Lecture 2: Image Classification

COMP 4970 / 7970

Deep Learning

Register for Free cloud compute credits

1. Amazon AWS Educate : $100 credit / student
   - Refs:
     - https://aws.amazon.com/education/awseducate/
2. Google Cloud Platform: $350 credit / student
   - Free $300 credit upon registration
   - Additional $50 from this class (thanks to Google!)
   - Refs:
     - http://cs231n.github.io/gce-tutorial/
     - https://cloud.google.com/getting-started/

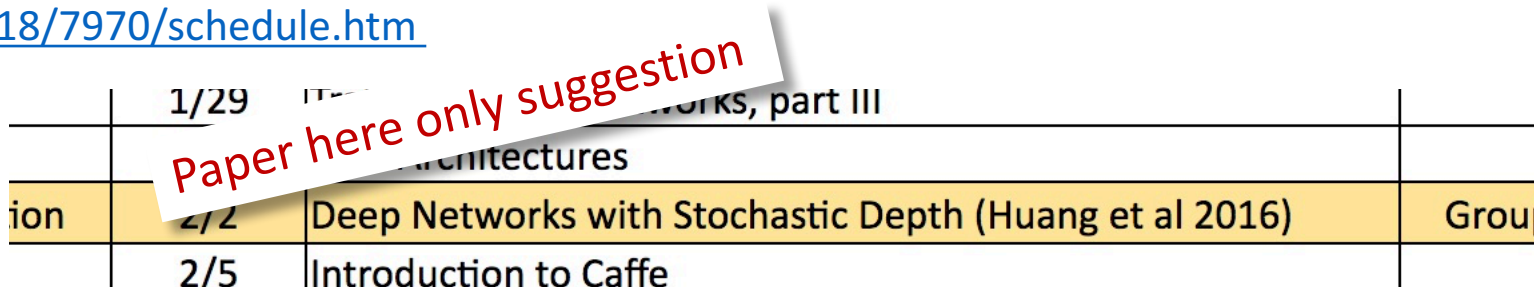
- Necessary for Projects (if you use GPUs)
- Optional for Assignments

Administrative

- Use Piazza for questions/discussions on
  - Sign-up: http://piazza.com/auburn/spring2018/comp49707970
  - Assignments
  - Presentation topic and papers
  - Proposal / Projects
  - Anything regarding Deep Learning
- Presentation Topic + Paper
  - Schedule: http://s.anhnguyen.me/2018/7970/schedule.htm

1. Choose a topic
2. Choose a paper

- Tips:
  - **You are excited about!**
  - Has talk/presentation videos online
  - Has demo
  - Has open-source code



A1 out Deadline: Jan 29

- http://cs231n.github.io/assignments2017/assignment1/
- Deliverable: Zip up your code as A1_<id>_<firstname>_<lastname>.zip and upload to Canvas.

In this assignment you will practice putting together a simple image classification pipeline, based on the k-Nearest Neighbor or the SVM/Softmax classifier. The goals of this assignment are as follows:

- understand the basic **Image Classification pipeline** and the data-driven approach (train/predict stages)
- understand the train/val/test **splits** and the use of validation data for **hyperparameter tuning**.
- develop proficiency in writing efficient **vectorized** code with numpy
- implement and apply a k-Nearest Neighbor (**kNN**) classifier
- implement and apply a Multiclass Support Vector Machine (**SVM**) classifier
- implement and apply a **Softmax** classifier
- implement and apply a **Two layer neural network** classifier
- understand the differences and tradeoffs between these classifiers
- get a basic understanding of performance improvements from using **higher-level representations** than raw pixels (e.g. color histograms, Histogram of Gradient (HOG) features)

Project ideas

>350 Datasets Available at:
http://archive.ics.uci.edu/ml/

Newest Data Sets:

2012-07-17:	Skin Segmentation
2012-07-17:	Planning Relax
2012-07-04:	Nomao
2012-06-22:	SMS Spam Collection
2012-06-09:	OPPORTUNITY Activity Recognition
2012-05-21:	ILPD (Indian Liver Patient Dataset)
2012-04-25:	Gas Sensor Array Drift Dataset

Most Popular Data Sets (hits since 2007):

352287:	Iris
249987:	Adult
219203:	Wine
179585:	Breast Cancer Wisconsin (Diagnostic)
165076:	Car Evaluation
139614:	Abalone
125301:	Poker Hand

Project ideas

>600 Datasets Available at: http://homepages.inf.ed.ac.uk/rbf/CVonline/Imagedbase.htm

Image/video database categories:

Action Databases
Attribute recognition
Autonomous Driving
Biological/Medical
Camera calibration
Face and Eye/Iris Databases
Fingerprints
General Images
General RGBD and depth datasets
General Videos
Hand, Hand Grasp, Hand Action and Gesture Databases
Image, Video and Shape Database Retrieval
Object Databases
People (static), human body pose
People Detection and Tracking Databases (See also Surveillance)
Remote Sensing
Scene or Place Segmentation or Classification
Segmentation
Simultaneous Localization and Mapping
Surveillance (See also People)
Textures
Urban Datasets
Other Collection Pages
Miscellaneous Topics

FYI: CVonline (http://homepages.inf.ed.ac.uk/rbf/CVonline)

has a lot of other resources:

- indexed pointers to about 2000 computer vision / image analysis topics

- list of image processing software

- list of image analysis applications

- list of about 950 computer vision / image processing / optics books

with links to 43 free online and 353 online subscription books

Project ideas http://cs231n.stanford.edu/project.html

- ICLR: International Conference on Learning Representations
- Past CS229 Projects: Example projects from Stanford machine learning class
- Kaggle challenges: An online machine learning competition website. For example, a Yelp classification challenge.

For applications, this type of projects would involve careful data preparation, an appropriate loss function, details of training and cross-validation and good test se evaluations and model comparisons. Don't be afraid to think outside of the box. Some successful examples can be found below:

- Teaching Deep Convolutional Neural Networks to Play Go
- Playing Atari with Deep Reinforcement Learning
- Winning the Galaxy Challenge with convnets

ConvNets also run in real time on mobile phones and Raspberry Pi's - feel free to go the embedded way. You may find DeepBeliefSDK helpful. This particular proje might be slightly out of date, but it may help you find more like it.

