



Applied Artificial Intelligence with Python

SEOW KHEE WEI



Quick Poll

<http://bit.ly/2Y8MTi2>





Introduction of trainer



Name

Seow Khee Wei

Telegram

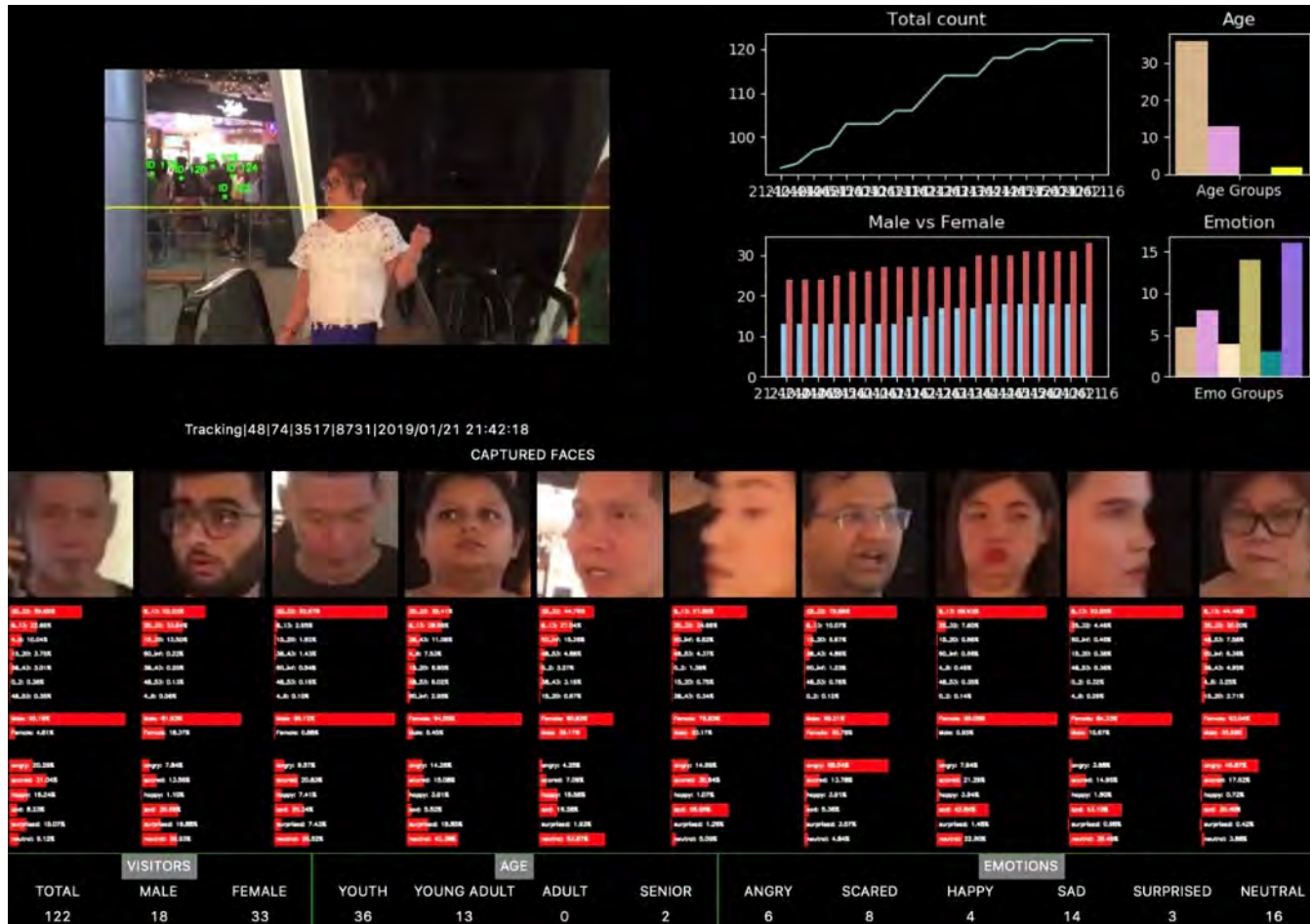
@kwseow

Email

seow_khee_wei@rp.edu.sg



Projects





Projects

Silver Cross Medical

SILVER CROSS CLINIC
COMMUNITY REGISTRATION NO: 1999000406
GST REGISTRATION NO: 1999000406
BLK 365 WOODLANDS ST 11, #01-75, SINGAPORE 730305
TEL: 63633646 / FAX: 63633349

MEDICAL CERTIFICATE

NAME: ONG DAO EN IDENTIFICATION: 10032106G
VISIT DATE: 02-08-2019 10-508061
190805-01
491187

This is to certify that the above mentioned has been given:
UNFIT FOR SCHOOL for 1 day(s) from 02-08-2019 to 02-08-2019

REMARKS:

Dr. Low Sau Wai
MD (Malaysia)
MCR No. M61492J

Silver Cross Clinic (Singapore)
Blk 365 Woodlands St 11,
#01-75 Singapore 730305
Tel: 6363 3646 Fax: 6363 3349

DR. LOW SAU WAI (M61492J)
DOCTOR

Not Valid for Absence from Court Attendance

Ref No: 20192141114256
Printed By: Clinic Assistant T SCWD (02-08-2019)





Programmes

Lecture 1.1 - What is machine learning

Activity 1.1 – Training a Regression

Model using Linear Regression

Lecture 1.2 – What is Neural Network

Activity 1.2 - Training a Regression Model
using Neural Network

Lecture 1.3 – What is deep learning

Lecture 1.4 – Speech Recognition

Activity 1.3 – Recognising Speech with
DeepSpeech

Lecture 1.5 – Natural Language Processing
with spaCy

Activity 1.4 - Extracting Facts

Activity 1.5 – Training and Using a Text
Classifier



Programmes

Lecture 2.1 – Computer Vision

Activity 2.1 – Building a Bird Classifier with CNN

Lecture 2.2 – Transfer Learning

Activity 2.2 – Building a Bird Classifier with transfer learning

Lecture 2.3 – Image Segmentation

Activity 2.3 – Using a Pre-trained Masked R-CNN Model

Activity 2.4 – Using a Pre-trained Mask R-CNN Model

Lecture 2.4 – Genetic Programming

Activity 2.5 – Knapsack problem



Prereqs and Preparations

Familiar with

- Python Language
- Ubuntu
- CLI
- Anaconda
- Any text editor

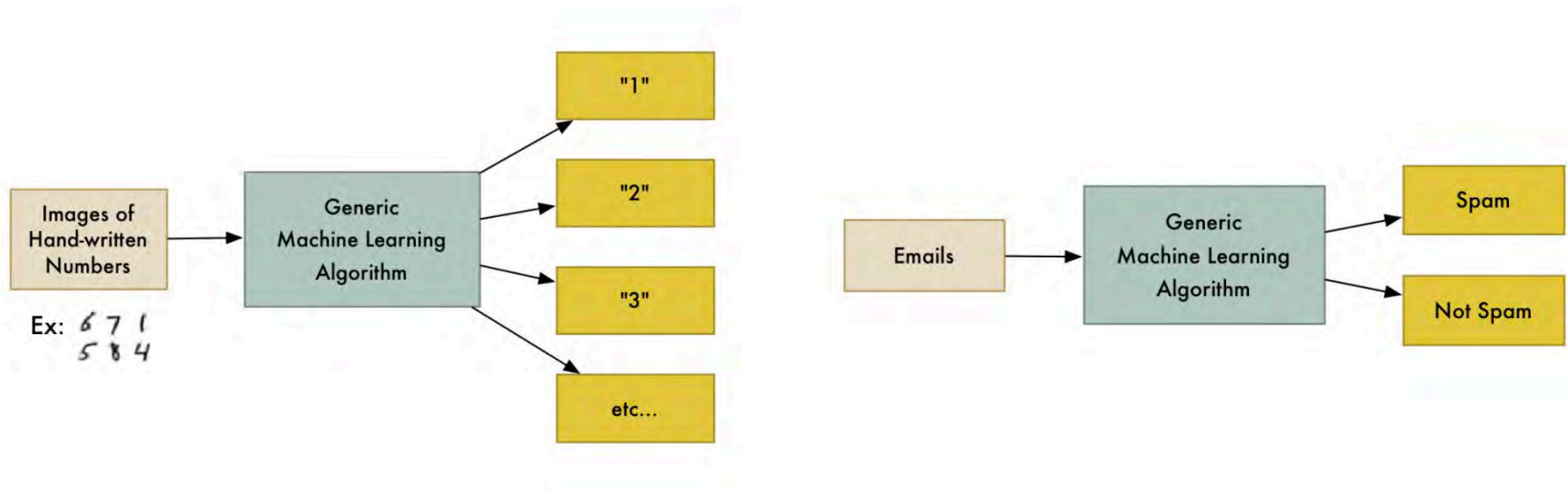


Workshop materials: <http://bit.ly/2VQacjh>



What is Machine Learning

Machine learning is the idea that there are generic algorithms that can tell you something interesting about a set of data without you having to write any custom code specific to the problem. Instead of writing code, you feed data to the generic algorithm and it creates its own logic based on the data.





Type of Machine Learning

- Supervised Learning

Bedrooms	Sq. Feet	Neighborhood	Sale Price
3	2000	Normaltown	\$250,000
2	800	Hipsterton	\$300,000
2	850	Normaltown	\$150,000
1	550	Normaltown	\$78,000
4	2000	Skid Row	\$150,000

Bedrooms	Sq. Feet	Neighborhood	Sale Price
3	2000	Hipsterton	???



Type of Machine Learning

- Unsupervised Learning

Bedrooms	Sq. Feet	Neighborhood
3	2000	Normaltown
2	800	Hipsterton
2	850	Normaltown
1	550	Normaltown
4	2000	Skid Row

This is kind of like someone giving you a list of numbers on a sheet of paper and saying, “I don’t really know what these numbers mean, but maybe you can figure out if there is a pattern or grouping or something—good luck!”



Type of Machine Learning

- **Semi-supervised Learning**

Semi-supervised learning is blending supervised and unsupervised learning to get better results with incomplete data. In the real estate example, you won't have a recent sale price for every house because not every house is sold every year. So you can combine data from sold houses with data from unsold houses in clever ways to come up with a more accurate housing price model. This is useful in real-world scenarios where you don't have perfect data sets.

- **Reinforcement Learning**

Reinforcement learning is training an algorithm to do a task by rewarding it when it does something right. It learns to seek out rewards and thus learns how to do a complicated task. This is how researchers teach computers how to play video games and train robots to perform complex movements. However, applying reinforcement learning is its own art that borders on voodoo and is still in the early stages of research. We will cover only very basic reinforcement learning in this workshop.



Let's write that programme

```
01 def sales_price(num_bedrooms, sqft, neighborhood):
02     price = 0
03     price_per_sqft = 200
04
05     if neighborhood == "hipsterton":
06         price_per_sqft = 400
07     elif neighborhood == "skid row":
08         price_per_sqft = 100
09
10     price = price_per_sqft * sqft
11
12     if num_bedrooms == 0: p
13         price = price - 20000
14     else:
15         price = price + (num_bedrooms * 1000)
16
17     return price
```



Let's write that programme

```
01 def sales_price(num_bedrooms, sqft, neighborhood):
02     price = 0
03
04     # a little pinch of this
05     price += num_bedrooms * .841231951398213
06
07     # and a big pinch of that
08     price += sqft * 1231.1231231
09
10     # maybe a handful of this
11     price += neighborhood * 2.3242341421
12
13     # a little extra salt for good measure
14     price += 201.23432095
15
16     return price
```



Let's write that programme

Step 1

```
01 def sales_price(num_bedrooms, sqft, neighborhood):
02     price = 0
03
04     # a little pinch of this
05     price += num_bedrooms * 1.0
06
07     # and a big pinch of that
08     price += sqft * 1.0
09
10     # maybe a handful of this
11     price += neighborhood * 1.0
12
13     # a little extra salt for good measure
14     price += 0
15
16     return price
```



Let's write that programme

Step 2:

Run every house you know about through your function and see how far off the function is at guessing the correct price for each house:

Step 3:

Repeat Step 2 over and over with every single possible combination of weights. Whichever combination of weights makes the cost closest to zero is what you use. When you find the weights that work, you've solved the problem



Training – Finding weights automatically

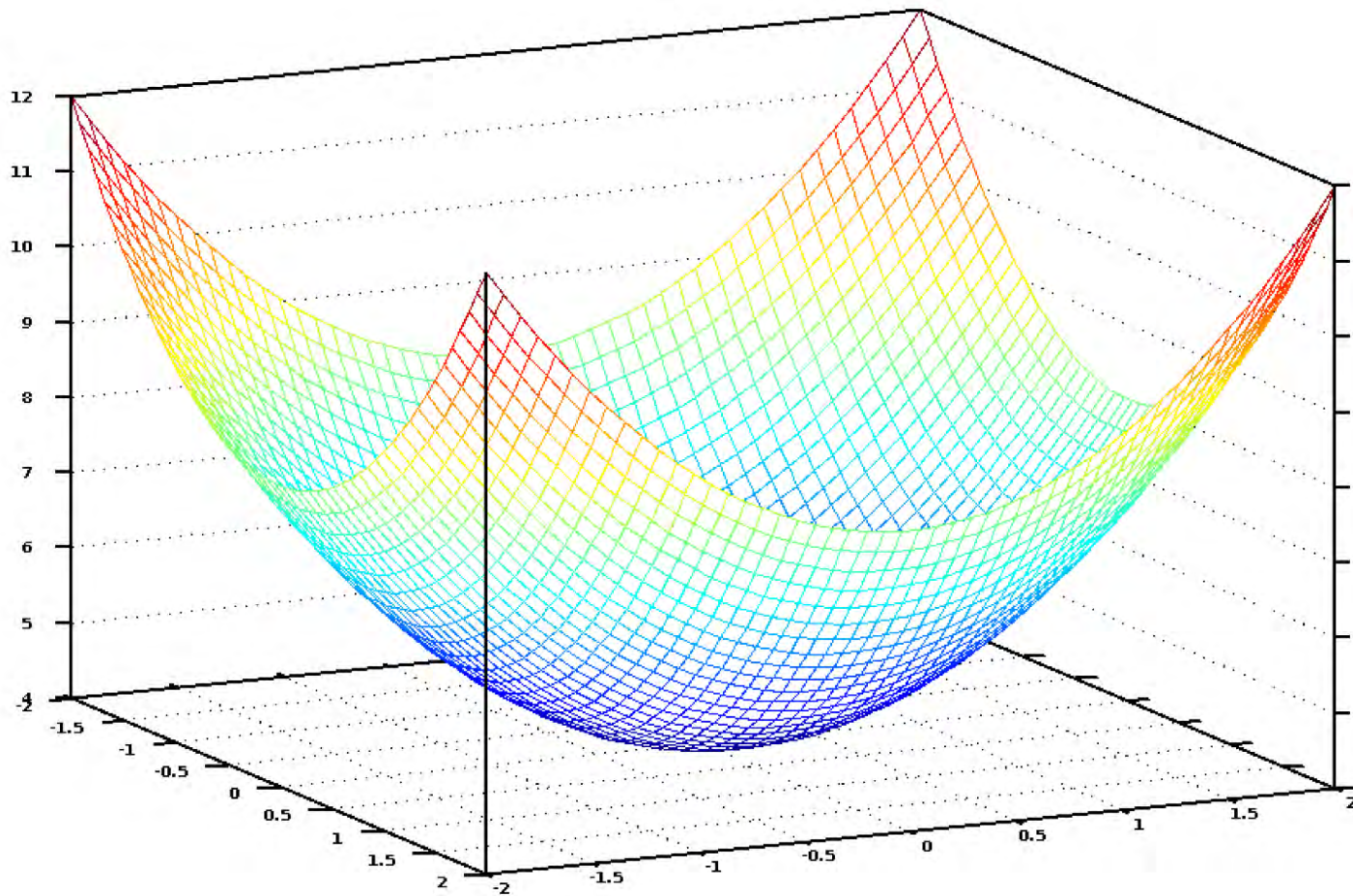
Of course, we cannot try every combination of all possible weights to find the combo that works the best. That would literally take forever, since you'd never run out of numbers to try.

What we really need is a **mathematical optimization algorithm** that can find good values for those weights without having to try every possible number. Luckily, there is a whole field of mathematics called numerical optimization and we have lots of great ways to work backward from data to find good weights.

How Gradient Descent works

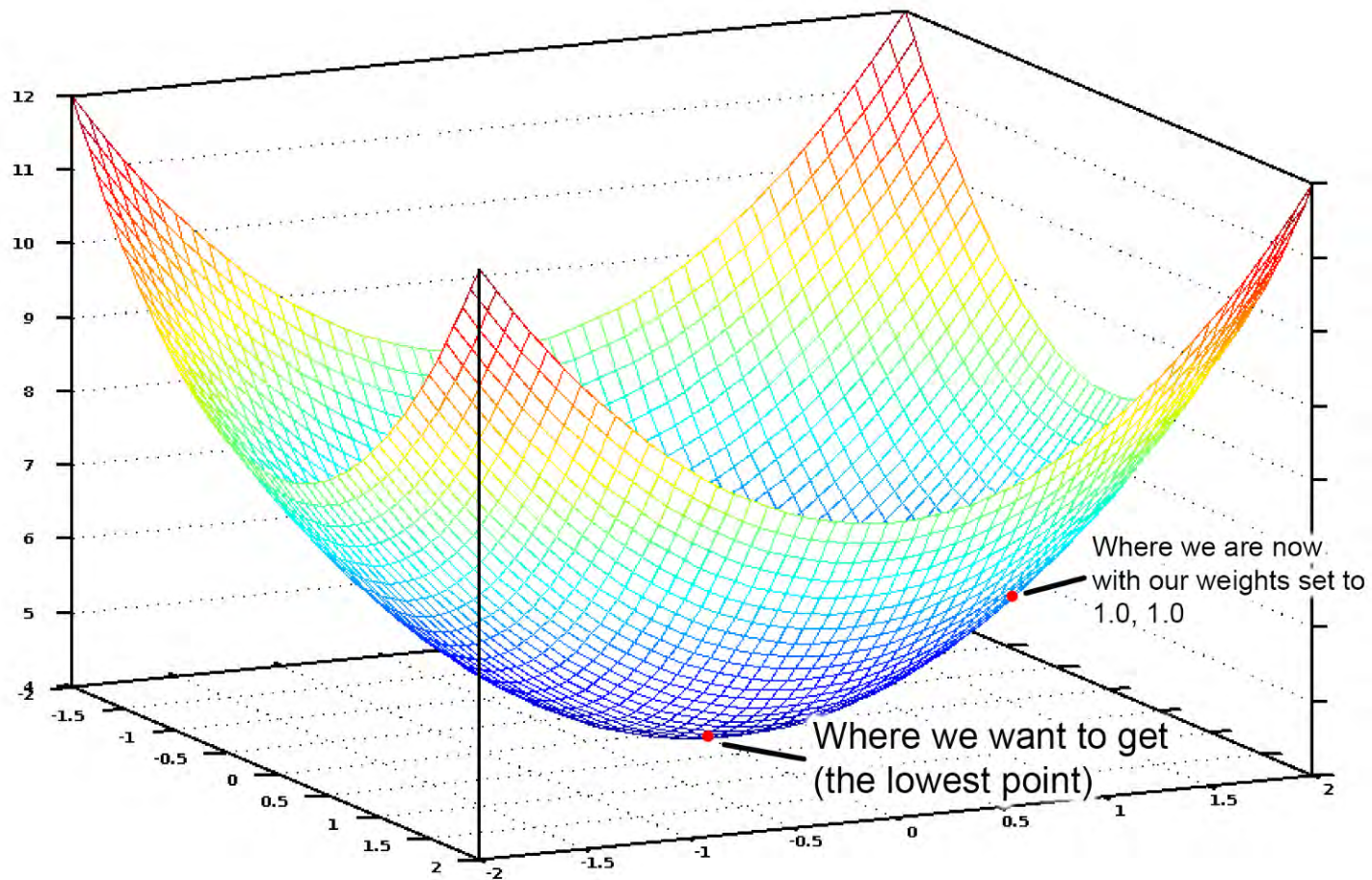
$$Cost = \frac{\sum_{i=1}^{500} (MyGuess(i) - RealAnswer(i))^2}{500 \cdot 2}$$

Gradient Descent





Gradient Descent





The Toolbox

- Choosing the right Operating System
- The Python 3 programming language
- The Python Library Ecosystem
- Speeding up Machine Learning with a GPU
- Running Models in the Cloud with Google Colab



Activity 1.1

- Training a Regression Model using Linear Regression

sq feet	num bedrooms	num bathrooms	sale price
785	2	2	170461
1477	2	2	271651
712	1	1	139912

Bedrooms	Bathrooms	Sq. Feet	Sale Price
3	2	2000	???

