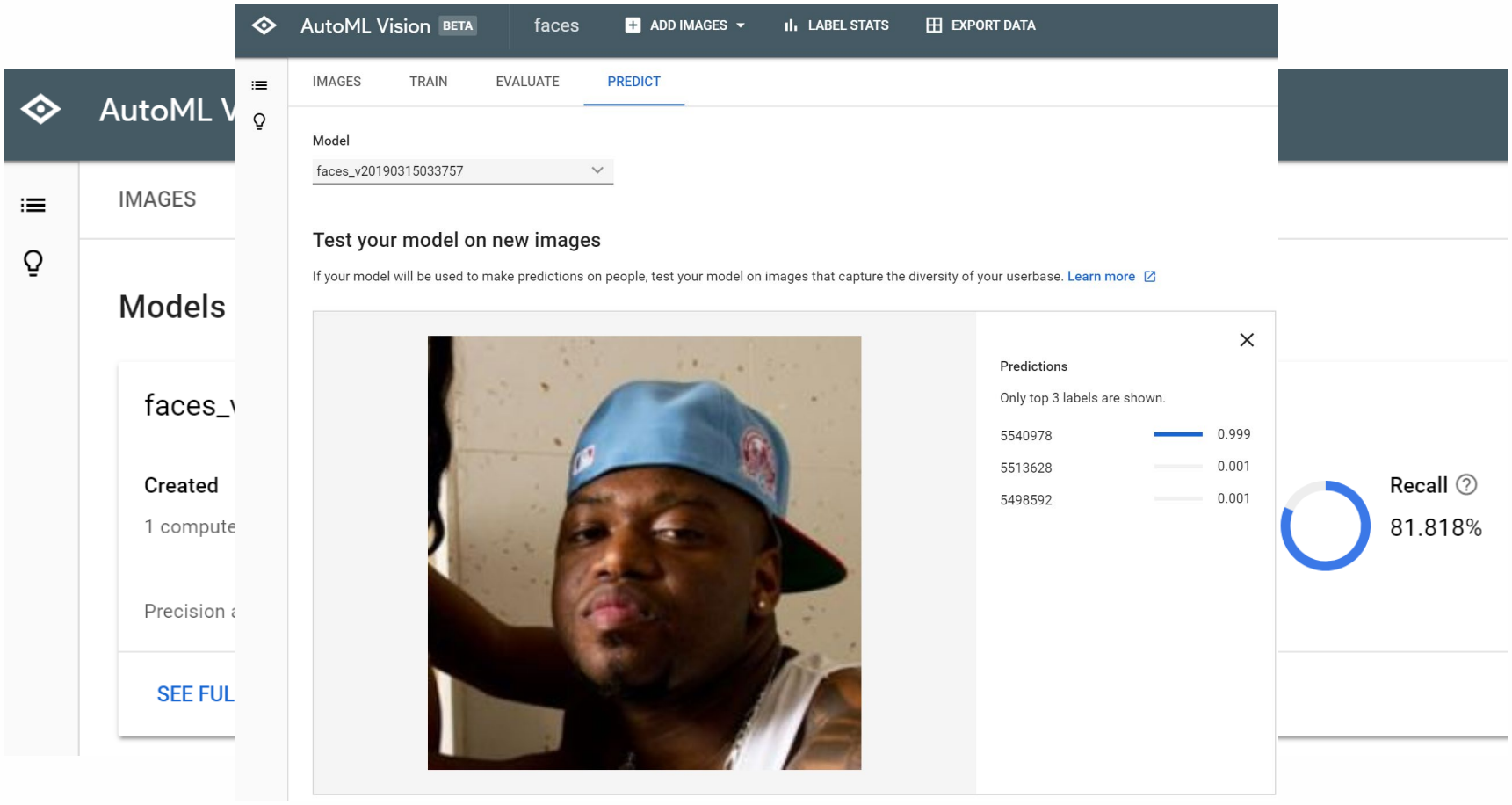


訓練MODEL


AutoML Vision BETA
faces
ADD IMAGES
LABEL STATS
IMAGES
TRAIN
EVALUATE
PREDICT
Try labeling more images before training
Each label should have at least 100 images assigned. Fewer images often result in inaccurate precision and recall scores. Learn more
100
0
500
1,000
5498592
22
5513628
16
5540978
28
Your images will be automatically split into training and test sets , so you can evaluate your model's performance. Unlabeled images will not be used.
Training images 46
Validation images 9
Test images 11
START TRAINING

Learned Model

- 可以馬上看Training 結果，Predict 可以上傳照片並辨識資料



Comparison

	Tensorflow (Own PC)	COLAB (Tensorflow)	AutoML
執行速度	快	中	中（要網路）
訓練速度	快	中	快（已上傳的話）
建立方法速度	慢	慢	超快
方便性	一般（要寫程式）	優（有網路就可）	優（要註冊，未來要錢）
客製化彈性	優（自己建立模型）	優	無
辨識效能	高	高	高一點
任務置換	容易	容易	麻煩（要換資料）
I/O效能	高（SSD版）	低（有限制）	很高
成本	一般（GPU的錢）	低（只要網路）	中（算訓練時間收費）