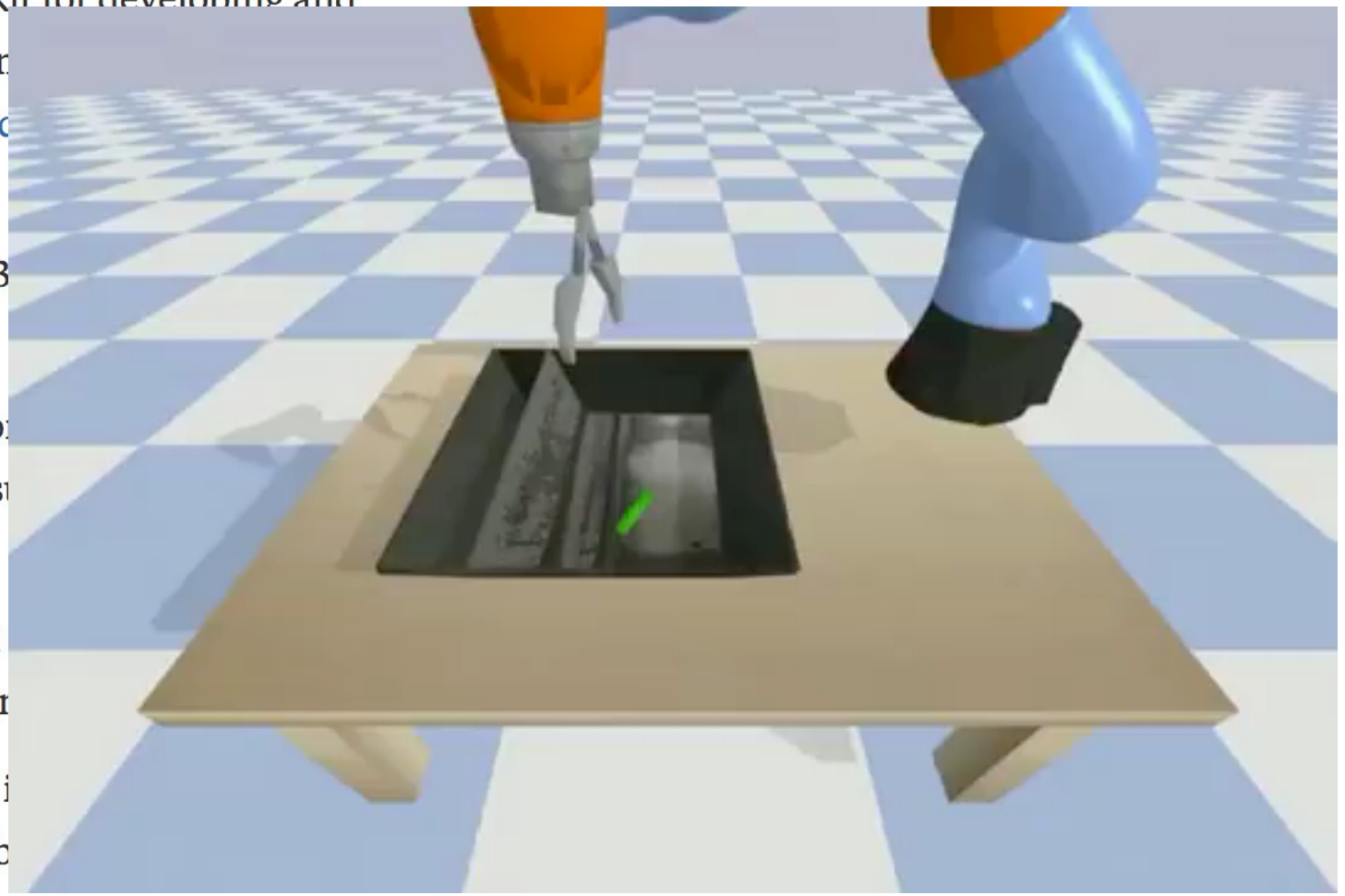
For models, ConvNets have been successfully used in a variety of computer vision tasks. This type of projects would involve understanding the state-of-the-art vision models, and building new models or improving existing models for a vision task. The list below presents some papers on recent advances of ConvNets in tl computer vision community.

- **Object recognition**: [Krizhevsky et al.], [Russakovsky et al.], [Szegedy et al.], [Simonyan et al.], [He et al.]
- **Object detection**: [Girshick et al.], [Sermanet et al.], [Erhan et al.]
- **Image segmentation**: [Long et al.]
- **Video classification**: [Karpathy et al.], [Simonyan and Zisserman]
- **Scene classification**: [Zhou et al.]
- **Face recognition**: [Taigman et al.]
- **Depth estimation**: [Eigen et al.]
- **Image-to-sentence generation**: [Karpathy and Fei-Fei], [Donahue et al.], [Vinyals et al.]
- **Visualization and optimization**: [Szegedy et al.], [Nguyen et al.], [Zeiler and Fergus], [Goodfellow et al.], [Schaul et al.]