- Use aaip_intro_py37 virtual environment
- Libraries: pandas, scikit-learn, tensorflow (2.x)





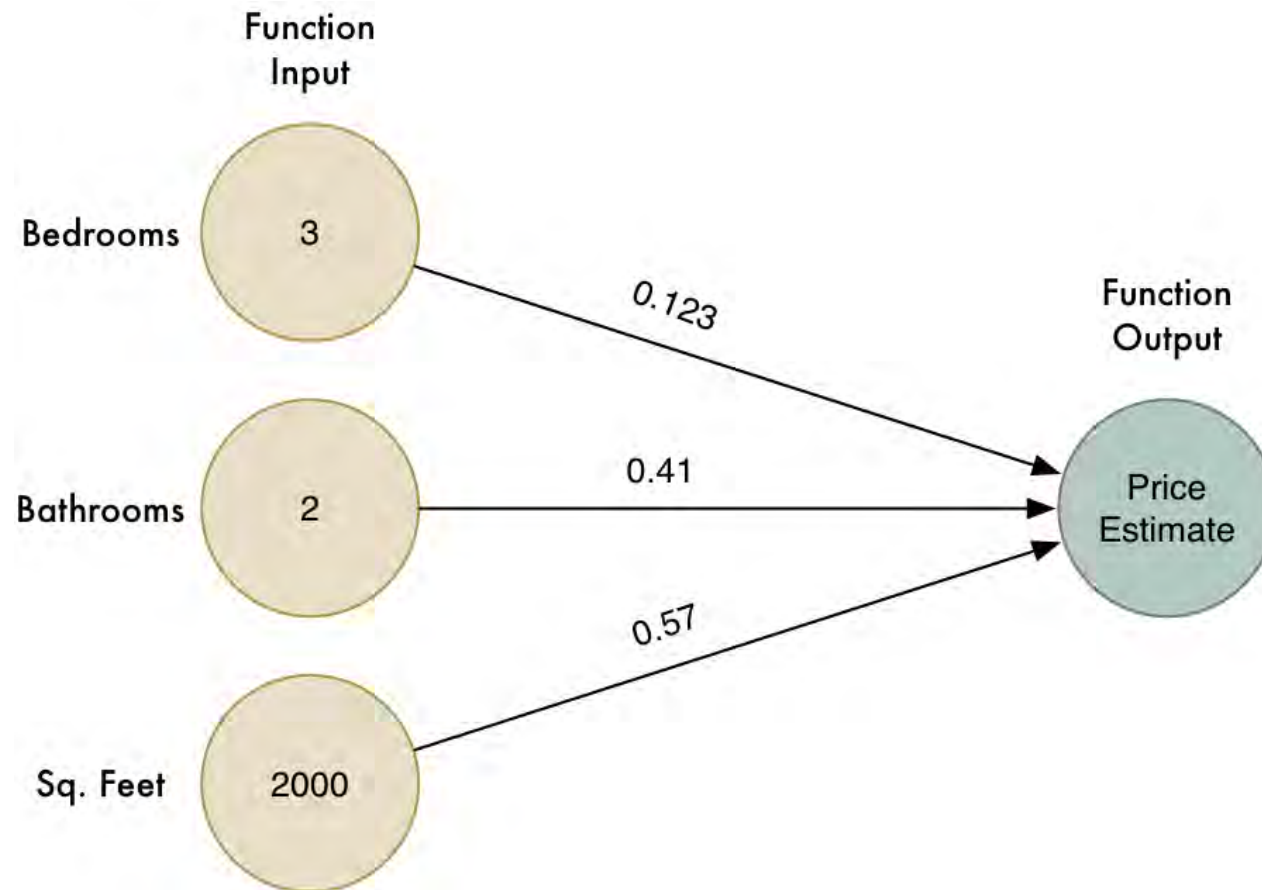
What is a Neural Network?

Is the previous activity, we created a simple estimation function:

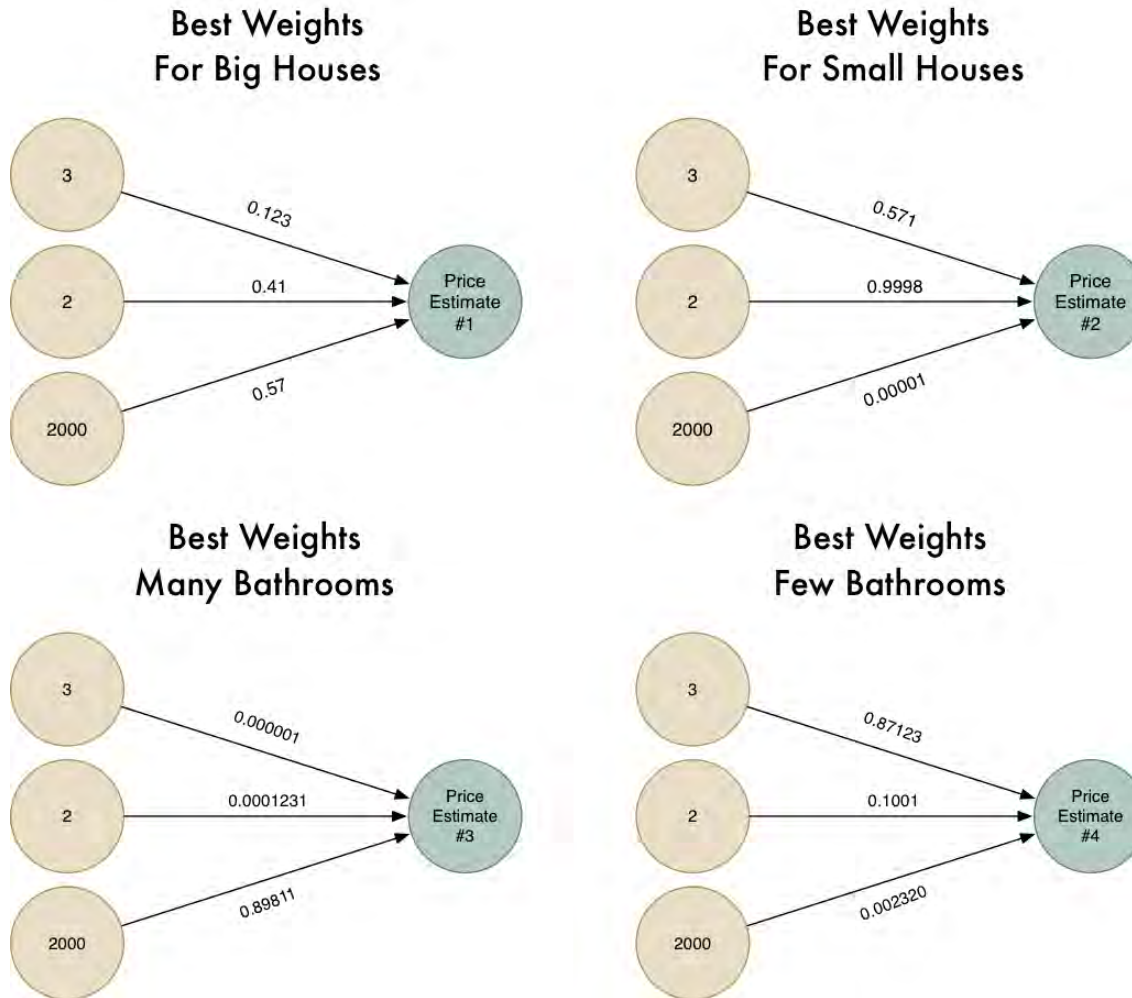
```
01 def sales_price(bedrooms, bathrooms, sqft):  
02     price = 0  
03  
04     # a little pinch of this  
05     price += bedrooms * 0.123  
06  
07     # maybe a handful of this  
08     price += sqft * 0.56  
09  
10     # a little extra salt for  
11     price += 201.23432095  
12  
13     return price
```




What is a Neural Network?

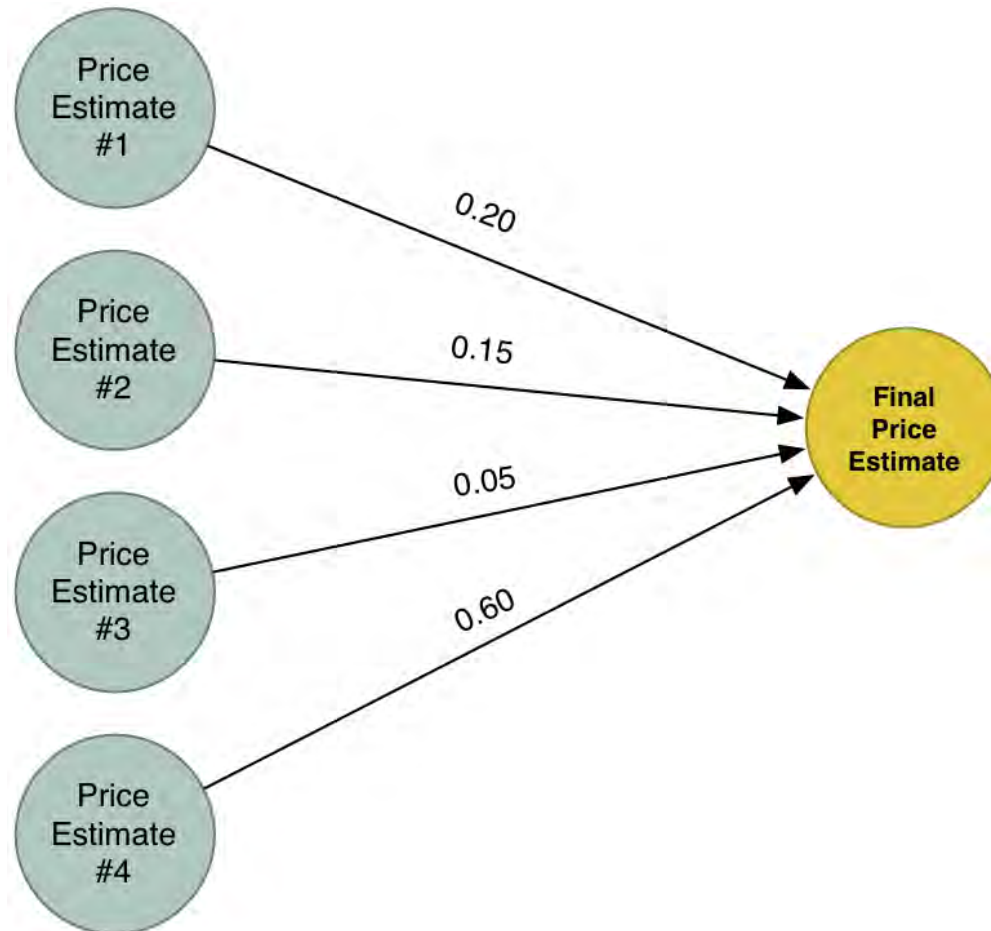


What is a Neural Network?

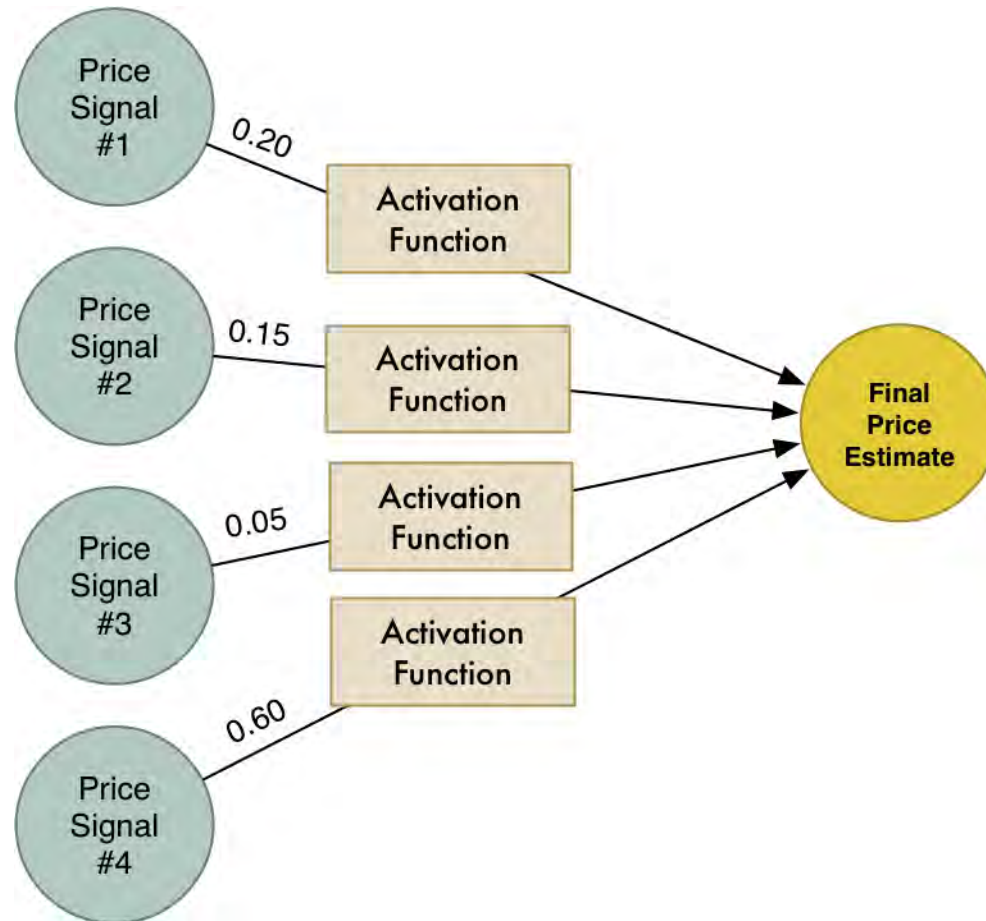




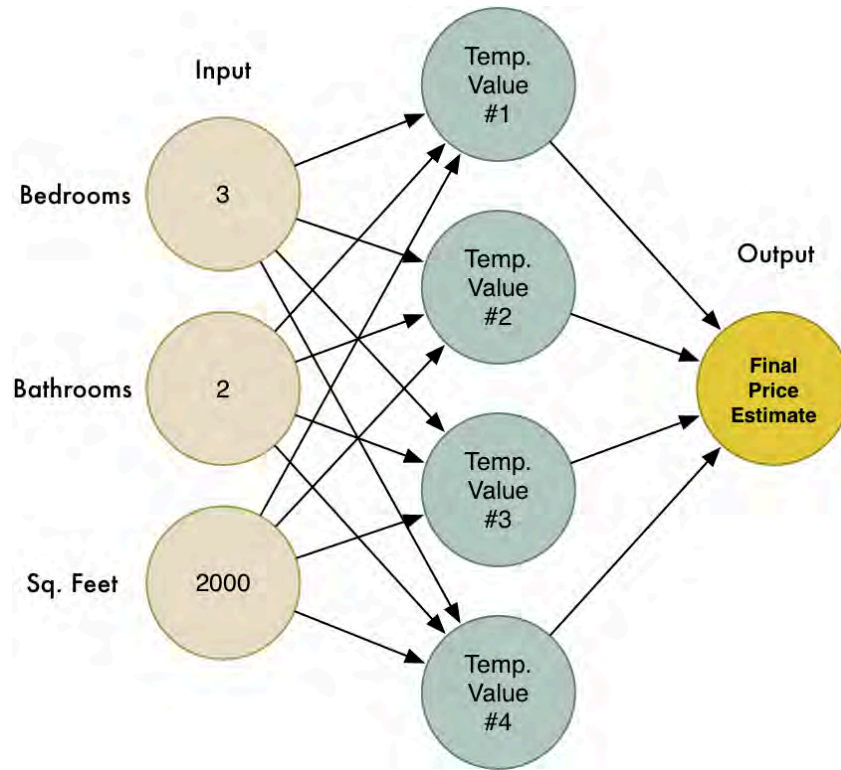
What is a Neural Network?



What is a Neural Network?