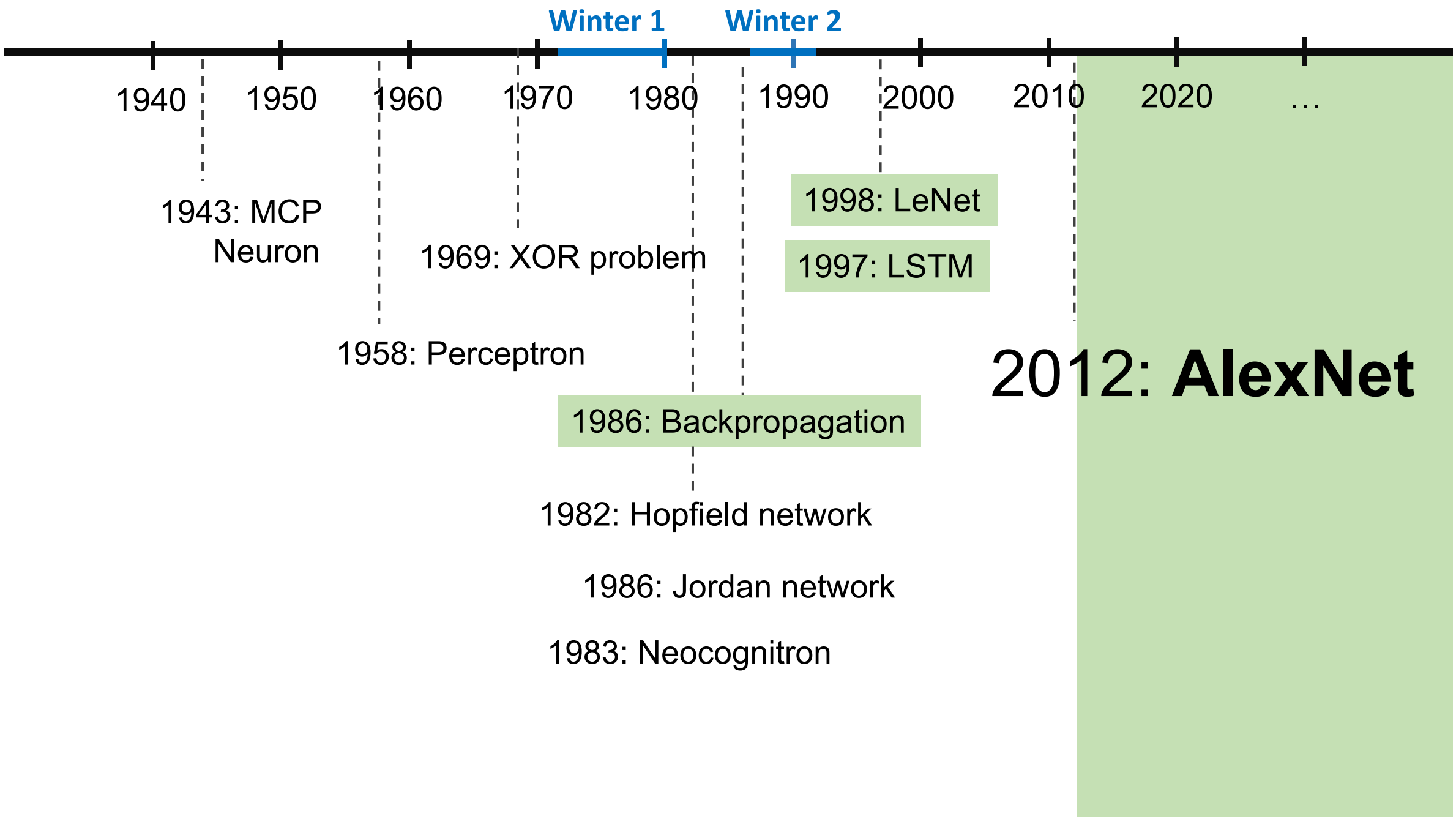
Project ideas

https://www.quora.com/What-is-the-best-platform-to-simulate-Reinforcement-Learning-agent

- OpenAI Gym has become a standard toolkit for developing and comparing reinforcement learning algorithm environments. Now, it also integrates robosc simulation.
- DeepMind Lab has a suite of challenging 3 solving tasks for learning agents.
- OpenAI Universe is a software platform fo AI's general intelligence across the world's s and other applications.
- DeepMind and Blizzard's Starcraft, which minigames for developing AI agents that car
- The Arcade Learning Environment (ALE) framework that allows researchers and hobb for Atari 2600 games.
- Microsoft's project Malmö is a platform for Artificial Intelligence experimentation and research built on top of Minecraft. We aim to

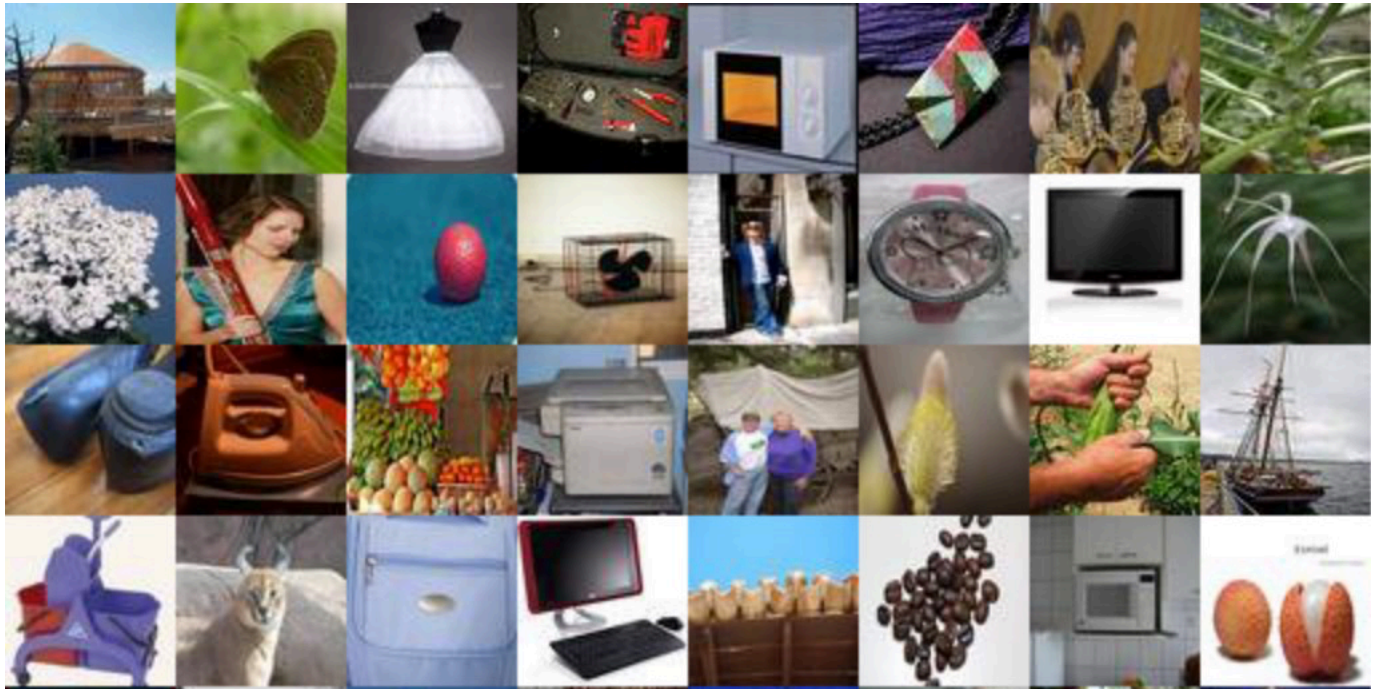


credit: David Ha


Winter 1
Winter 2
1940
1950
1960
1970
1980
1990
2000
2010
2020
…
1943: MCP Neuron
1958: Perceptron
1969: XOR problem
1998: LeNet
1997: LSTM
2012: AlexNet
1986: Backpropagation
1982: Hopfield network
1986: Jordan network
1983: Neocognitron

IMAGENET **Challenge**

- 1.3M images
- 1000 categories



2

0 2000 2010 2020 …

998: LeNet

997: LSTM

gation

ork

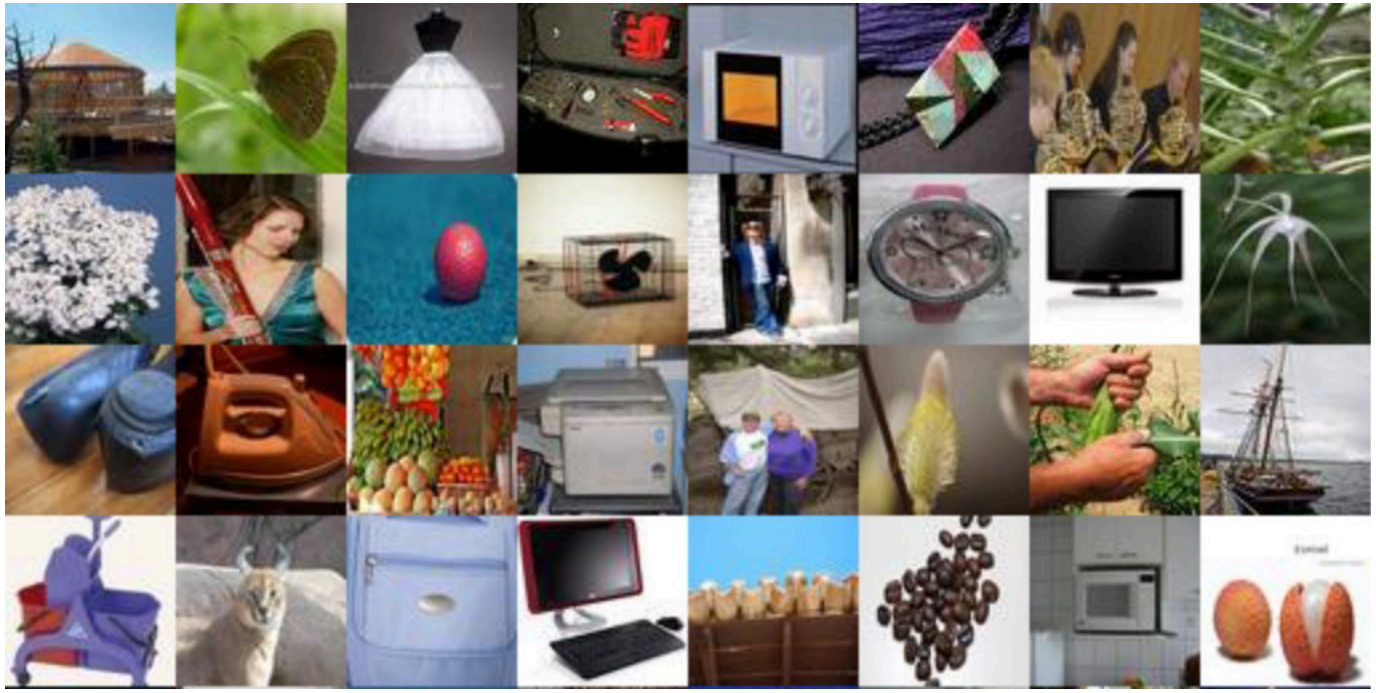
twork

n

2012: **AlexNet**

IMAGENET **Challenge**

- 1.3M images
- 1000 categories



2

2000 2010 2020 …

998: LeNet

997: LSTM

gation

ork

tw

n

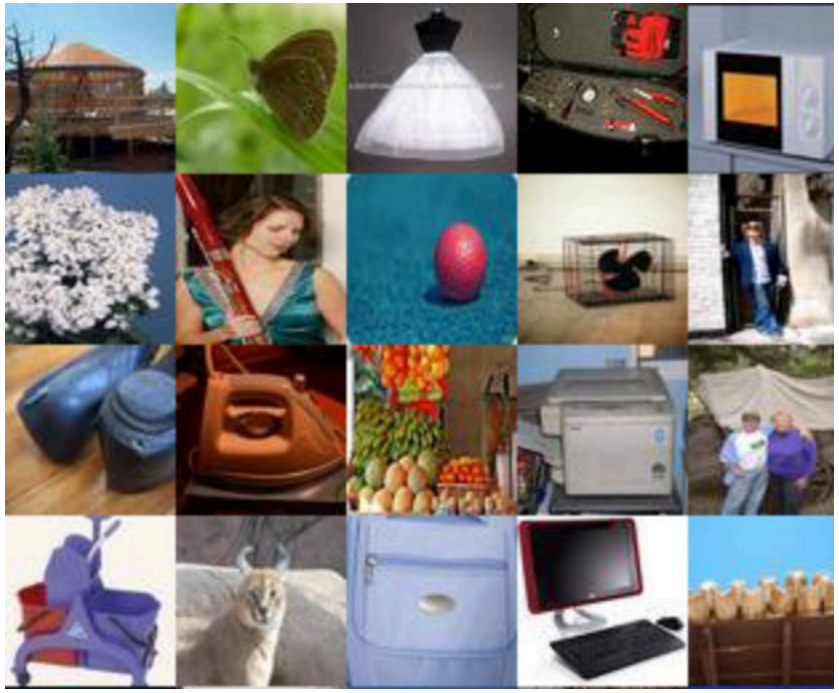
2012: **AlexNet**

- Convolutional Neural Networks
- Image Classification

IMAGENET **Challenge**

Image Classification

- 1.3M images
- 1000 categories



given set of discrete labels

→

dog

cat

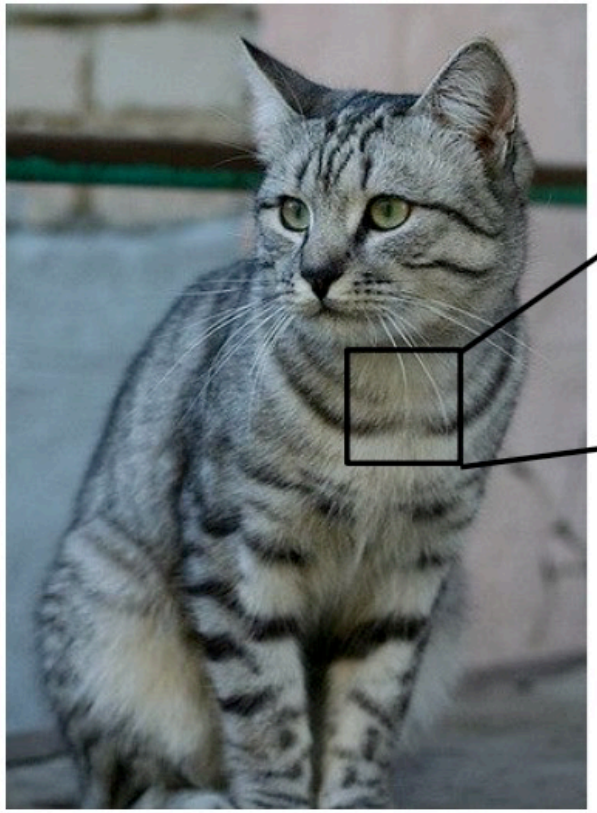
truck

plane

...