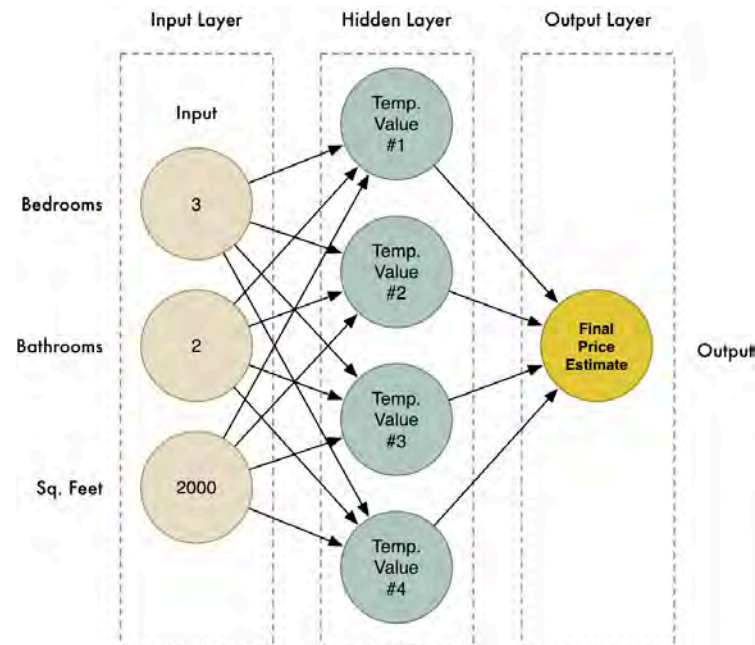
What is a Neural Network?



This is a neural network! Each node knows how to take in a set of inputs, apply a unique set of weights to them, and calculate an output value. Those output values pass through a non-linear activation function and a set of weights that makes each signal contribute more or less to the final value.

Each node is pretty simple by itself. But by chaining together lots of these nodes, we can model things that are too complicated to be modelled by one single neuron.

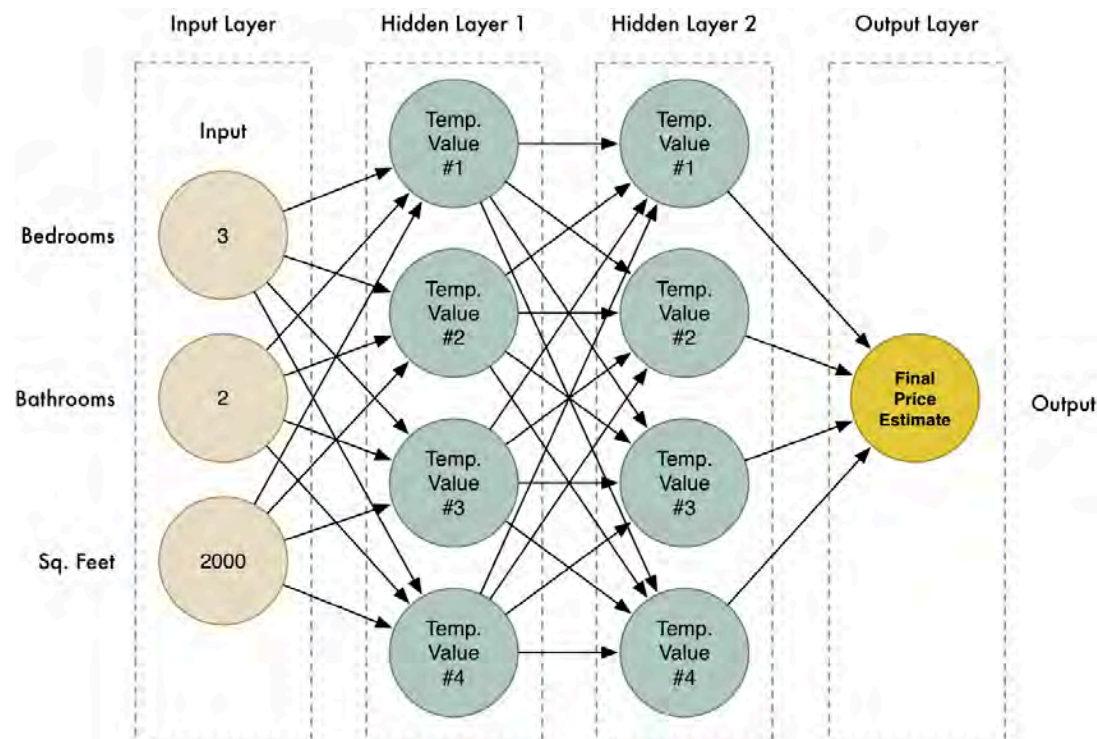
Stacking more layers



- The Input Layer has three input nodes where we pass in the input values for the current house.
- The Hidden Layer has four nodes, so it creates four price estimate signals based on the input values.
- The Output Layer has a single node, so it produces a single output.

Stacking more layers

- To make the neural network capable of modelling more complex relationships, we can add more hidden layers to it:





NN Require Feature Scaling

- Because neural networks take the initial input values as signals and recombine them over and over to produce a final value, it's really important that the input signals are all numbers that are roughly the same size.

Bedrooms	Bathrooms	Sq. Feet	Sale Price
1	1	400	\$59,000
3	2	2000	\$250,000
10	6	9500	\$4,320,000

We'll just scale the data proportionally so the smallest values in each column are zero and the largest are one.

Here is the data again after we scale it:

Bedrooms	Bathrooms	Sq. Feet	Sale Price
0.000	0.0	0.0000	0.0000
0.333	0.1	0.2197	0.0586
1.000	1.0	1.0000	1.0000

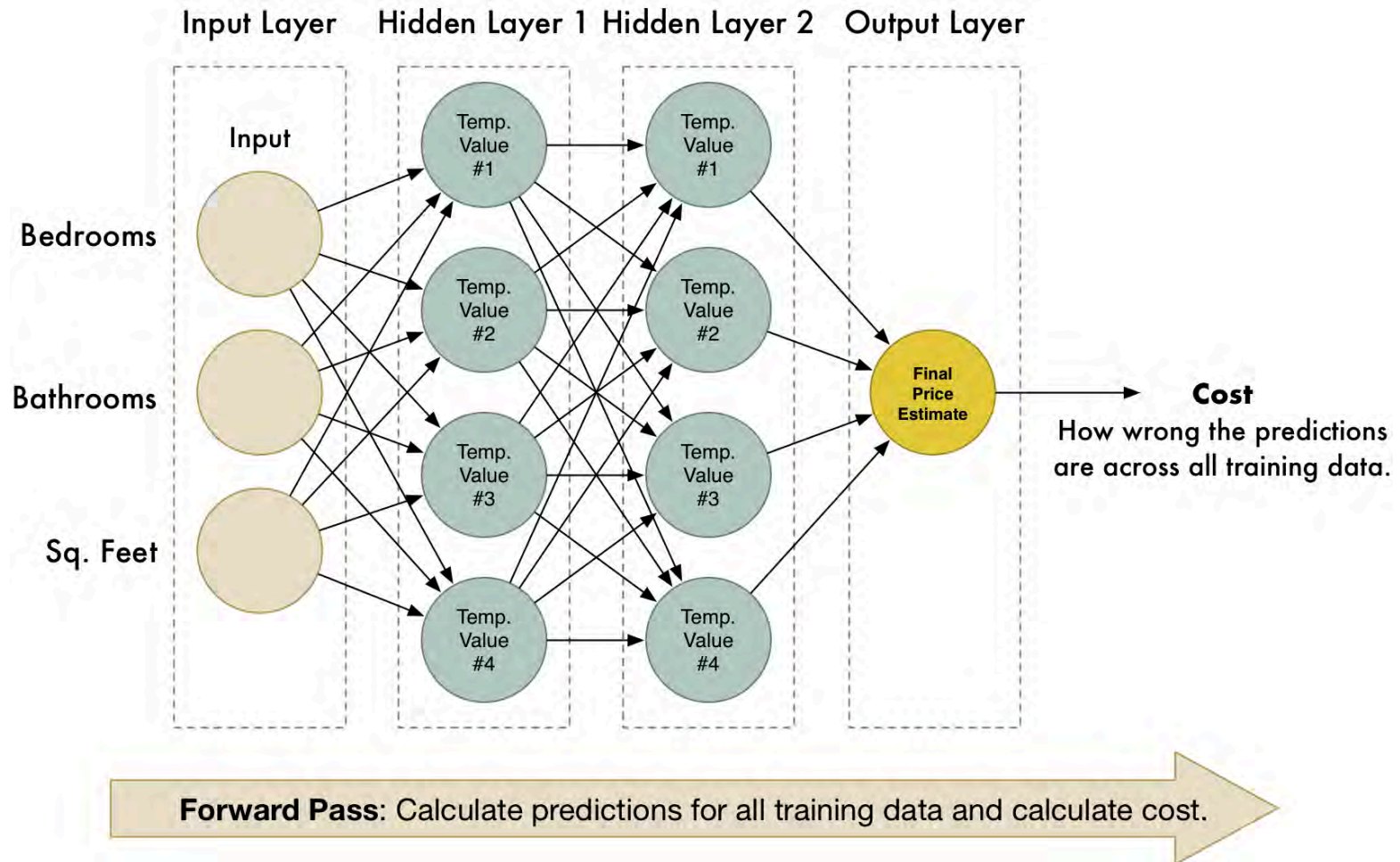


How to Train a NN

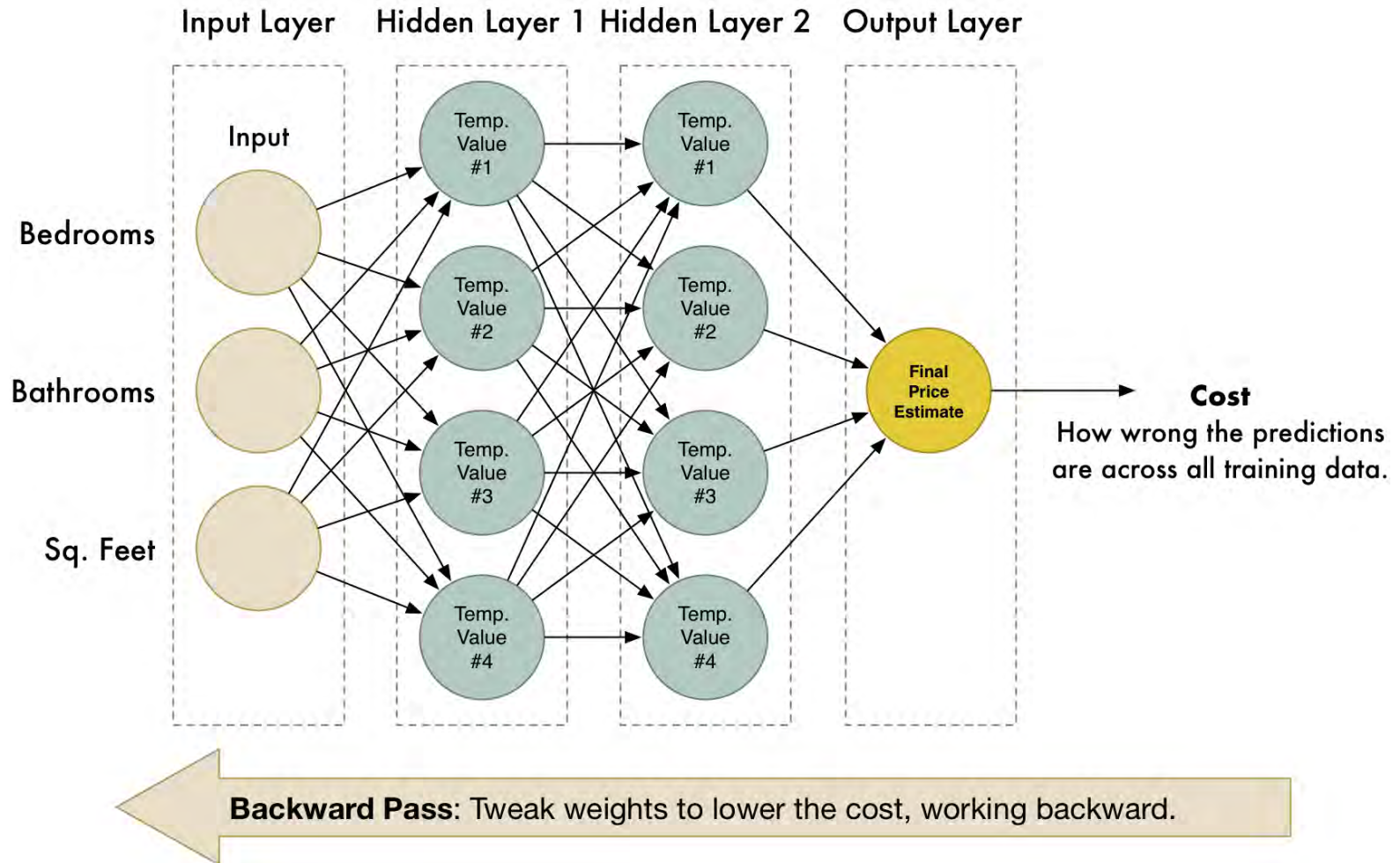
In the last section, we talked about how we find a good value for each of the weights in a linear regression model using a multi-step process:

- Set all of the weights in the model to an initial value like 1.0
- Run all of the training data through the model to get a prediction for each row of the training data.
- Use a cost function (also called a loss function) to calculate how wrong our predictions are across all the training data rows.
- Use the gradient descent algorithm to repeatedly tweak the weights in our model, which in turn adjusts our predictions with the goal of getting the cost as low as possible.

Forward pass



Backward pass



Activity 1.2

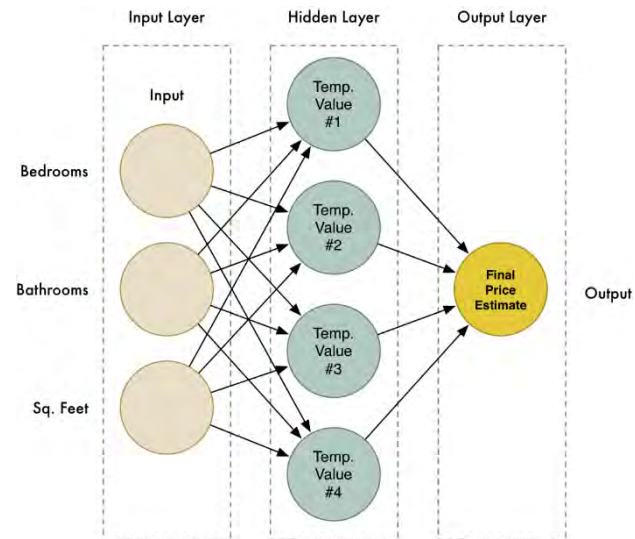
- use tensorflow & keras to train and use a neural network model to estimate the price of a house





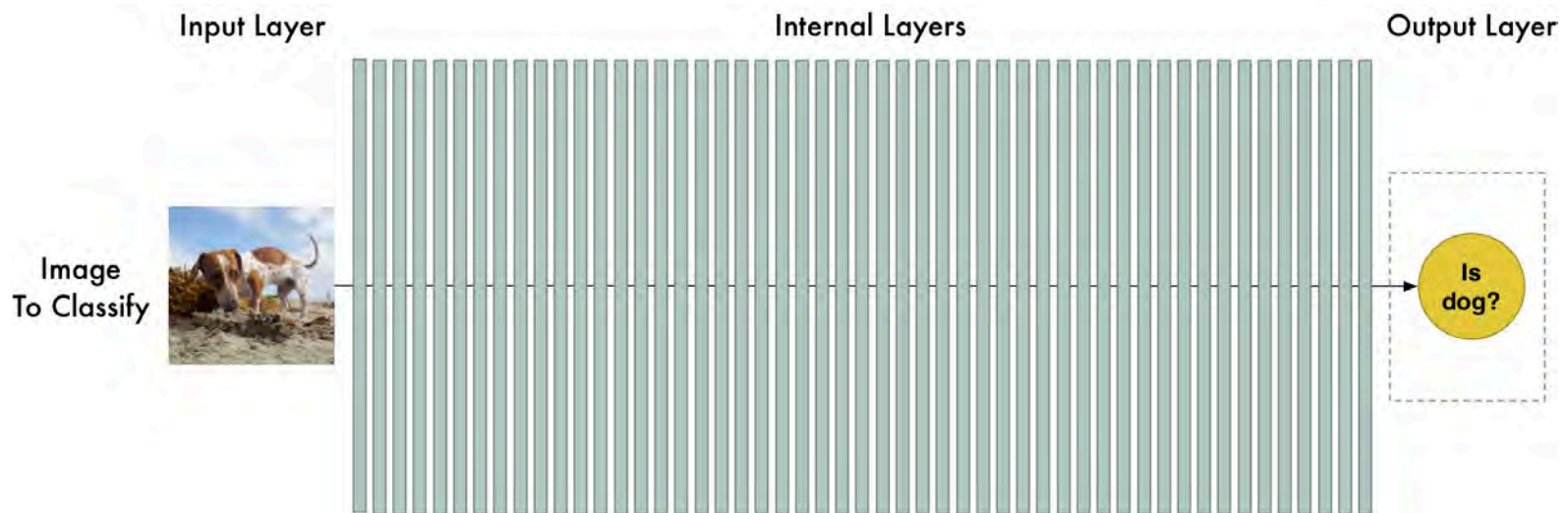
What is Deep Learning?

- **Deep learning** is one of the most futuristic-sounding buzzwords that the computer industry has ever produced. It is also one of the greatest gifts ever given to software sales teams because they can make it seem like they are selling futuristic alien technology. But it's really not that complicated!
- We learned earlier that neural networks are a kind of machine learning model where you can add multiple layers to make them capable of solving more complex problems.



What is Deep Learning?

- With deep learning we stack lots and lots of layers to build models that can learn to solve really complex problems. For example, it's common to use a 50-layer or 100-layer neural network for image recognition.





Does Deep Learning Count as AI?

Today, the terms deep learning and AI are often used interchangeably. But is deep learning really artificial intelligence? Absolutely not—at least not in the way that most people think of it.

When most people think of artificial intelligence, they think of androids that can speak and think like humans. This is what researchers call Strong AI. But deep learning so far has not produced anything that is even close to that. Deep learning is really just the ability to build statistical models using so much training data that they appear to have intelligence. But they don't really have any intelligence. They are just complicated pattern matchers.

No one knows yet whether deep learning will help us get to Strong AI or whether it's a dead end. Any company that claims to have an intelligent robot or tries to get their product declared a “robot citizen” is a company that deserves a lot of scepticism.



Why Deep Learning is so useful?

The big advantage of deep learning models is that they have much more capacity for learning from data than traditional machine learning models. With a traditional machine learning model, it will hit a point where more training data doesn't help improve accuracy. With deep learning, the model is able to keep learning from more data for a much longer time.

A great example is image recognition. To build a model that can accurately recognize a bird, the model also has to be able to differentiate a bird from all the other millions of kinds of objects that could appear in a picture. Traditional machine learning models just don't have the capacity to solve a problem that complex. Showing a traditional model 100 million images probably won't improve its performance at all over 10 million images. But with a deep neural network, it can.



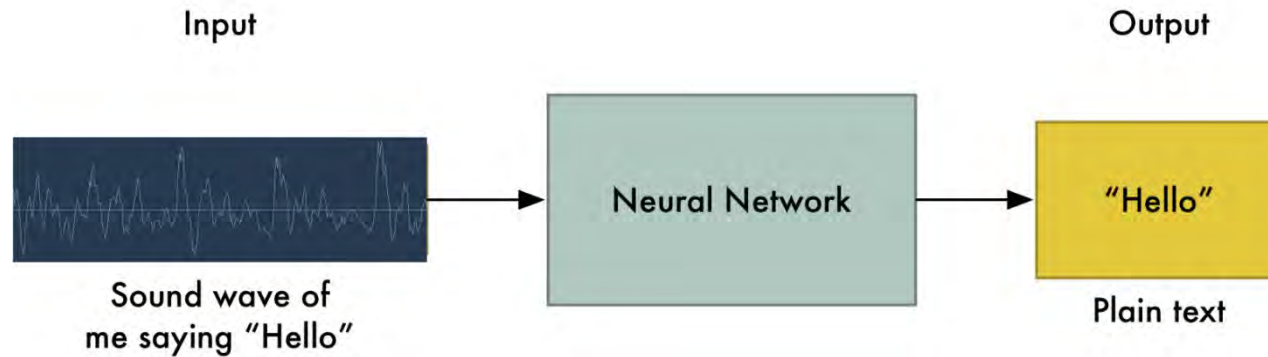
Quiz

<http://bit.ly/2Y8MTi2>



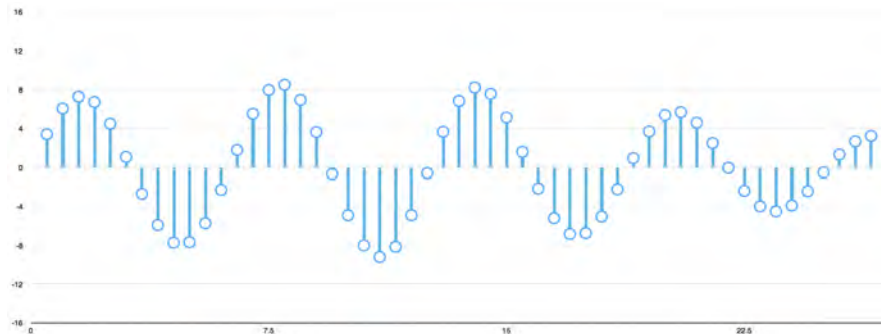
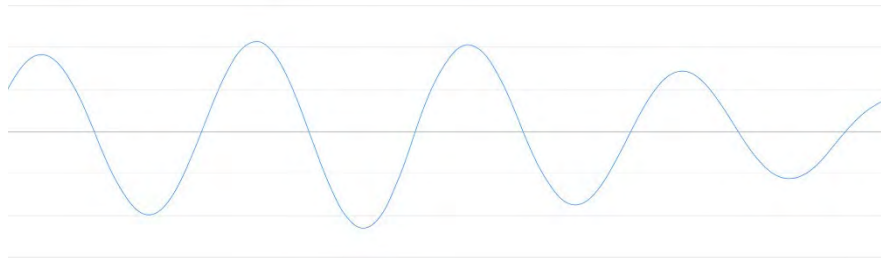
Speech Recognition

- Speech Recognition Is (Still) Difficult



Speech Recognition

- **Turning Sounds into Bits**





Speech Recognition

- **Pre-Processing Sampled Sound Data**
 - To make this data easier for a neural network to process, we are going to break apart this complex sound wave into its component parts. We'll pull out the low-pitched parts, the next-lowest-pitched parts, and so on. Then by adding up how much energy is in each of those frequency bands (from low to high), we create a fingerprint of sorts for this audio snippet.

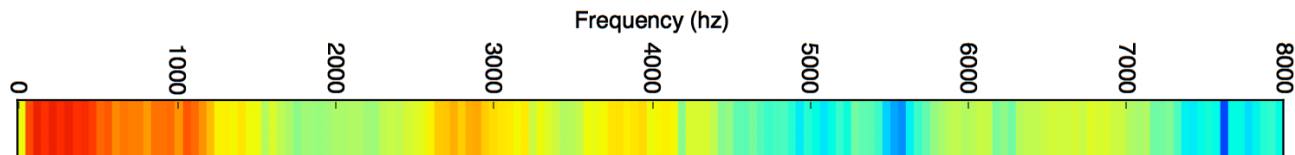


Speech Recognition

Imagine you had a recording of someone playing a C Major chord on a piano. That sound is the combination of three musical notes—C, E, and G— all mixed together into one complex sound. We want to break apart that complex sound into the individual notes to discover that they are C, E, and G. This is the exact same idea.

We do this using a mathematic operation called a **Fourier transform**. It breaks apart the complex sound wave into the simple component sound waves that make it up. Once we have those individual sound waves, we add up how much energy is contained in each one.

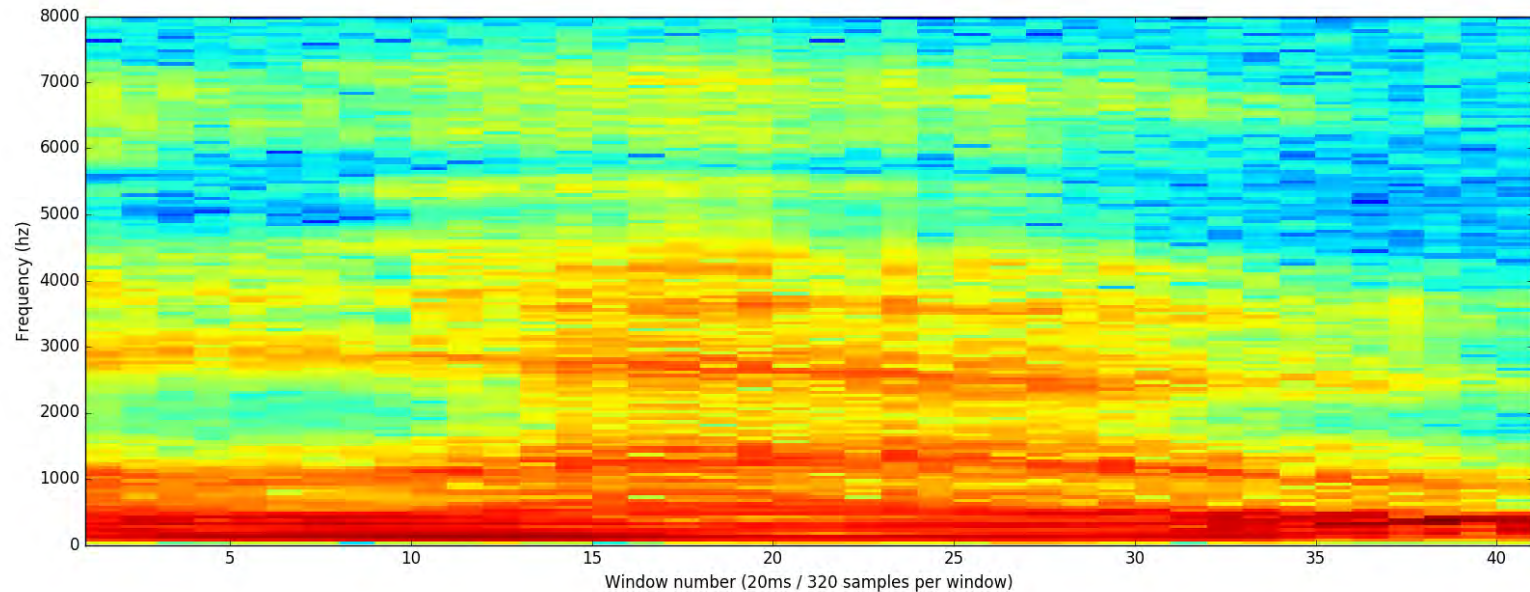
The result is a score of how important each frequency range is, from low pitch (i.e., bass notes) to high pitch. Each coloured line below represents how much energy was in that 50hz band of our 20-millisecond audio clip.



Speech Recognition

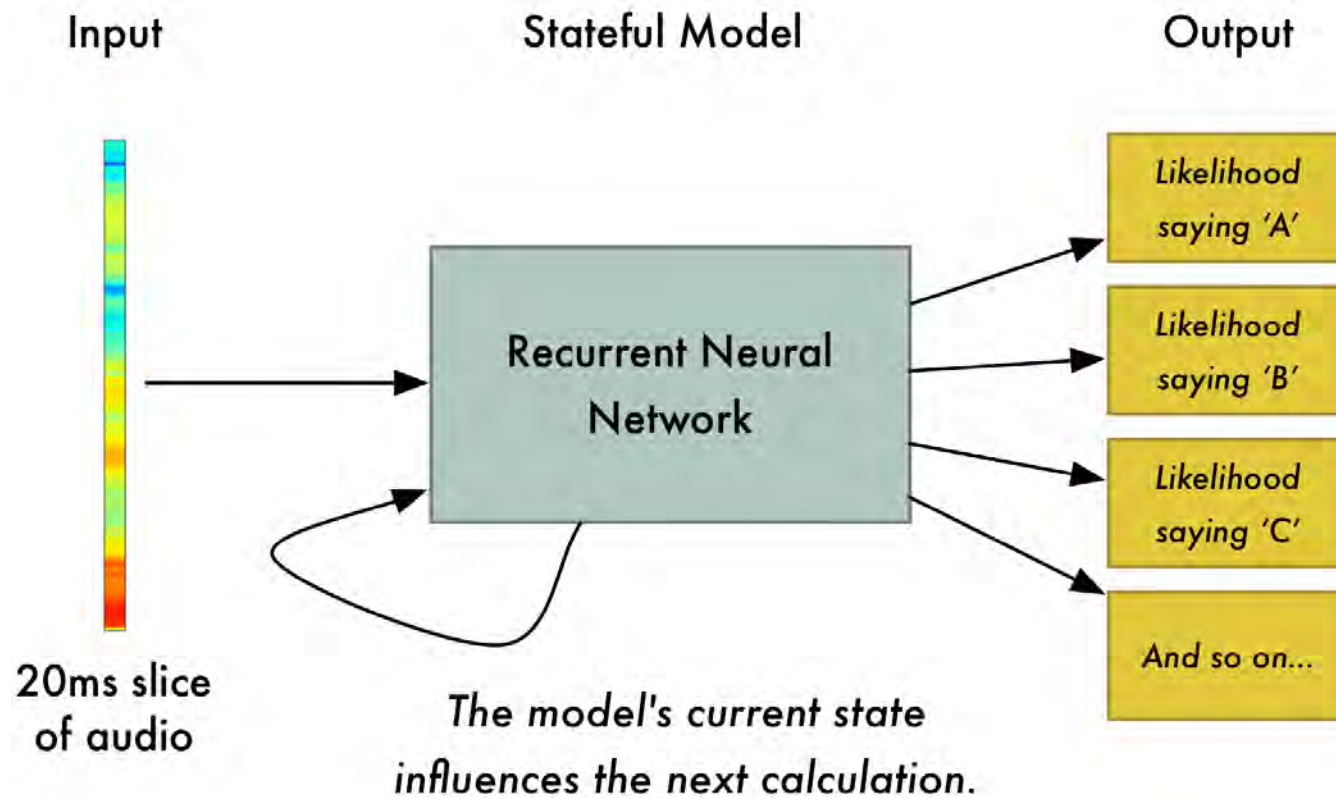
You can see that there's a lot of energy in the lower frequencies closer to zero and dense areas around 3000hz and 4000hz.

If we repeat this process on every 20-millisecond chunk of audio we have in our sound clip, we'll end up with a spectrogram. (Each column from left to right is one 20ms chunk.)



Speech Recognition

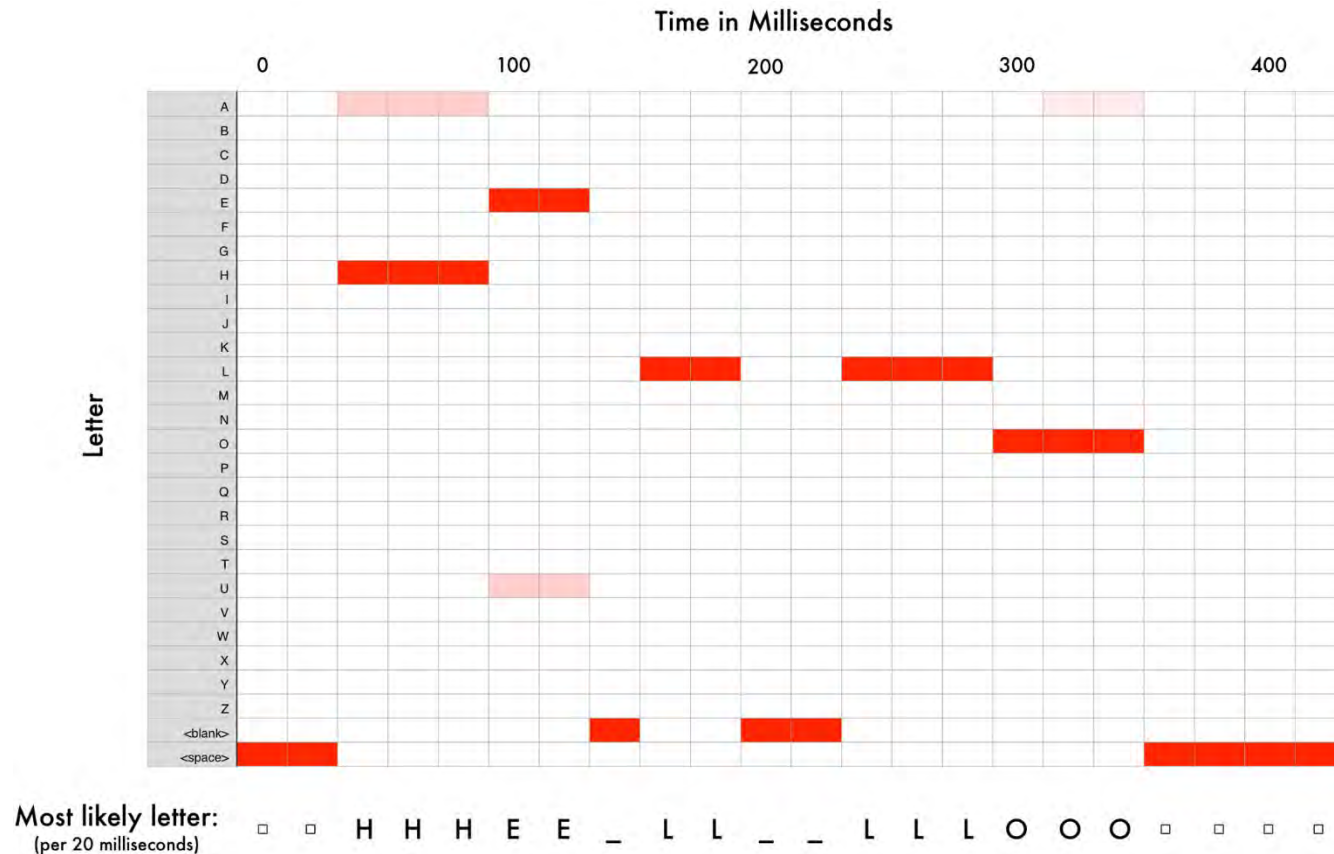
Recognizing Characters from Short Sounds.
connectionist temporal classification or CTC





Speech Recognition

Hello



Activity 1.3

- **Recognising Speech with DeepSearch**

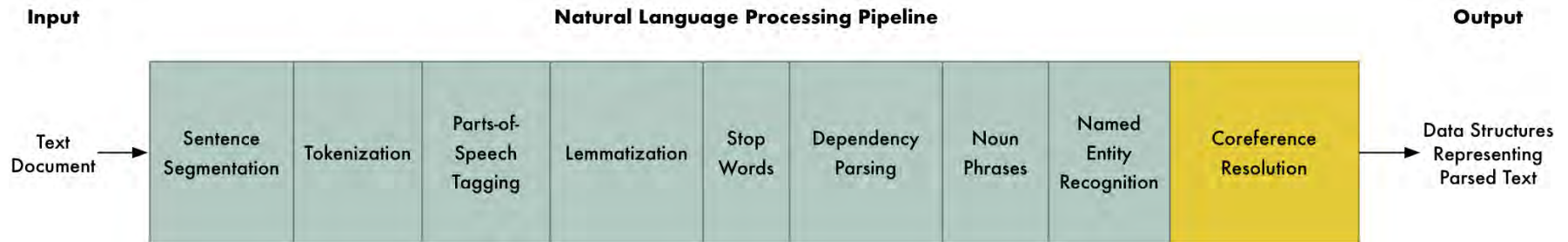


- Use aaip_sr_py37 virtual environment
- `pip install deepspeech==0.5.1`



Natural Language Processing

NLP pipeline with spacy





NLP

London is the capital and most populous city of England and the United Kingdom. Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia. It was founded by the Romans, who named it Londinium.



Step 1: Sentence Segmentation

London is the capital and most populous city of England and the United Kingdom. Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia. It was founded by the Romans, who named it Londinium.

“London is the capital and most populous city of England and the United Kingdom.”

“Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia.”

“It was founded by the Romans, who named it Londinium.”



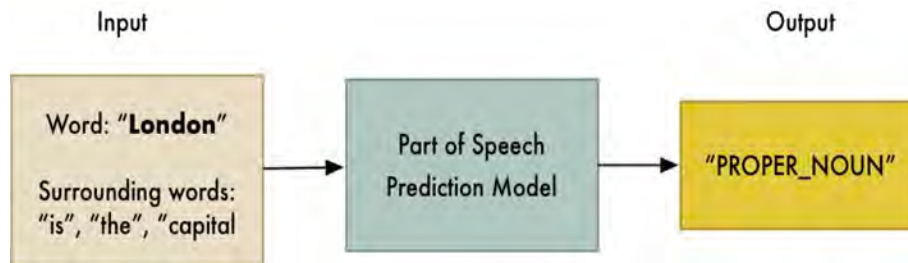
Step 2: Word Tokenization

“London is the capital and most populous city of England and the United Kingdom.”

“London”, “is”, “ the”, “capital”, “and”, “most”,
“populous”, “city”, “of”, “England”, “and”, “the”,
“United”, “Kingdom”, “.”



Step 3: Predicting Parts of Speech for Each Token



London	is	the	capital	and	most	populous ...
Proper Noun	Verb	Determiner	Noun	Conjunction	Adverb	Adjective



Step 4: Text Lemmatization

I had a **pony**.

I had two **ponies**.