The Problem: Semantic gap

credit: cs231n



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]:

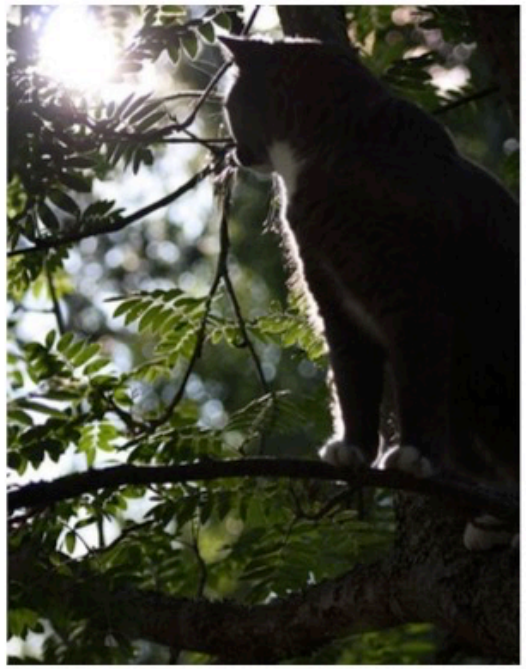
e.g. 800 x 600 x 3
(3 channels RGB)

Challenges: Illumination

credit: cs231n

This image is CC0 1.0 public domain

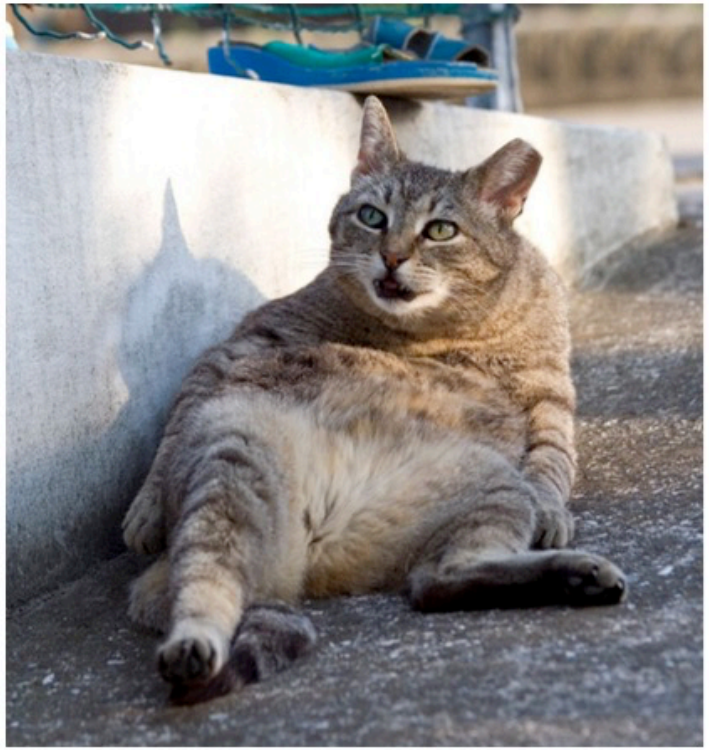
This image is CC0 1.0 public domain



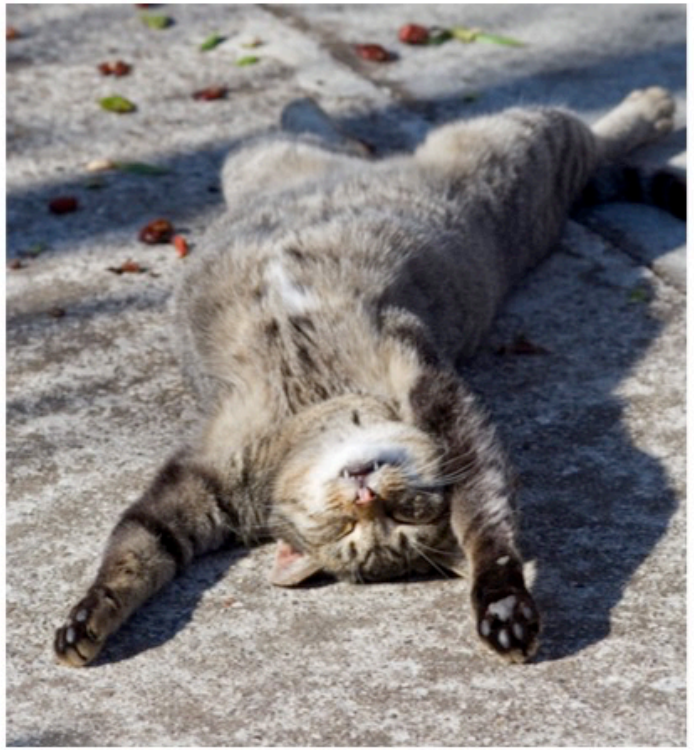
This image is CC0 1.0 public domain

Challenges: Deformation

credit: cs231n



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

Challenges: Occlusion

credit: cs231n

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: Intra-class variation

credit: reddit

Challenges: Inter-class similarity

credit: ImageNet

Tabby cat

Tiger cat

An image classifier

credit: cs231n

```
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.

Data-driven

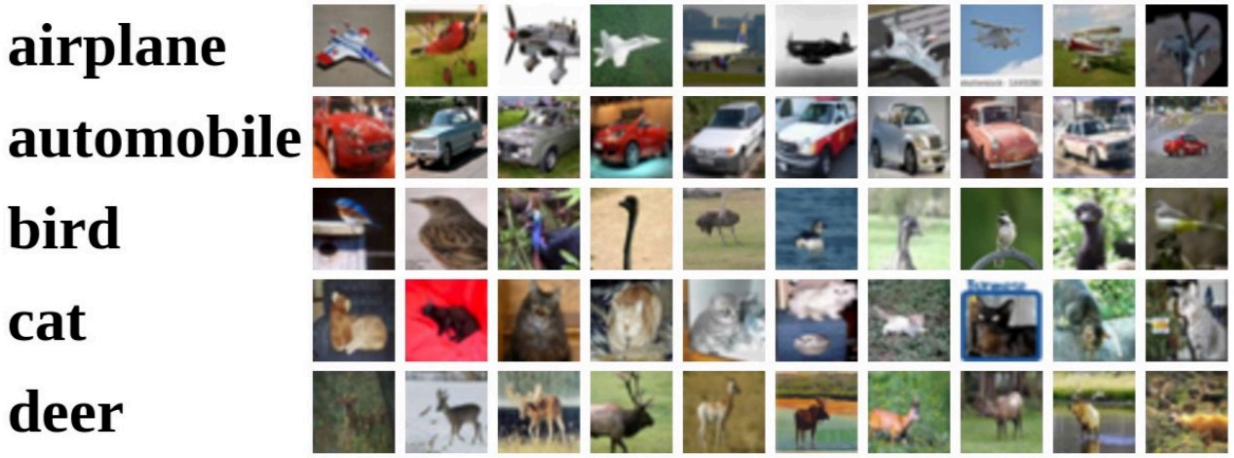
credit: cs231n

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

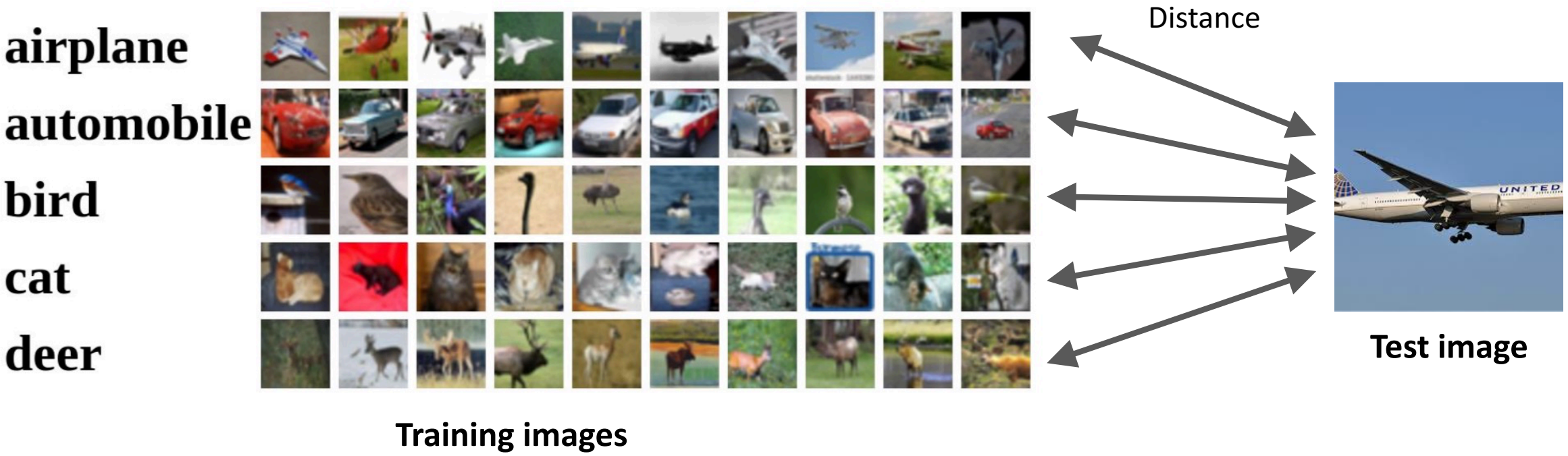
```
def train(images, labels):
  # Machine learning!
  return model
```

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

Example training set

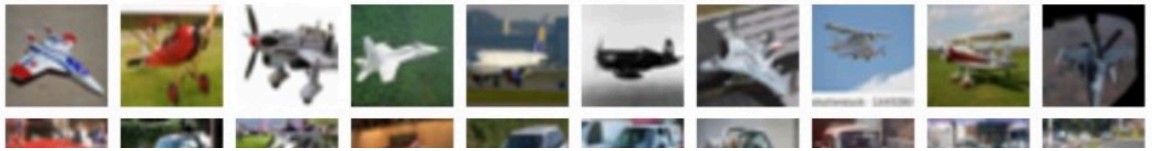


Nonparametric approach: Nearest Neighbor



Nonparametric approach: Nearest Neighbor

airplane



Distance

Test image

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