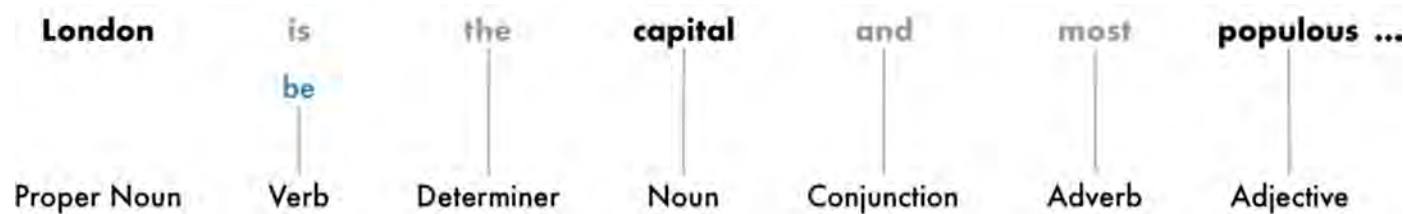
In NLP, we call finding this process **lemmatization**—figuring out the most basic form or lemma of each word in the sentence. The same thing applies to verbs. We can also lemmatize verbs by finding their root, unconjugated form. So “**I had two ponies**” becomes “**I [have] two [pony]**”.





Step 5 : Identifying the Stop Words

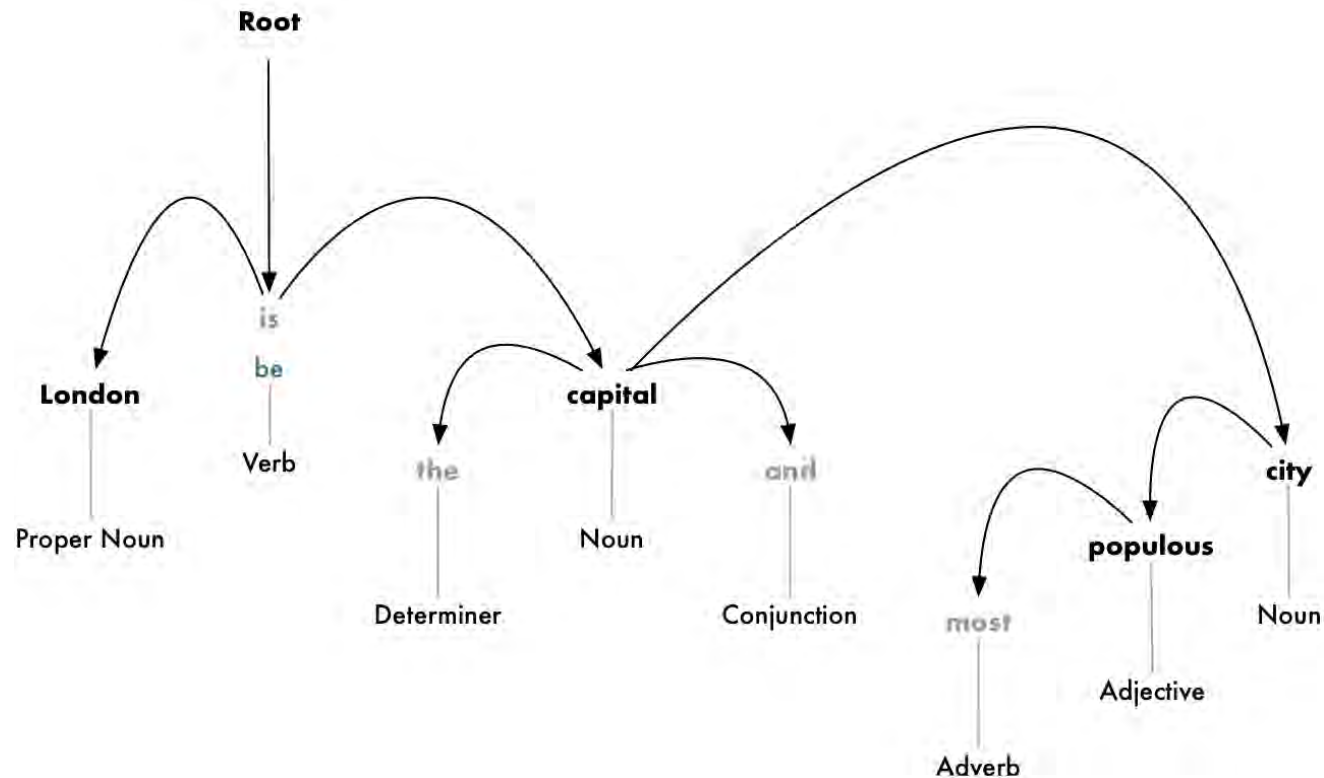
Next, we want to consider the importance of each word in the sentence. English has a lot of filler words that appear very frequently like “and”, “the”, and “a”. When doing statistics on text, these words introduce a lot of noise since they appear way more frequently than other words. Some NLP pipelines will flag them as stop words—that is, words that you might want to filter out before doing any statistical analysis.





Step 6 : Dependency Parsing

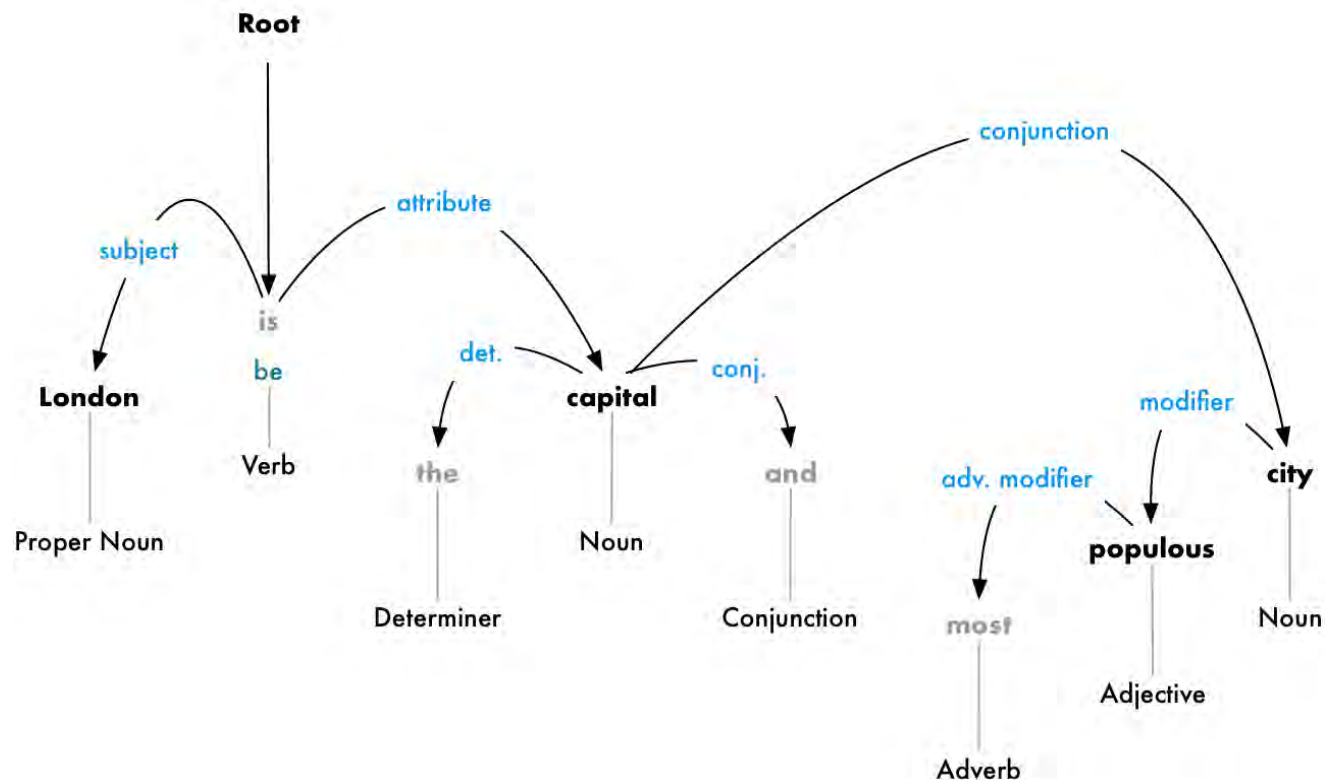
The next step is to figure out how all the words in our sentence relate to each other. This is called dependency parsing.





Step 6 : Dependency Parsing

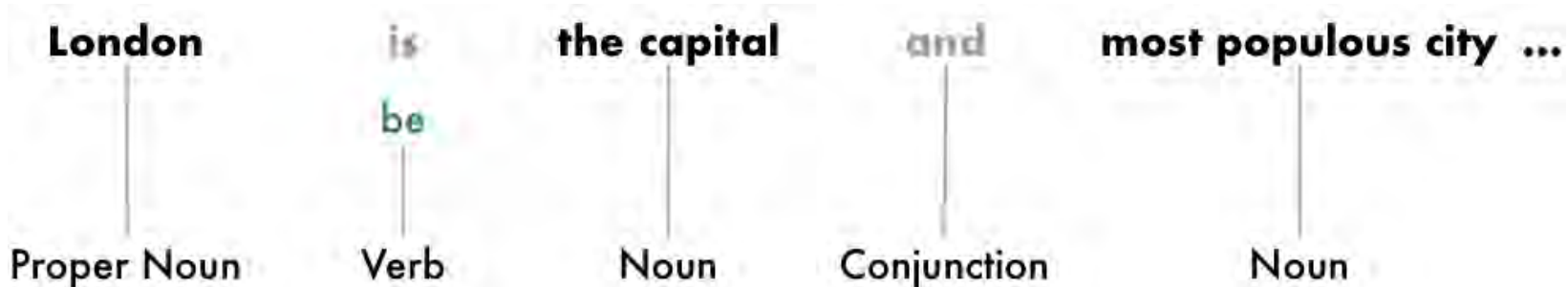
But we can go one step further. In addition to identifying the parent word of each word, we can also predict the type of relationship that exists between those two words.





Step 6a : Finding Noun Phrases

So far, we've treated every word in our sentence as a separate entity. But sometimes it makes more sense to group together the words that represent a single idea or object. We can use the information from the dependency parse tree to automatically group together words that are all talking about the same thing. For example, instead of this:





Step 7: Named Entity Recognition

London is the capital and most populous **city** of **England** and the **United Kingdom**.

The goal of **named entity recognition**, or **NER**, is to detect and label these nouns with the real-world concepts that they represent. Here's what our sentence looks like after running each token through our NER tagging model:

London is the capital and most populous city of **England** and the **United Kingdom**.

Geographic
Entity

Geographic
Entity

Geographic
Entity



Step 8: Coreference Resolution

is the capital and most populous city of England and the United Kingdom. Standing on the River Thames in the south east of the island of Great Britain, **London** has been a major settlement for two millennia. **It** was founded by the Romans, who named it Londinium.

With coreference information combined with the parse tree and named entity information, we should be able to extract a lot of information out of this document!

Coreference resolution is one of the most difficult steps in our pipeline to implement. It's even more difficult than sentence parsing. Recent advances in deep learning have resulted in new approaches that are more accurate, but it isn't perfect yet



Activity 1.4

- Extracting Entities from text
- Building a Data Scrubber
- Extracting Facts from Text
- Extracting Noun Chunks

spaCy



Activity 1.5

- Training and Using a Text Classifier
- Using Classification Models to Extract Meaning
- Training and Using a Text Classifier
- Writing the Data Preparation Script
- Training the Model
- Testing the Model
- Iterating on the Model
- Using the Model in a Program



*fast*Text



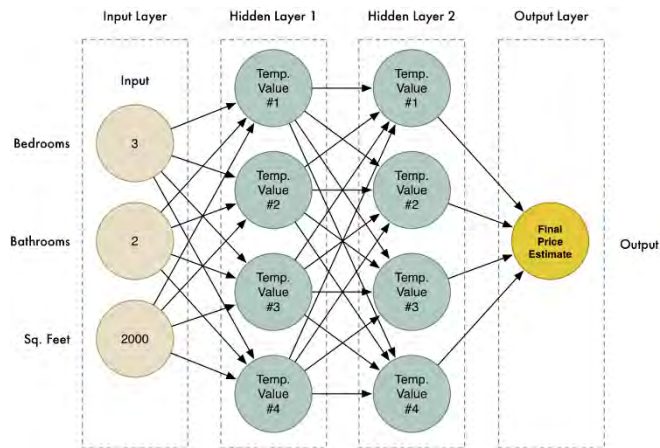
Quiz

<http://bit.ly/2Y8MTi2>



Computer Vision

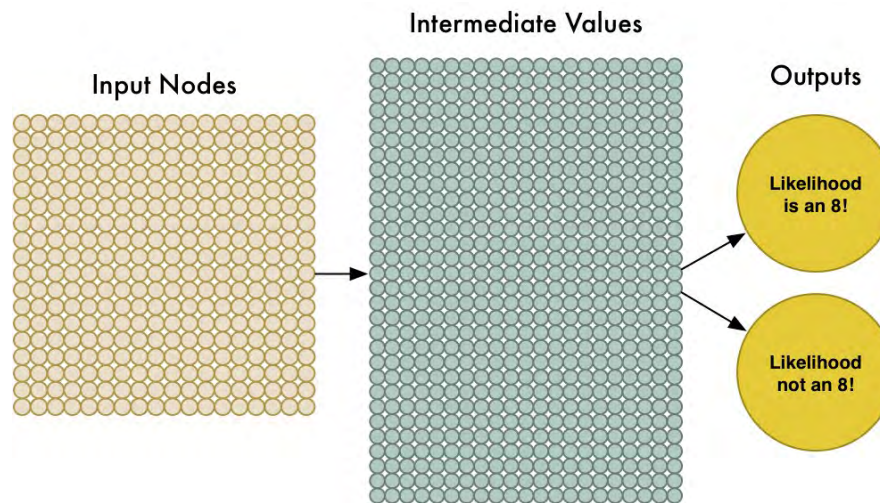
- **Simple Image Recognition - CNN**

[illegible][illegible]

An image is really just a grid of numbers that represent the darkness or intensity of each pixel:

Computer Vision

To handle 324 input numbers, we'll enlarge our neural network to have 324 input nodes:



Computer Vision


We can train a neural network in only a few minutes on a modern laptop. When it's done, it will be able to recognize pictures of "8"s with a pretty high accuracy



Computer Vision

- Recognizing more complex Images


Test Image #1



Prediction from our network

100% an "8"!


Test Image #1



Prediction from our network

No idea?!


Test Image #2



Prediction from our network

100% not an "8"!

Test Image #2



Prediction from our network

What is this?!@

Brute Force Idea #1: Searching with a Sliding Window

Test Image



Prediction from our network

100% an "8"!

Brute Force Idea #2: More Data and a Deep Neural Net

Original Training Image



More training images generated by a script



Convolution for Translational Invariance



As a human, you instantly recognize the hierarchy in this picture:

- The ground is covered in grass and concrete
- There is a child
- The child is sitting on a bouncy horse
- The bouncy horse is on top of the grass

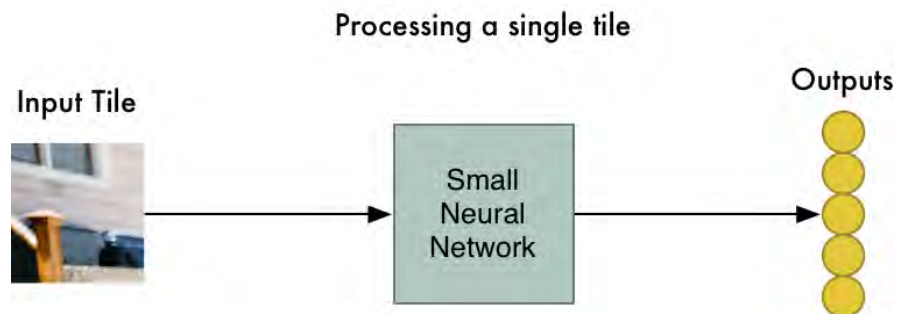
We need to give our neural network understanding of translation invariance—an “8” is an “8” no matter where it shows up in the picture. We’ll do this using a process called **convolution**. The idea of convolution is inspired partly by computer science and partly by biology

How Convolution Works

- Break the image into overlapping image tiles

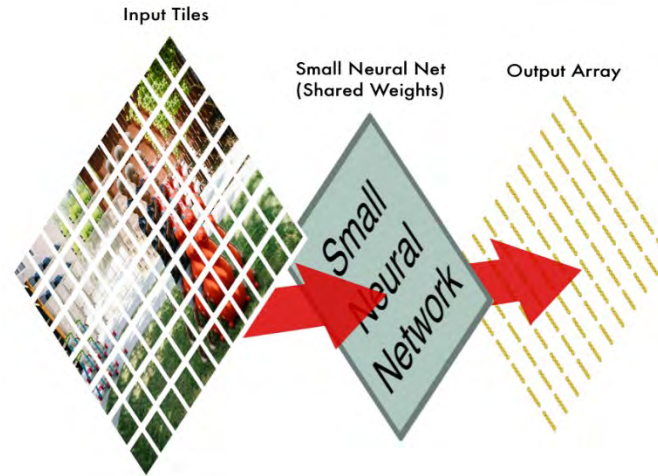


- Feed each image tile into a small neural network



How Convolution Works

- Save the results from each tile into a new array



We can repeat steps 2 and 3 as many times as we like. Each time we do it, we're training the neural network to try to find an important pattern in the image data. By doing it multiple times with the same input tiles, we can train each of the small neural networks to look for different patterns that might be important to the final classification.