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

add → 456

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

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
```

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

- Model size = Training set size
  - (*nonparametric*)
- Time complexity
  - Train: O(1)
  - Test: 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

- Model size = Training set size
  - (*nonparametric*)
- Time complexity
  - Train: O(1)
  - Test: O(n)
- Q: Is this a good classifier?

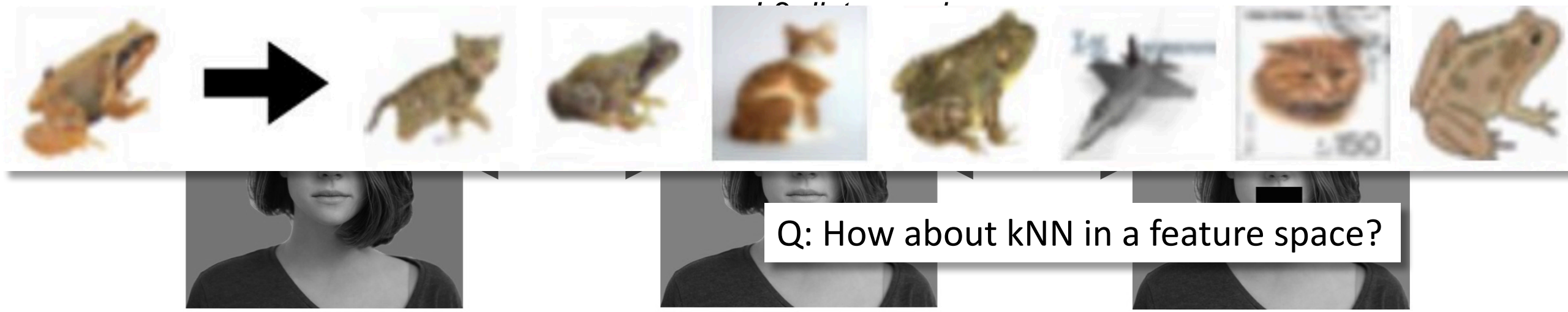
Demo:
http://vision.stanford.edu/teaching/cs231n-demos/knn/

Problems with Nearest Neighbor

1. Model size (= training set size)
2. Slow test time
3. Curse of dimensionality
   - # training points needed is exponential wrt dimension
4. Distance metrics (e.g. L1, L2) in pixel space

Problems with Nearest Neighbor

1. Model size (= training set size)
2. Slow test time
3. Curse of dimensionality
   - # training points needed is exponential wrt dimension
4. Distance metrics (e.g. L1, L2) in pixel space

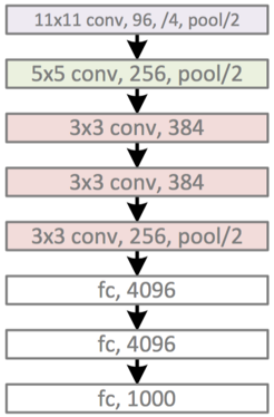


Q: How about kNN in a feature space?

Linear Classifiers

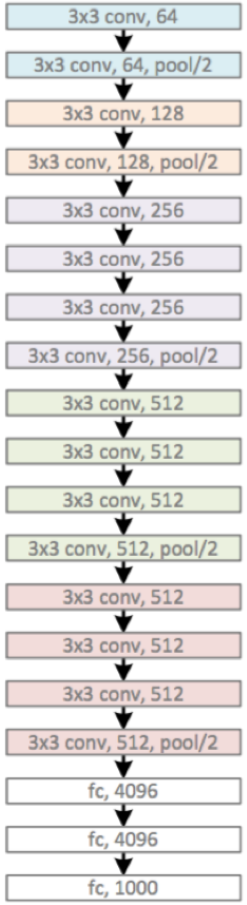
IMAGENET classification

2012: AlexNet
8 layers



Error: 15.3%

2014: VGG
19 layers



8.5%

2015: GoogLeNet
22 layers

7.8%

2016: ResNet
>100 layers

Human: **5.1%**

4.4%

2017

...