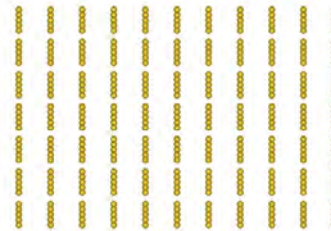
How Convolution Works

- Normalizing the data again
 - Whenever we feed data into a neural network, we have to normalize that data so that all the values are between 0 and 1.
- Downsampling with max pooling

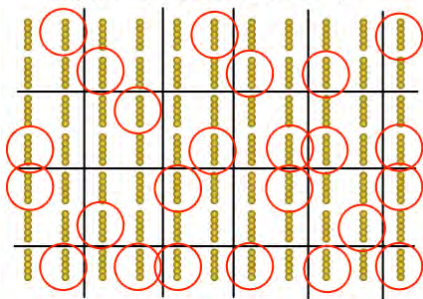
Original Input Image



Array resulting from convolution in Step 3



Find the max value in each grid square in our Array



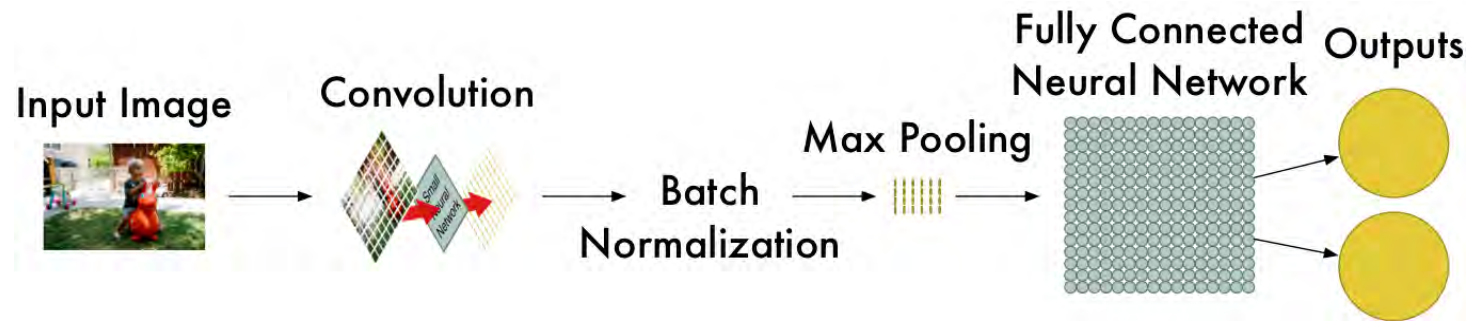
Max-pooled array



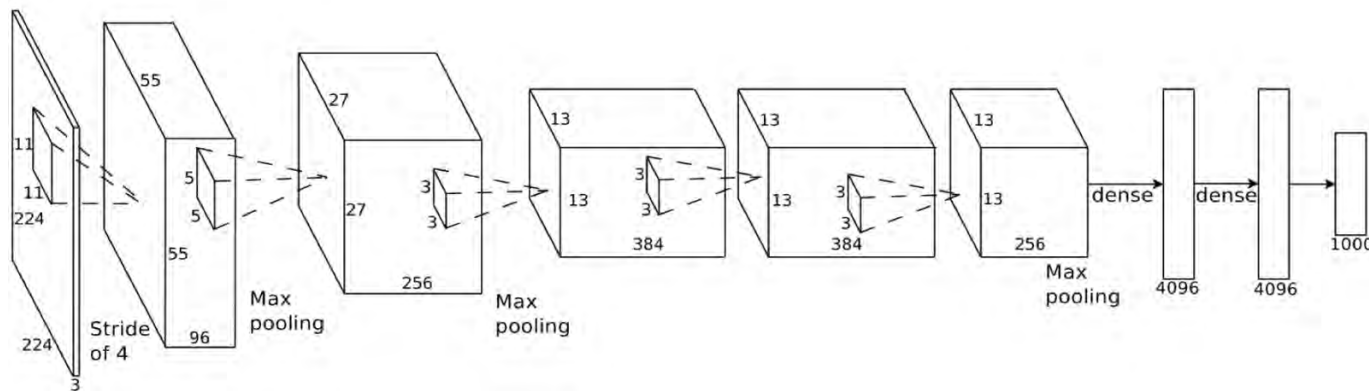
The idea here is that if we found something interesting in any of the four input tiles that make up each 2x2 grid square, we'll just keep the most interesting bit. This reduces the size of our array while keeping the most important bits.

How Convolution Works

- Final Step: Make a prediction



- Adding even more steps





How to Design a Neural Network for Maximum Accuracy

- VGG—The Foundational Design
- ResNet—A Solution for Deeper Networks
- Inception-Based Designs
- ResNeXt: The Best of ResNet and Inception-Based Designs
- Mobile-Focused Designs

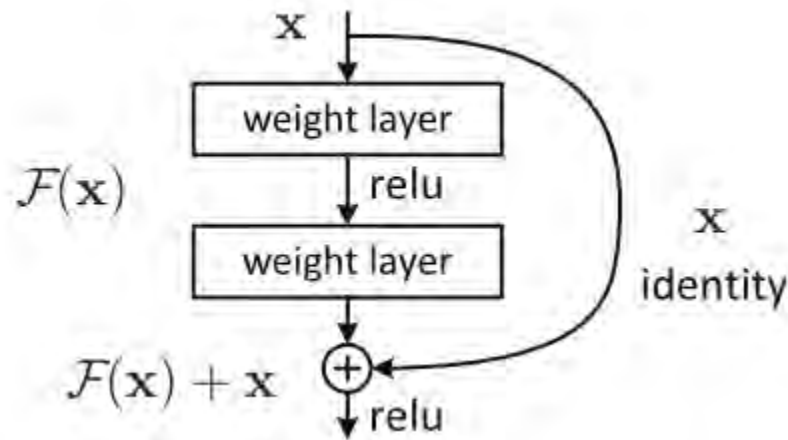
How to Design a Neural Network for Maximum Accuracy

- VGG—The Foundational Design



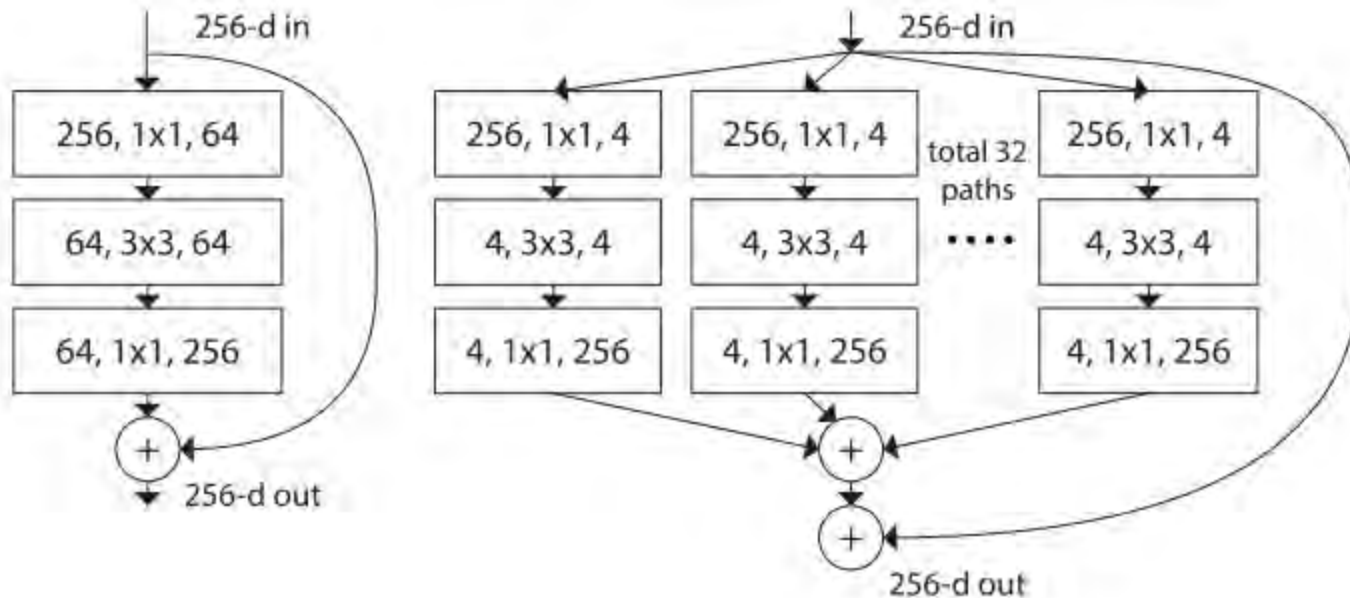
How to Design a Neural Network for Maximum Accuracy

- ResNet—A Solution for Deeper Networks
- The core idea of ResNet is introducing a so-called “identity shortcut connection” that skips one or more layers, as shown in the following figure:



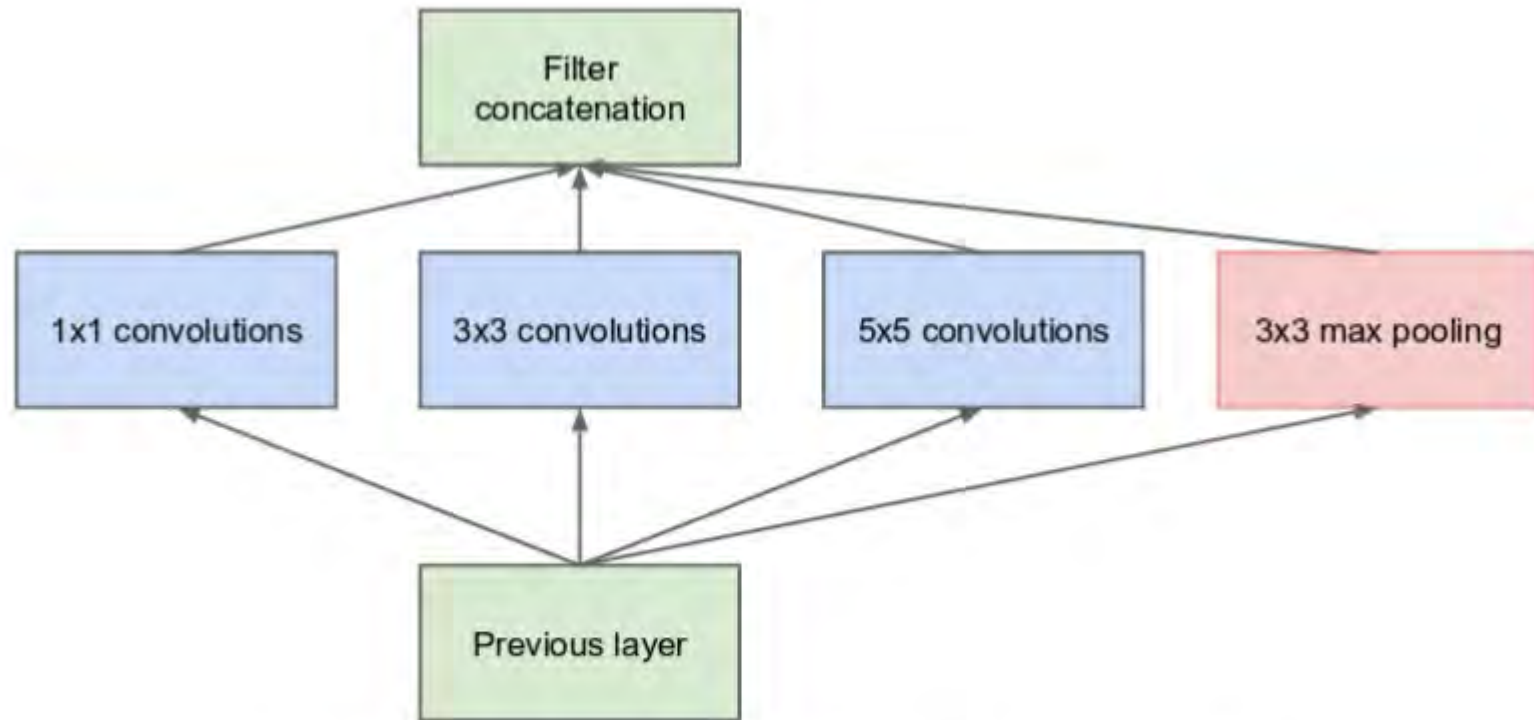
How to Design a Neural Network for Maximum Accuracy

- ResNeXt: The Best of ResNet and Inception-Based Designs



How to Design a Neural Network for Maximum Accuracy

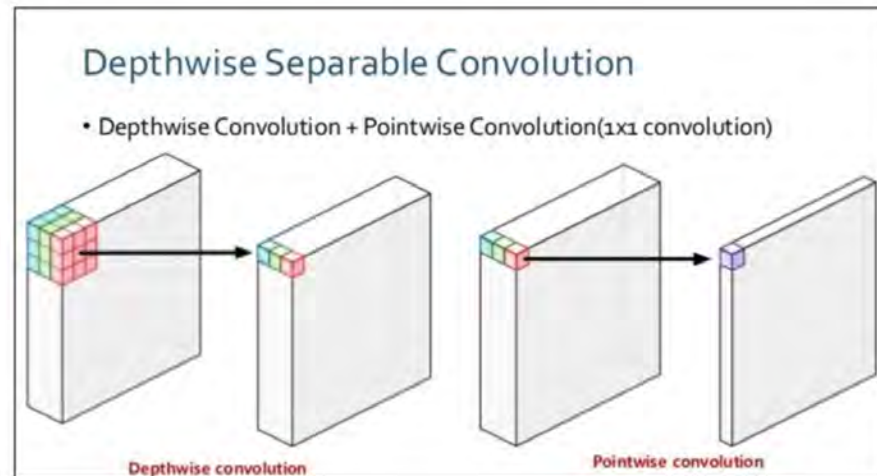
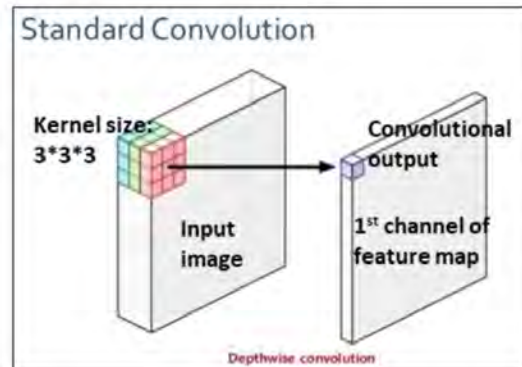
- Inception-Based Designs



(a) Inception module, naïve version

How to Design a Neural Network for Maximum Accuracy

- Mobile-Focused Designs - very less computation power to run





Activity 2.1 - Building a Bird Classifier with CNN

- Getting the Training Data
- Coding the Model
- Defining the Neural Network
- Training the Neural Network
- Testing the Neural Network
- How accurate is 95 Percent Accurate





Evaluations

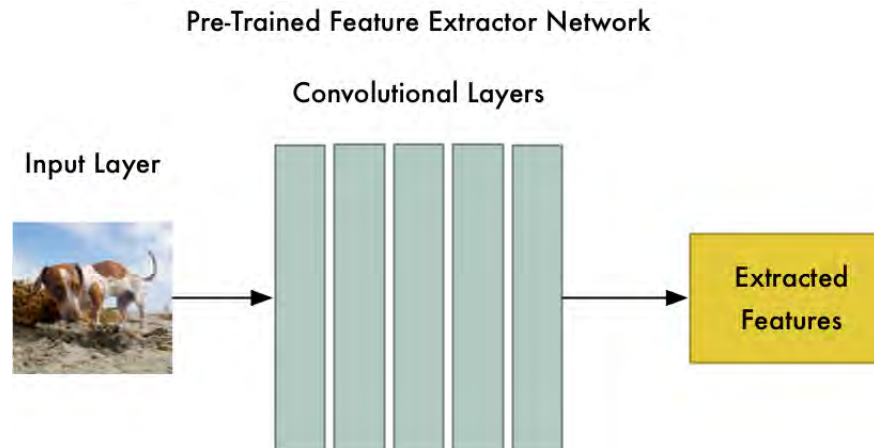
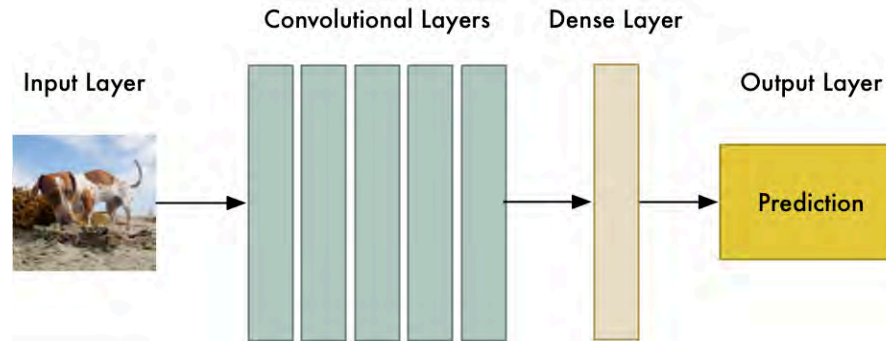
Class	Predicted 'Bird'	Predicted 'Not Bird'
Bird	549 True Positives	451 False Negatives
Not Bird	48 False Positives	8952 True Negatives

Instead of just looking at overall accuracy, we can use these measurements to calculate precision and recall metrics.

- **Precision** measures how often it was really a bird in cases where we predicted “bird”.
- **Recall** measures the percent of real birds we were able to identify.

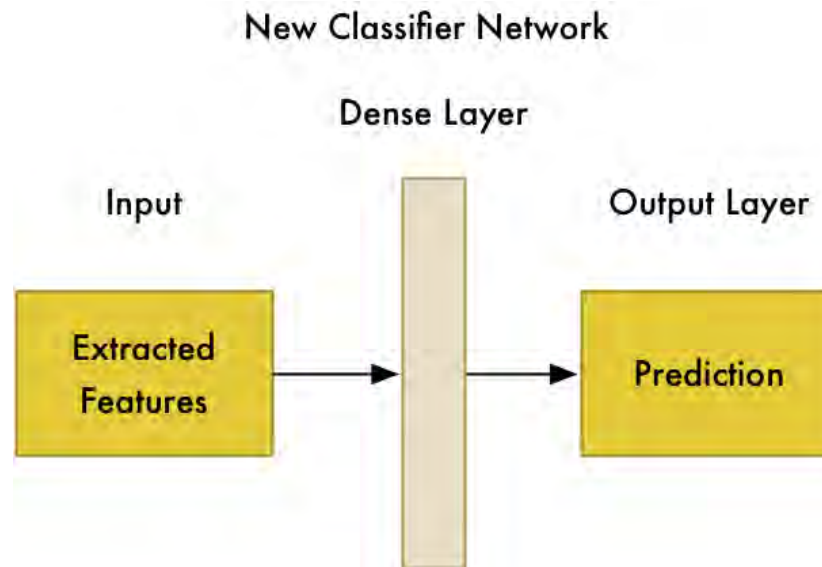
Transfer Learning

- How to Transfer Knowledge Between Networks



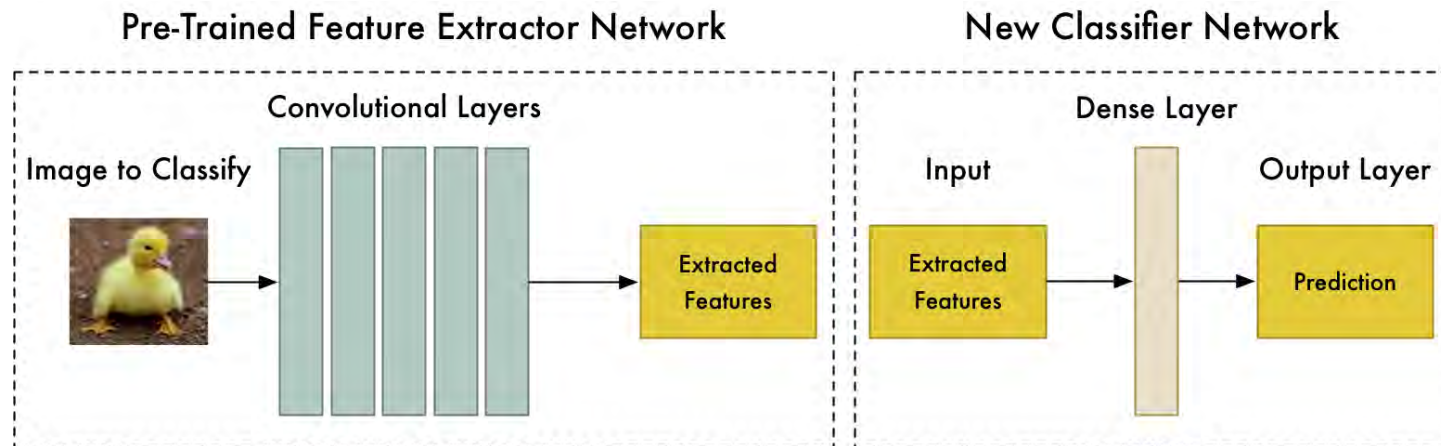
Transfer Learning

- To handle the last step of classifying those extracted features as “bird” or “not bird”, we’ll create a new neural network that just contains the dense layers that do the job of classification.