3%

Neural Network

Linear classifiers

Recall CIFAR10

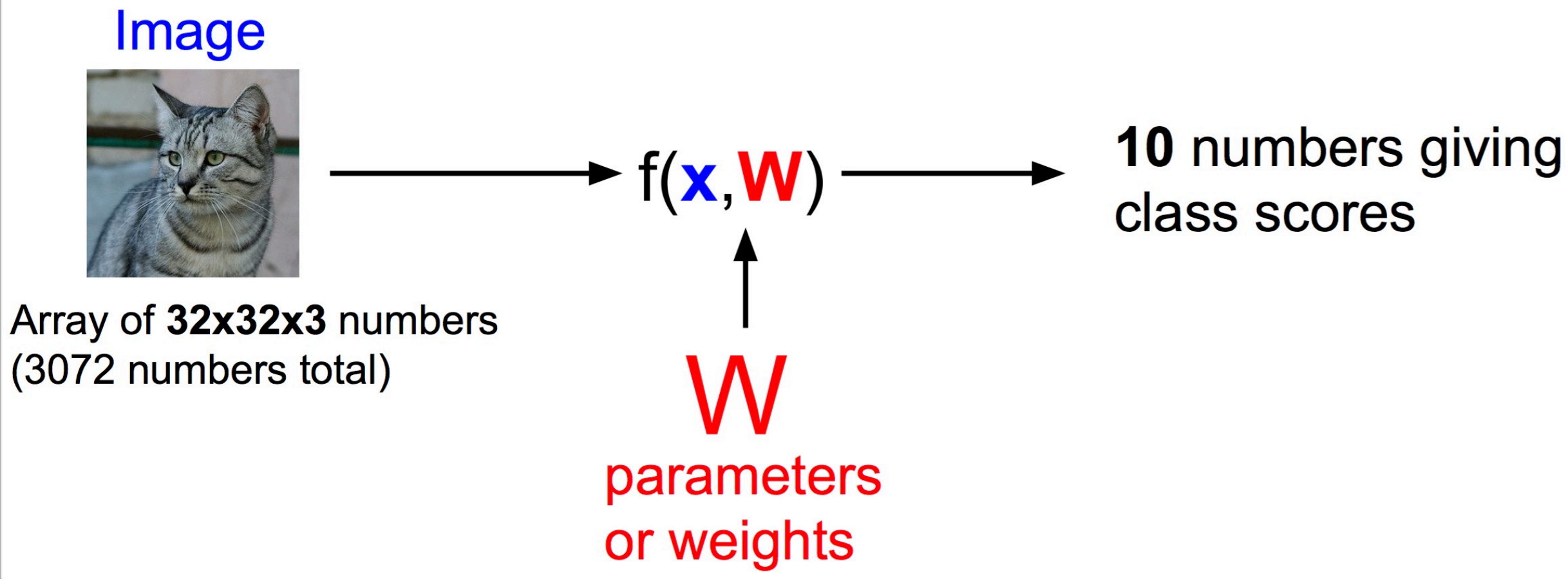
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

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

10,000 test images.

Parametric Approach

credit: cs231n



Parametric Approach: Linear Classifier

credit: cs231n

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

Image

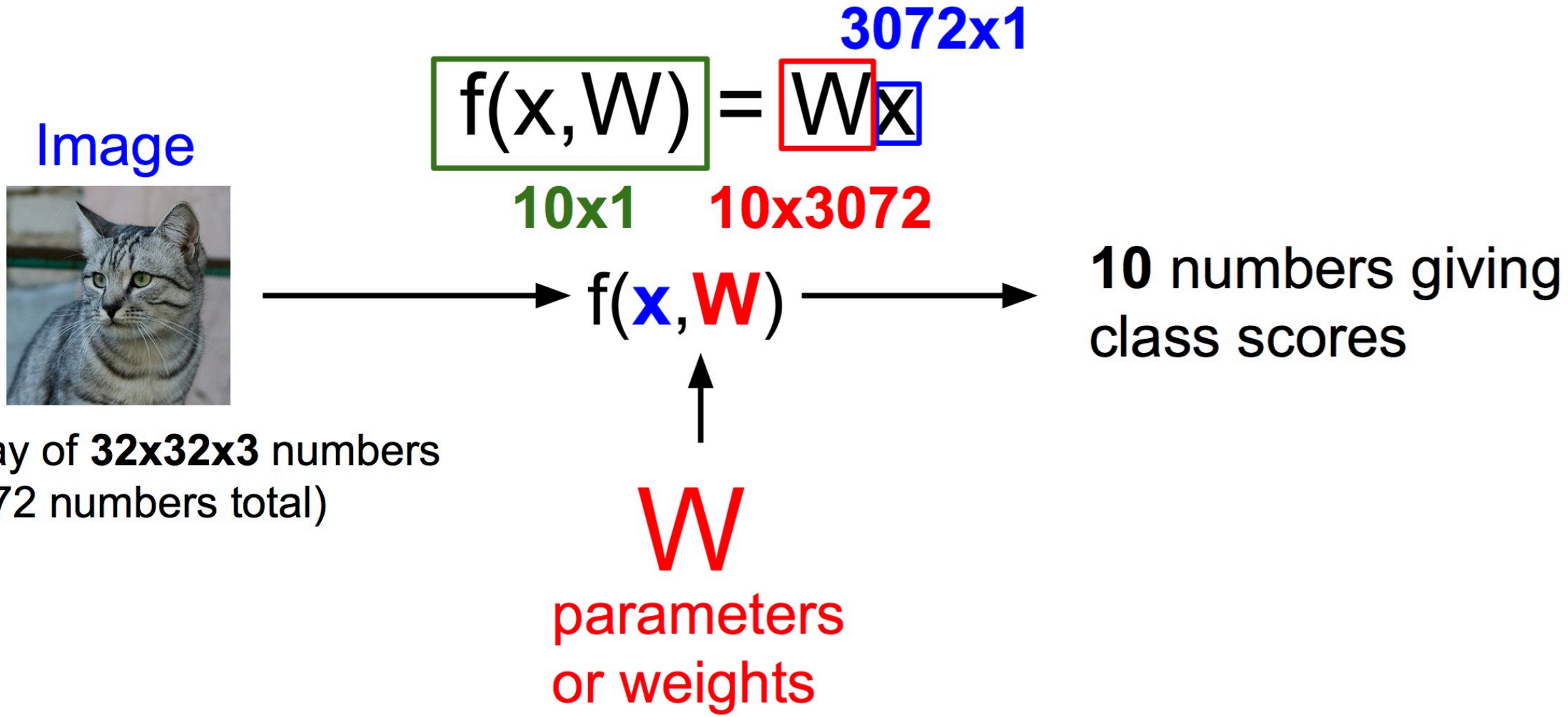
Array of **32x32x3** numbers
(3072 numbers total)

f(**x**,**W**) → **10** numbers giving class scores

W
parameters
or weights

Parametric Approach: Linear Classifier

credit: cs231n



Parametric Approach: Linear Classifier

credit: cs231n

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

3072x1 (x)