Transfer learning

- Making Predictions with Transfer Learning



To classify a new image, we have to first pass it through the feature extractor network and save the extracted features. Then we can feed the extracted features into the classifier network to get a final answer.



Where to Get a Pre-Trained Neural Network

- there are lots of pre-trained neural networks that have been shared by researchers. Each year, there is a competition called the ImageNet Large Scale Visual Recognition Challenge or ILSVRC. Researchers are given a dataset of millions of images that need to be classified as one of 1,000 types of common objects, like breeds of dogs and household objects. Models trained on this dataset are widely available.

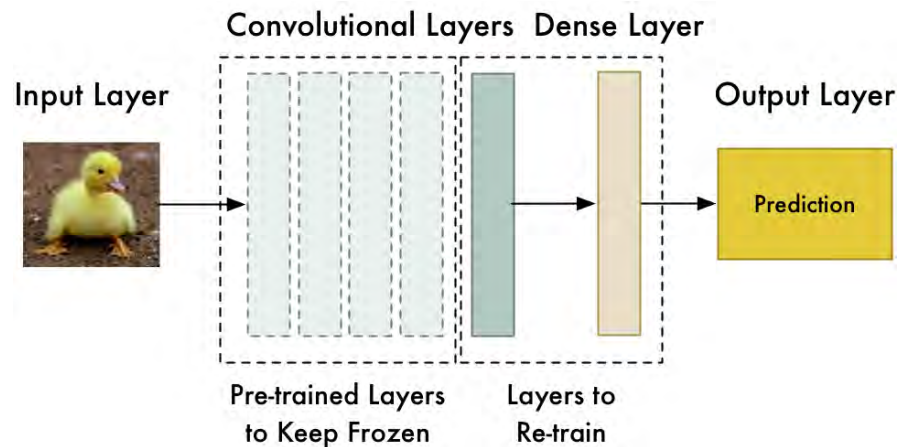


When to Use Transfer Learning

- There are two reasons transfer learning is so great:
 - It can cut the time required to train a new image classifier from weeks to minutes.
 - It can make it possible to train a new image classification system with much less training data.
-

Other Forms of Transfer Learning

- **Retraining Convolutional Layers**



- **Fine-Tuning the Entire Neural Network**

- When you have a lot of training data, you might be able to train a neural network from scratch. But even in those cases it usually makes sense to start the training process by loading a pre-trained neural network's weights as a starting point. The idea is that you are retuning an existing neural network instead of starting from scratch. This usually speeds up the training process and gets you good results much more quickly than starting with a blank slate.



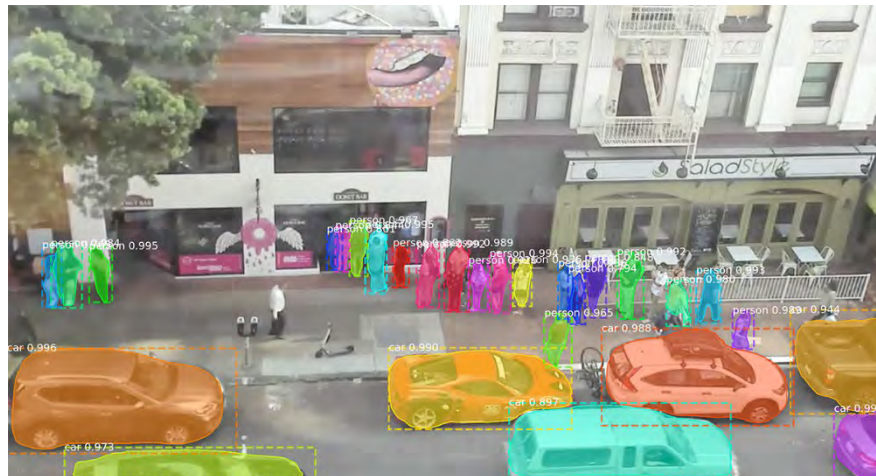
Activity 2.2 Building a bird classifier with Transfer Learning

- Extracting Features with a Pre-Trained CNN
- Training Our New Classifier
- Using the Model to Make Predictions
- Checking Precision and Recall

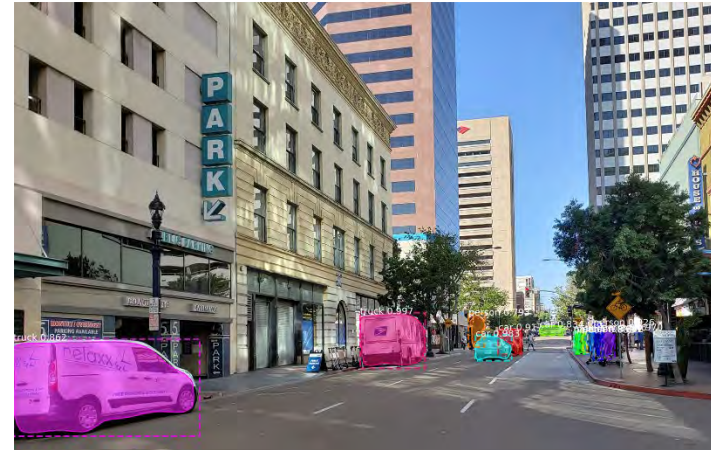
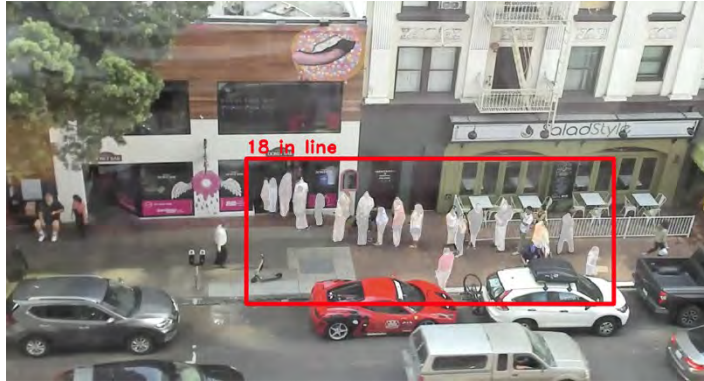


Image Segmentation

- What is image segmentation
 - expand on CNNs to build more complex models that are able to find multiple objects within a picture, label them, and even trace their outlines. This is called image segmentation.
 - not only labelling objects in a picture, but we are locating their position in the picture and finding which exact pixels belong to each object.



The Uses of Image Segmentation



The Uses of Image Segmentation



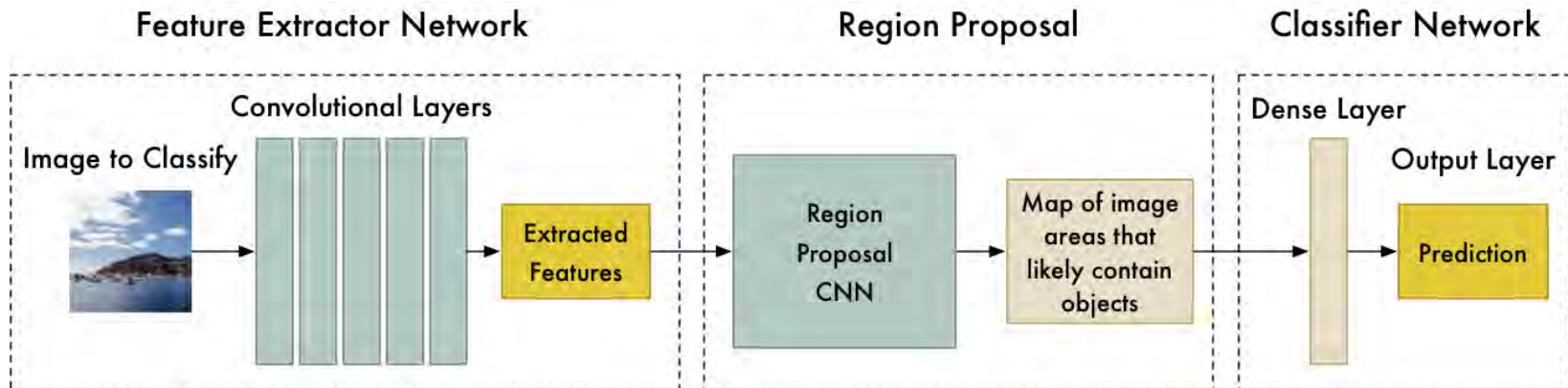
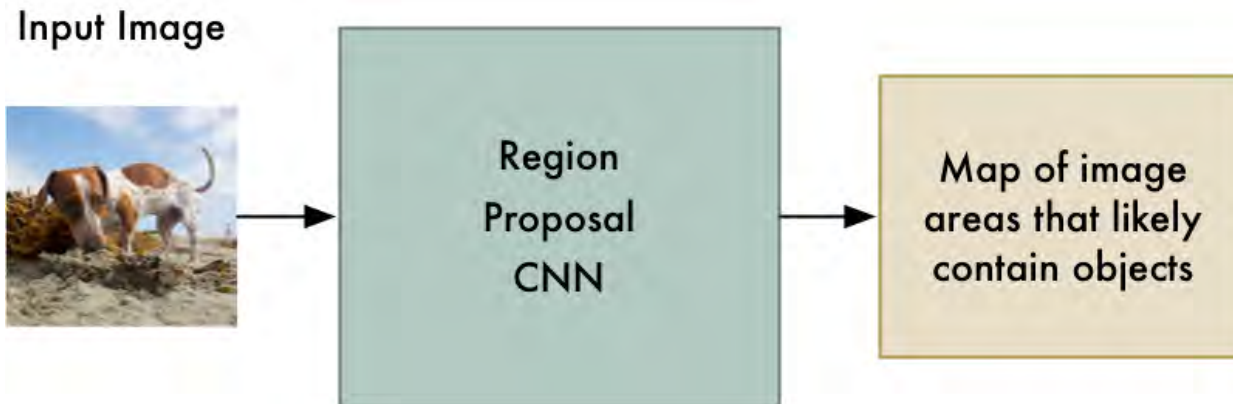
Building an Image Segmentation Model



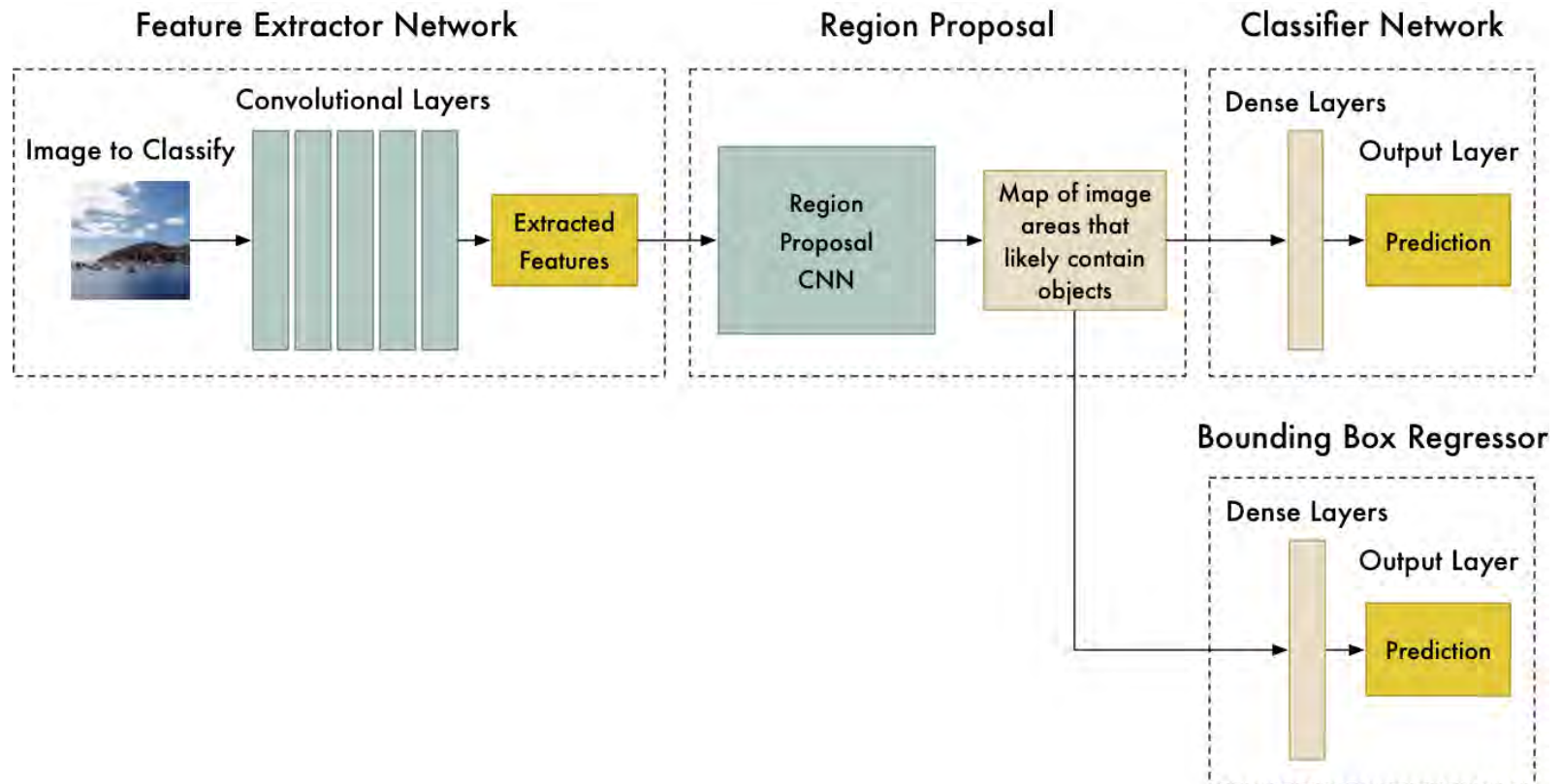
Localization Approach 1: Sliding Window Detector



Localization Approach 2: Region Proposal Network



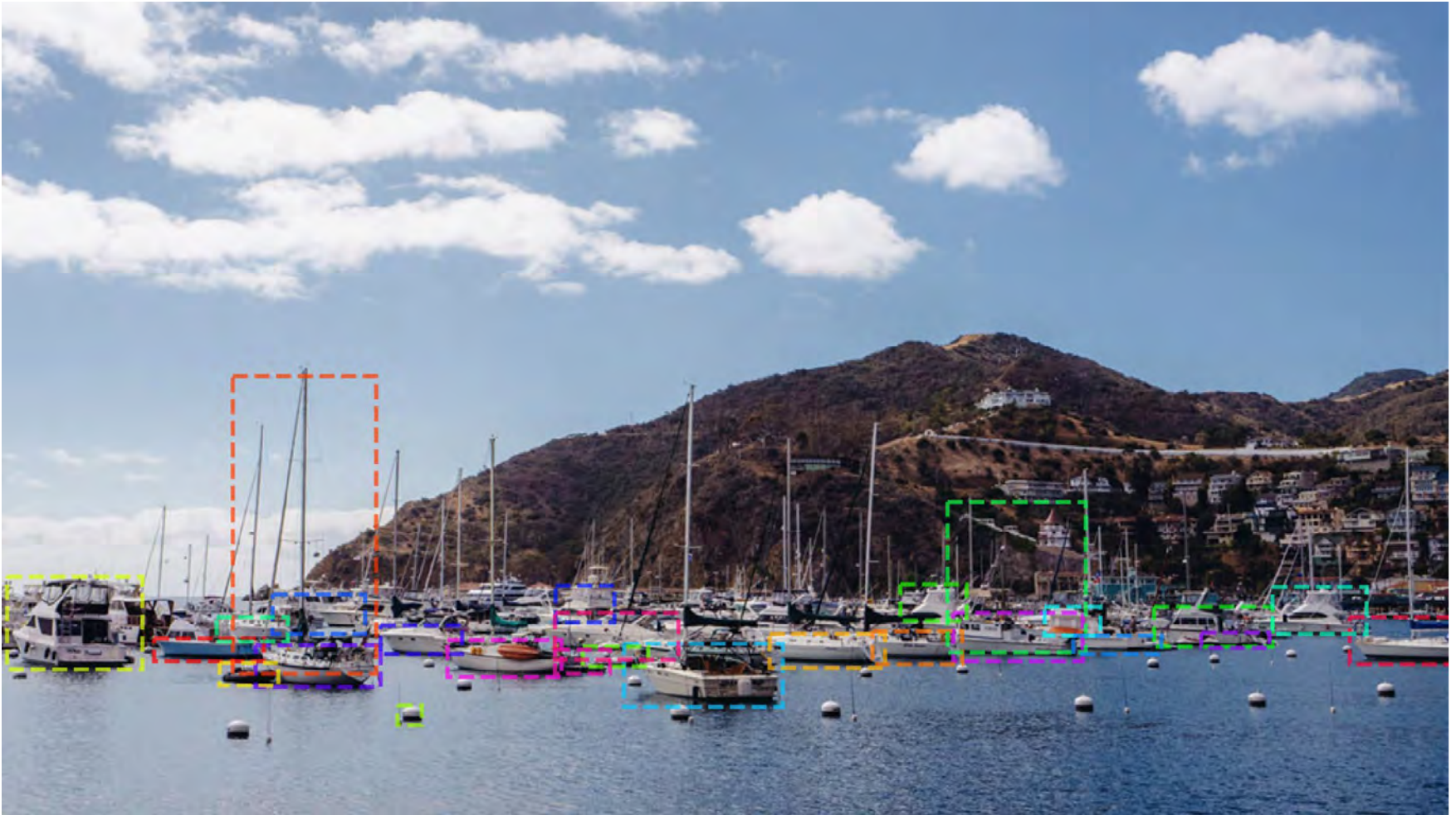
Localization Approach 2: Region Proposal Network



Localization Approach 2: Region Proposal Network



Localization Approach 2: Region Proposal Network

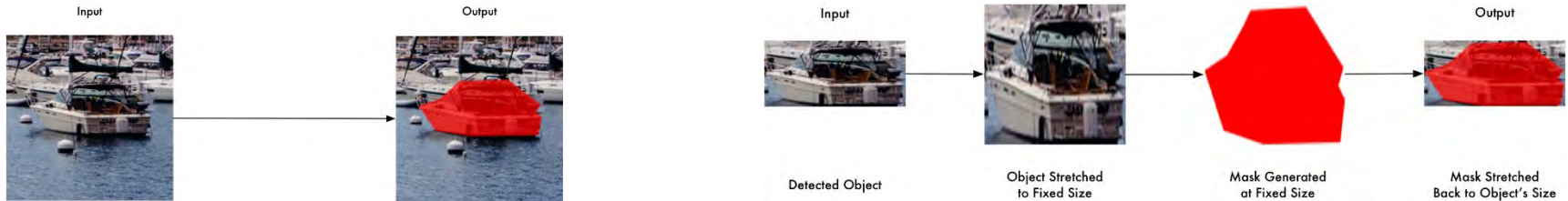


non-maximum suppression algorithm

Classifying the Detected Objects

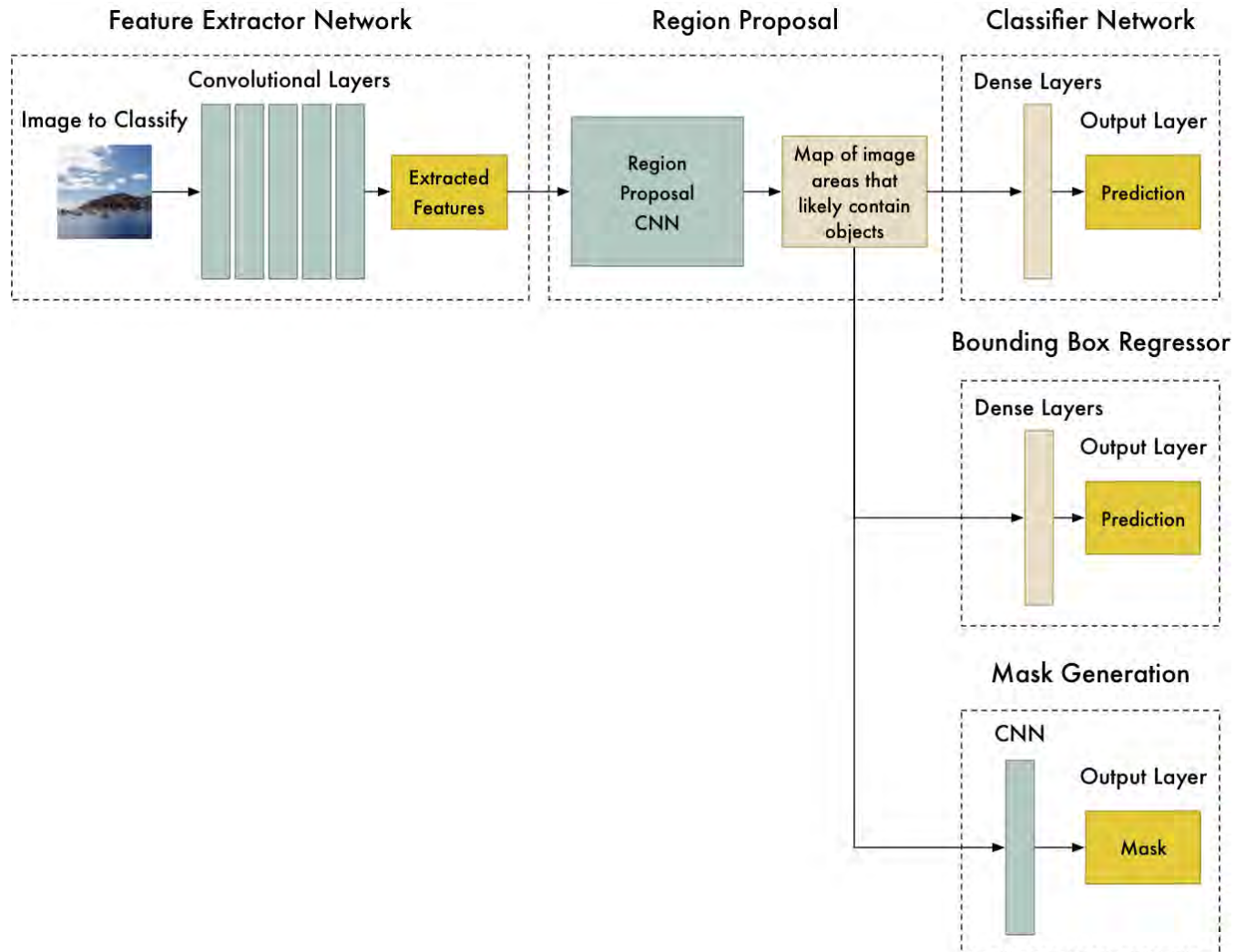


Adding Object Mask Generation to Our Model



Predicting pixel-by-pixel masks is harder than predicting a bounding box because you have to separate the object from the background image. To train a system like this, all your training data needs to be segmented in the same way. It takes a lot more work to manually trace out shapes in training images than to just draw boxes around objects.

The complete model





Activity 2.3 - Using a Pre-trained Mask R-CNN Model

- Use a Mask R-CNN model to build a program that can count the number of people waiting in a line
- Activating the Mask-RCNN Virtual Environment
- Setting up the Model
- Use some image processing techniques in OpenCV to isolate the queue line
- Perform image segmentation and count the number of human in the line

Activity 2.4 – Training a Mask R-CNN segmentation Model

- Warning: This project is especially computationally intensive and it may take a long time to run. You can reduce the number of training epochs (from 30 to 1) to see less accurate results in less time. This project requires at least 8GB of RAM to run.
- Learn how to train Mask R-CNN to detect a new object
- Gathering and Annotating Training Data (DEMO)
- Coding the Training Script
- Running the Training Process
- Using the Trained Model for Image Segmentation





Quiz

<http://bit.ly/2Y8MTi2>





Genetic Programming

- **Introduction to Genetic Algorithm (Darwinian Evolution)**
- **The principle of variation:** The traits (attributes) of individual specimens belonging to a population may vary. As a result, the specimens differ from each other to some degree; for example, in their behaviour or appearance.
- **The principle of inheritance:** Some traits are consistently passed on from specimens to their offspring. As a result, offspring resemble their parents more than they resemble unrelated specimens.
- **The principle of selection:** Populations typically struggle for resources within their given environment. The specimens possessing traits that are better adapted to the environment will be more successful at surviving, and will also contribute more offspring to the next generation.
-



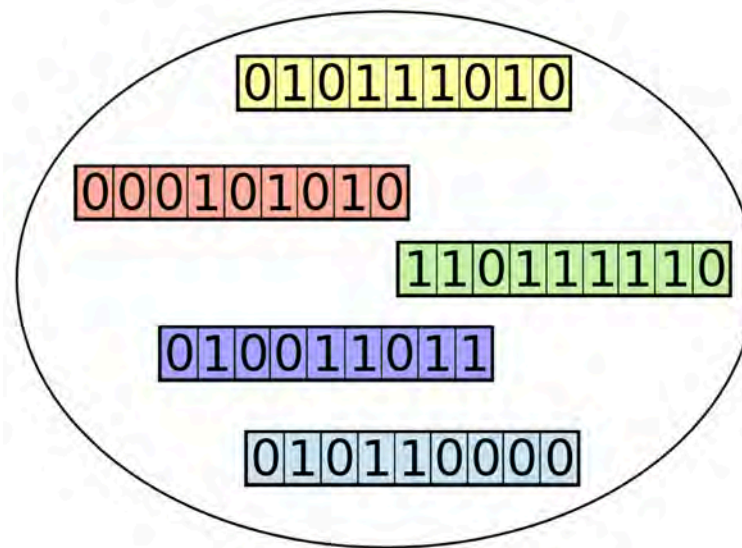
The genetic algorithm analogy

- Genotype
 - In nature, breeding, reproduction, and mutation are facilitated via the genotype – a collection of genes that are grouped into chromosomes. If two specimens breed to create offspring, each chromosome of the offspring will carry a mix of genes from both parents.
 - Mimicking this concept, in the case of genetic algorithms, each individual is represented by a chromosome representing a collection of genes. For example, a chromosome can be expressed as a binary string, where each bit represents a single gene:

0	1	0	1	1	1	0	1	0
---	---	---	---	---	---	---	---	---

The genetic algorithm analogy

- Population
 - At any point in time, genetic algorithms maintain a population of individuals – a collection of candidate solutions for the problem at hand. Since each individual is represented by some chromosome, this population of individuals can be seen as a collection of such chromosomes





The genetic algorithm analogy

- **Fitness function**

- At each iteration of the algorithm, the individuals are evaluated using a fitness function (also called the target function). This is the function we seek to optimize or the problem we attempt to solve.
- Individuals who achieve a better fitness score represent better solutions and are more likely to be chosen to reproduce and be represented in the next generation. Over time, the quality of the solutions improves, the fitness values increase, and the process can stop once a solution is found with a satisfactory fitness value.

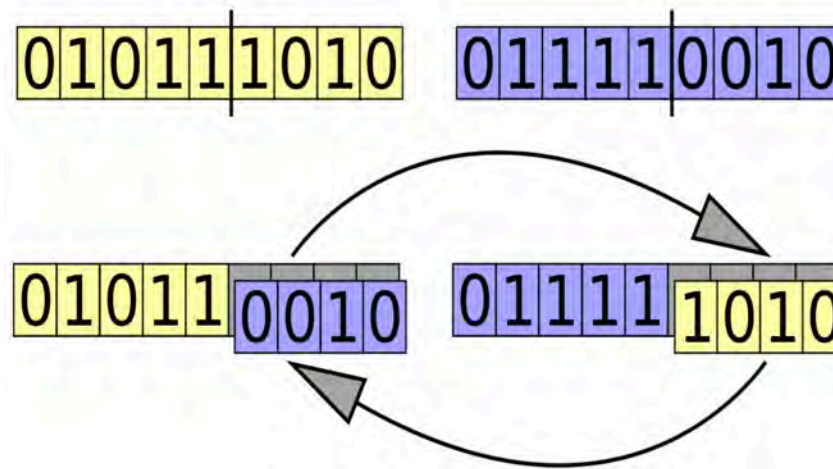
- **Selection**

- After calculating the fitness of every individual in the population, a selection process is used to determine which of the individuals in the population will get to reproduce and create the offspring that will form the next generation. This selection process is based on the fitness score of the individuals. Those with higher score values are more likely to be chosen and pass their genetic material to the next generation.
- Individuals with low fitness values can still be chosen, but with lower probability. This way, their genetic material is not completely excluded.

The genetic algorithm analogy

- **Crossover**

- To create a pair of new individuals, two parents are usually chosen from the current generation, and parts of their chromosomes are interchanged (crossed over) to create two new chromosomes representing the offspring. This operation is called crossover, or recombination:

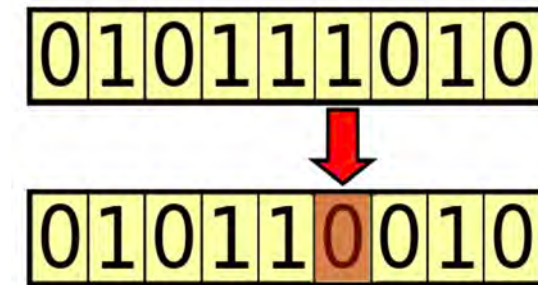


Source: <https://commons.wikimedia.org/wiki/File:Computational.science.Genetic.algorithm.Crossover.One.Point.svg> Image by Yearofthedragon. Licensed under Creative Commons CC BY-SA 3.0: <https://creativecommons.org/licenses/by-sa/3.0/deed.en> Mutation

The genetic algorithm analogy

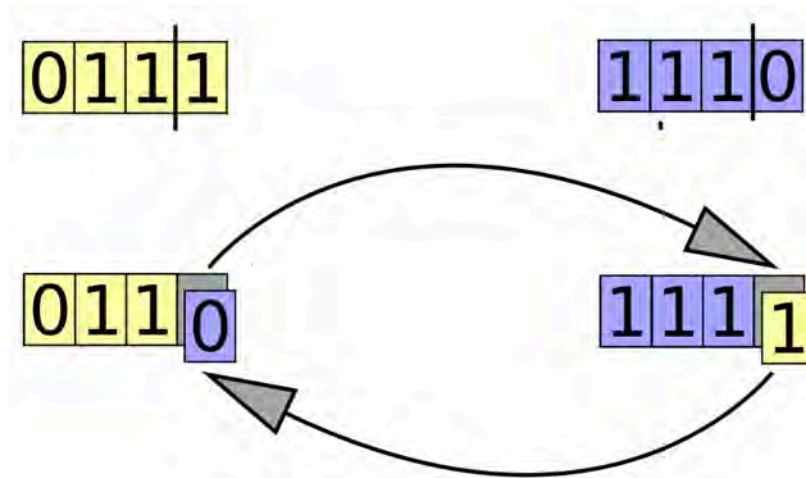
- **Mutation**

- The purpose is to periodically and randomly refresh the population, introduce new patterns into the chromosomes, and encourage search in uncharted areas of the solution space.
- may manifest itself as a random change in a gene.
- implemented as random changes to one or more of the chromosome values; for example, flipping a bit in a binary string:



The theory behind genetic algorithms

- Key idea : each generation is better than the previous one and contains more individuals that are closer to the optimal solution.



Differences from traditional algorithms

- The key characteristics of genetic algorithms distinguishing them from traditional algorithms are:
 - Maintaining a population of solutions
 - Using a genetic representation of the solutions
 - Utilizing the outcome of a fitness function
 - Exhibiting a probabilistic behaviour



Advantages of genetic algorithm

- Global optimization capability
- Handling problems with a complex mathematical representation
- Handling problems that lack mathematical representation
- Resilience to noise
- Support for parallelism and distributed processing
- Suitability for continuous learning



Limitation of genetic algorithms

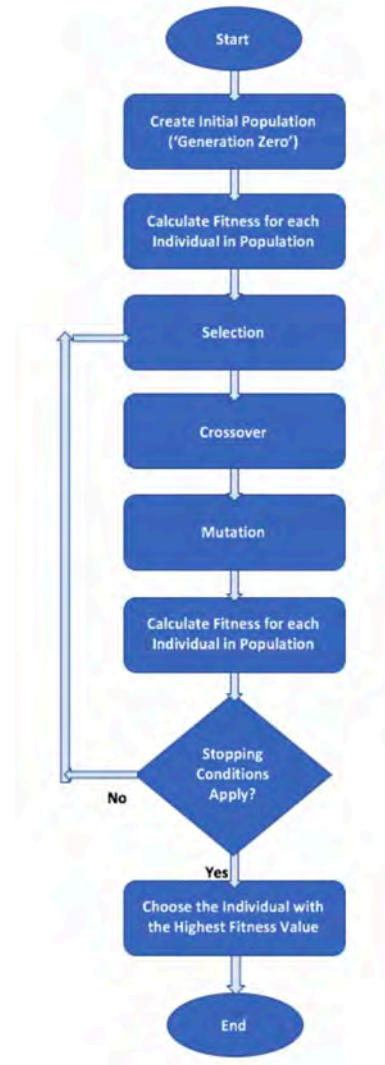
- The need for special definitions
- The need for hyperparameter tuning
- Computationally-intensive operations
- The risk of premature convergence
- No guaranteed solution



Use cases of genetic algorithms

- Problems with complex mathematical representation
- Problems with no mathematical representation
- Problems involving a noisy environment
- Problems involving an environment that changes over time

Understanding the key components of Genetic Algorithms



Solving Problems with Genetic Algorithms

- Using the DEAP Framework

DEAP is a novel evolutionary computation framework for rapid prototyping and testing of ideas. It seeks to make algorithms explicit and data structures transparent. It works in perfect harmony with parallelisation mechanism such as multiprocessing and [SCOOP](https://deap.readthedocs.io/en/master/). The following documentation presents the key concepts and many features to build your own evolutions.



DISTRIBUTED
EVOLUTIONARY
ALGORITHMS IN
PYTHON

<https://deap.readthedocs.io/en/master/>



Activity 2.5 Knapsack problem

- The knapsack problem consists of the following components:
 - A set of items, each of them associated with a certain value and a certain weight
 - A bag/sack/container (the knapsack) of a certain weight capacity
 - Our goal is to come up with a group of selected items that will provide the maximum total value, without exceeding the total weight capacity of the bag.

The Rosetta Code website (rosettacode.org) provides a collection of programming tasks, each with contributed solutions in numerous languages



Activity 2.5 Knapsack problem

The weight capacity of the tourist's bag in this problem is 400, and the list of items, along with their associated values and weights, is as follows:

Item	Weight	Value
Map	9	150
Compass	13	35
Water	153	200
Sandwich	50	160
Glucose	15	60
Tin	68	45
Banana	27	60
Apple	27	60
Cheese	23	30
Beer	52	10
Suntan cream	11	70
Camera	32	30
t-shirt	24	15
Trousers	48	10
Umbrella	73	40
Waterproof trousers	42	70
Waterproof overclothes	43	75
Note-case	22	80
Sunglasses	7	20
Towel	18	12
Socks	4	50
Book	30	10



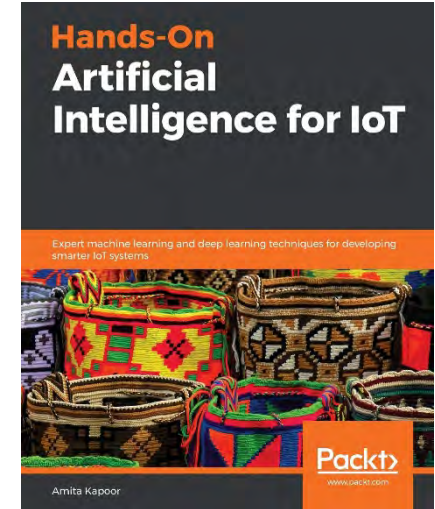
Quiz

<http://bit.ly/2Y8MTi2>





References





Thank you