10x1 (b)

10x1 (f(x,W))

10x3072 (W)

Image

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

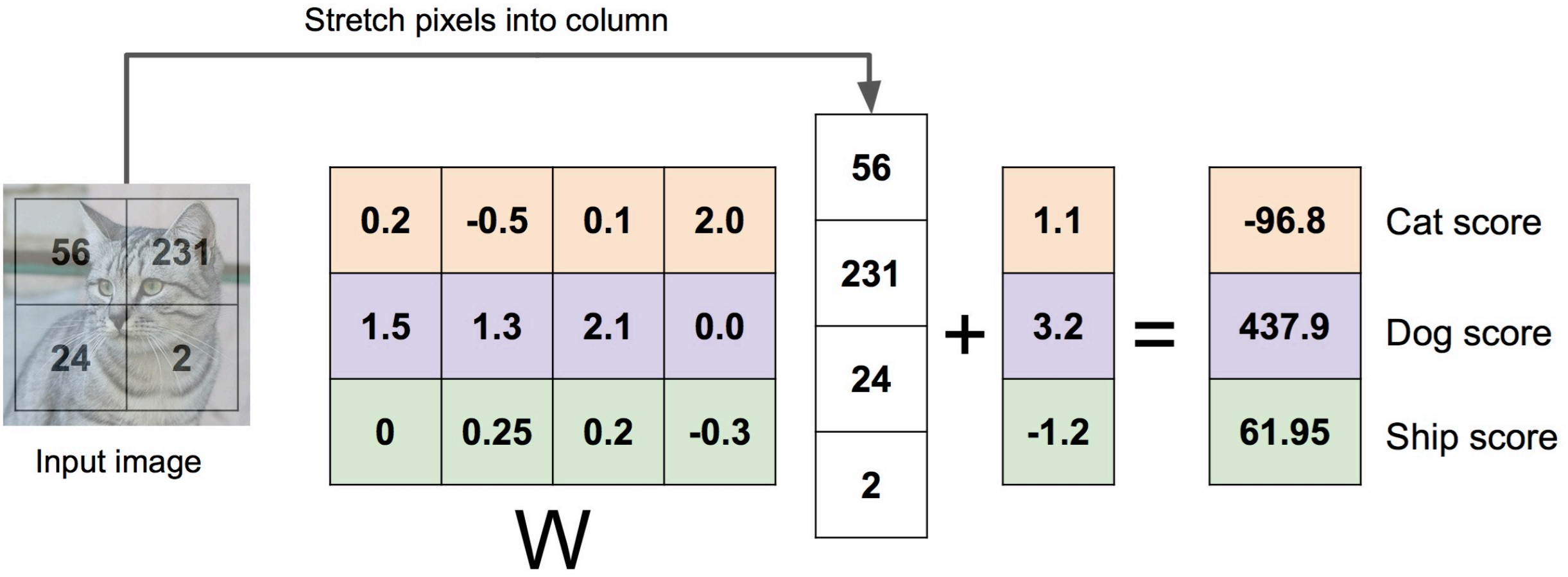
f(**x**,**W**)

10 numbers giving class scores

W

parameters or weights

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



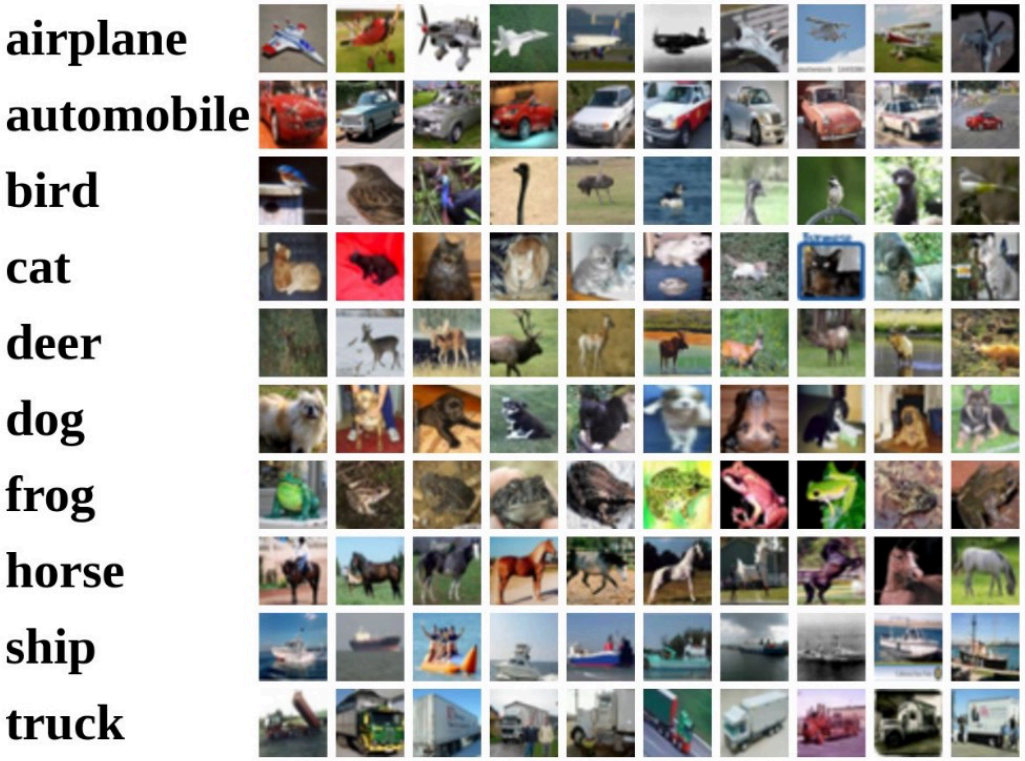
Interpreting a Linear Classifier

Q: How to learn W?

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

Example trained weights of a linear classifier trained on CIFAR-10:

Interpreting a Linear Classifier



Q: How to learn W?

$f(x,W)$

Demo: https://goo.gl/Qgr84h
OR http://playground.tensorflow.org/

Example trained weights of a linear classifier trained on CIFAR-10: