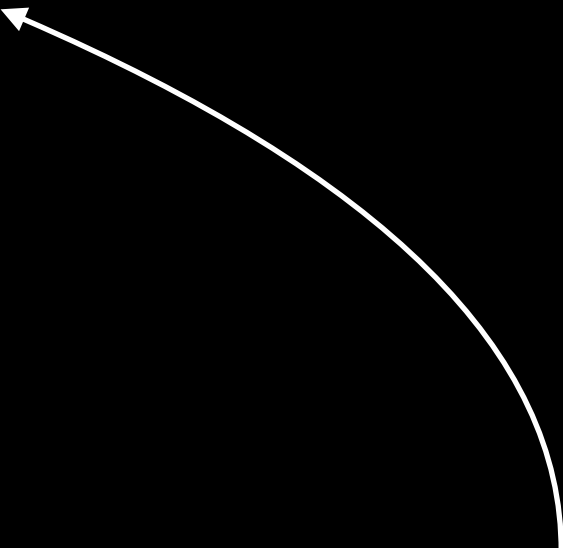




Deep Learning with TensorFlow

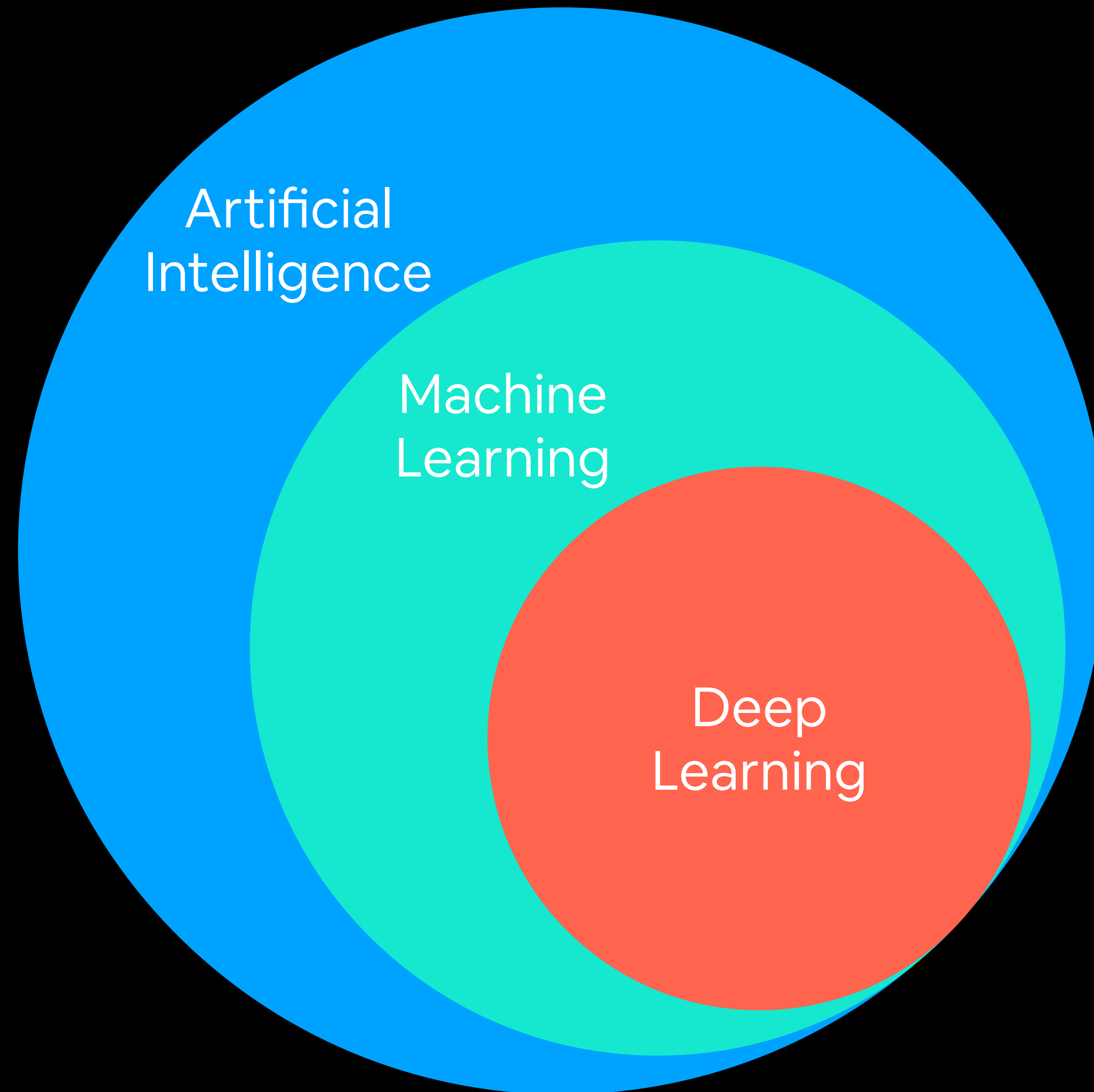
“What is deep learning?”

Machine learning is turning things (data)
into numbers and **finding patterns** in those
numbers.



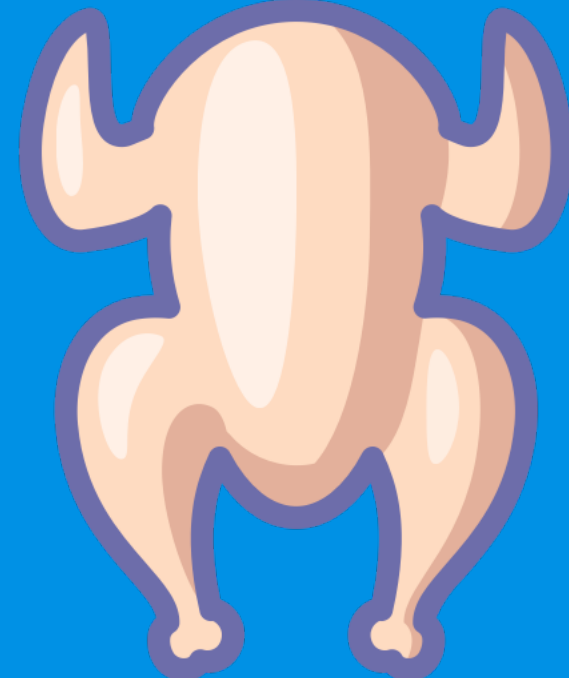
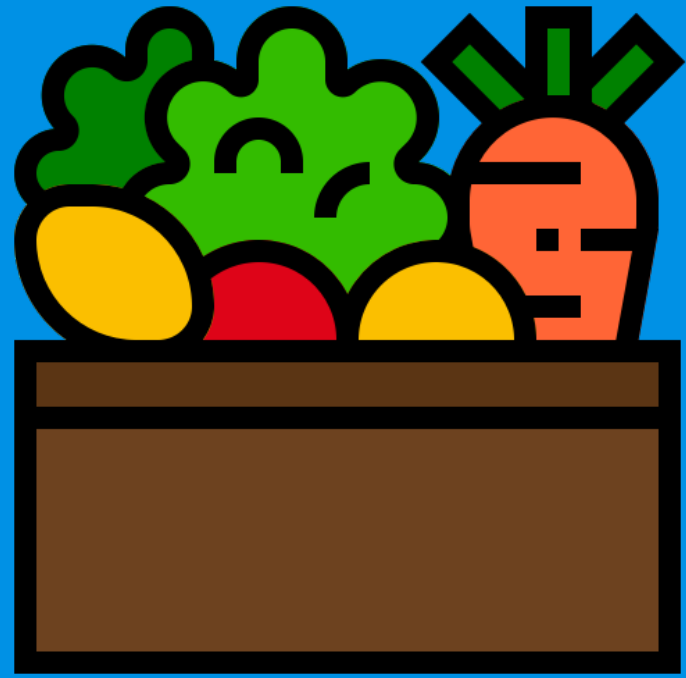
The computer does this part.
How?
Code & math.
We're going to be writing the code.

Machine Learning vs. Deep Learning



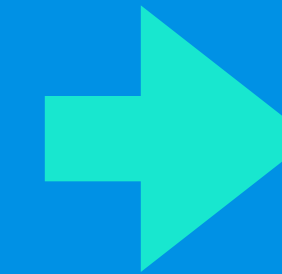
Traditional programming

Inputs

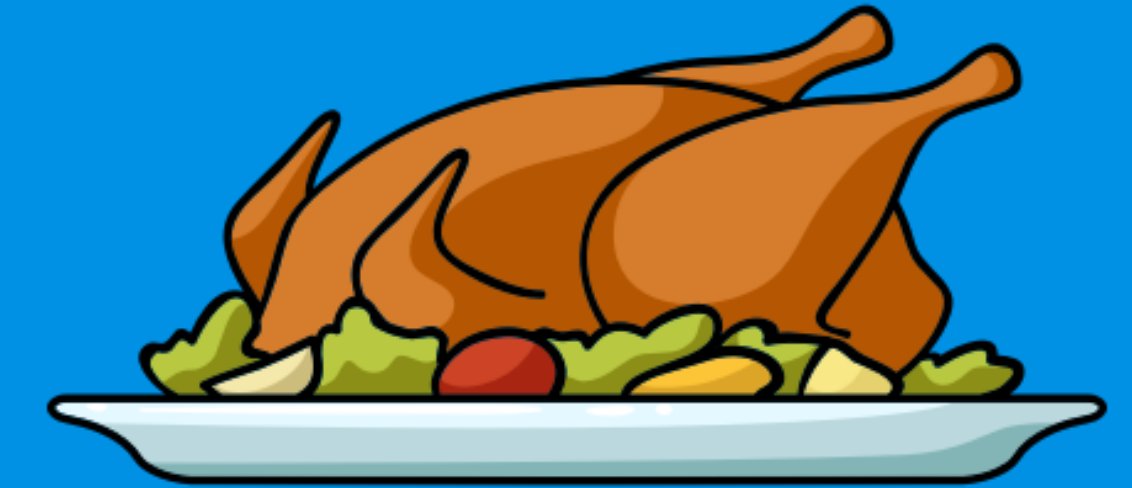


Rules

1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



Output

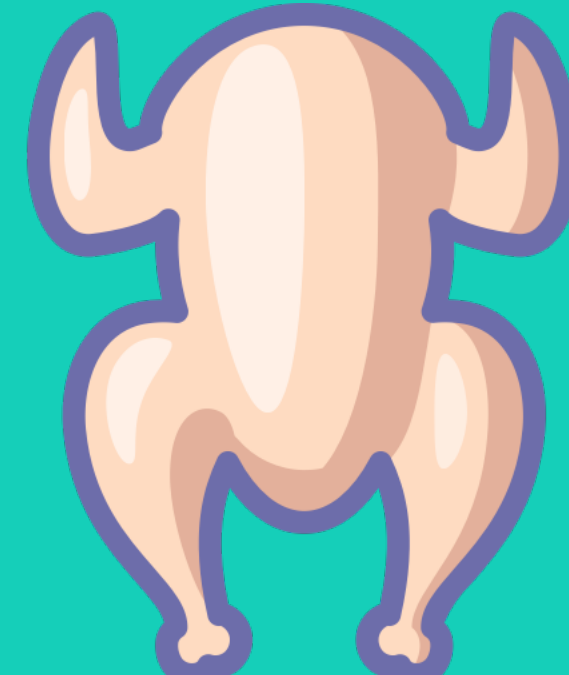


Starts with

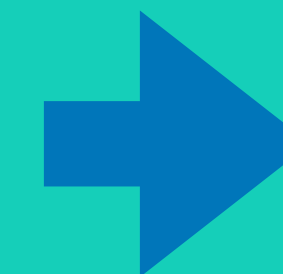
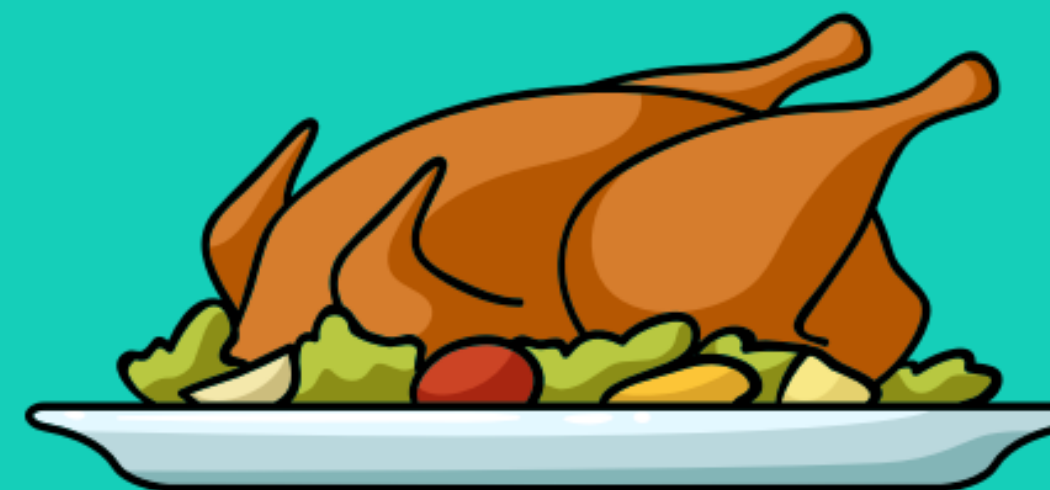
Makes

Machine learning algorithm

Inputs



Output



Rules

1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables

Starts with

Figures out

**“Why use machine learning (or
deep learning)?”**

Good reason: ~~Why not?~~

Better reason: For a complex problem, can you think of all the rules?
(probably not)

(maybe not very simple...)

“If you can build a **simple rule-based** system that doesn't require machine learning, do that.”

— A wise software engineer... (actually rule 1 of Google's Machine Learning Handbook)

What deep learning is good for

- **Problems with long lists of rules**—when the traditional approach fails, machine learning/deep learning may help.
- **Continually changing environments**—deep learning can adapt (‘learn’) to new scenarios.
- **Discovering insights within large collections of data**—can you imagine trying to hand-craft rules for what 101 different kinds of food look like?

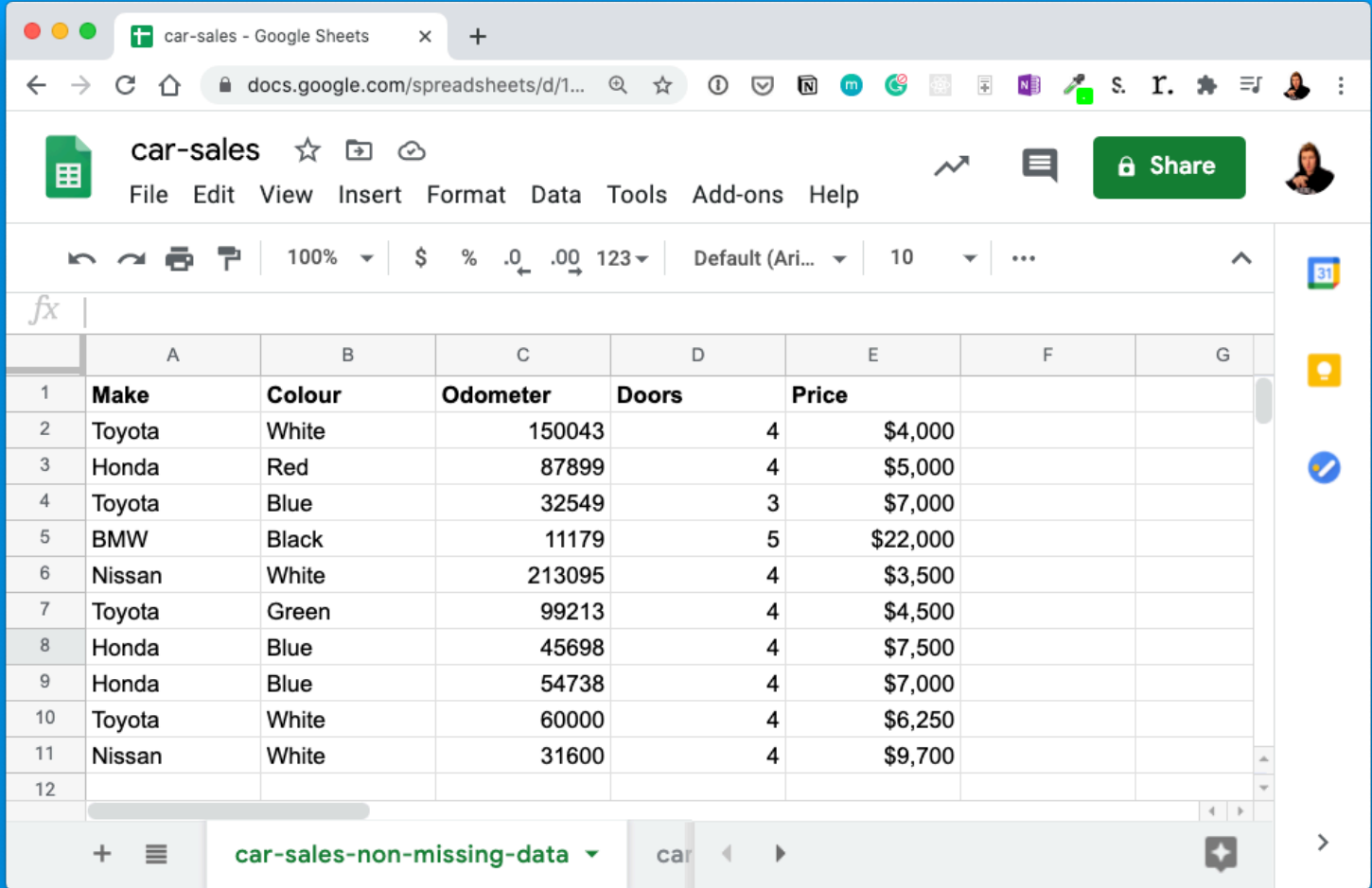
What deep learning is ^(typically) not good for

- **When you need explainability**—the patterns learned by a deep learning model are typically uninterpretable by a human.
- **When the traditional approach is a better option** — if you can accomplish what you need with a simple rule-based system.
- **When errors are unacceptable** — since the outputs of deep learning model aren't always predictable.
- **When you don't have much data** — deep learning models usually require a fairly large amount of data to produce great results.

(though we'll see how to get great results without huge amounts of data)

Machine Learning vs. Deep Learning

Machine Learning

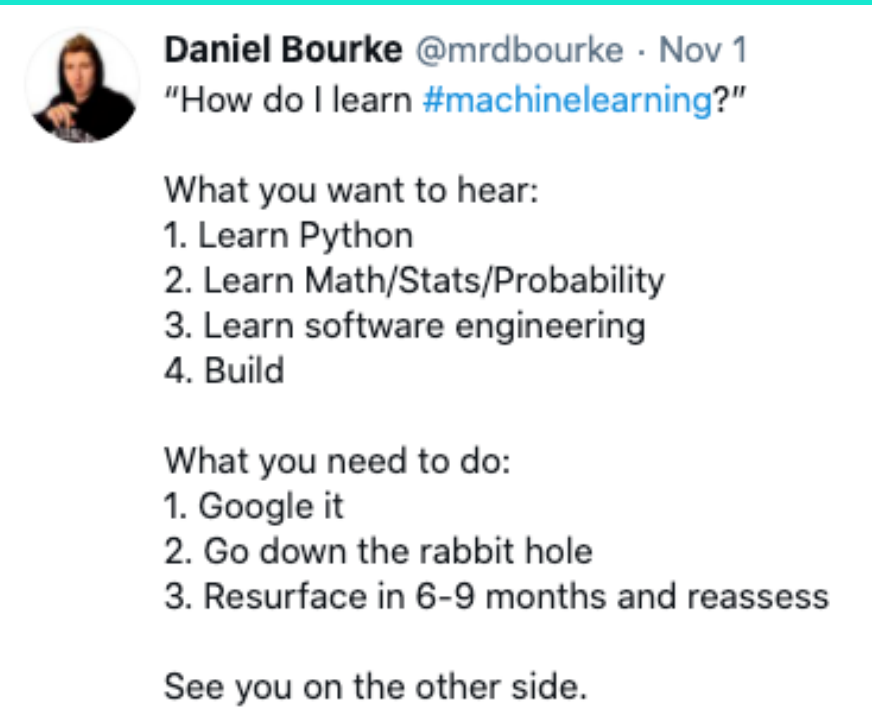


	A	B	C	D	E	F	G
1	Make	Colour	Odometer	Doors	Price		
2	Toyota	White	150043	4	\$4,000		
3	Honda	Red	87899	4	\$5,000		
4	Toyota	Blue	32549	3	\$7,000		
5	BMW	Black	11179	5	\$22,000		
6	Nissan	White	213095	4	\$3,500		
7	Toyota	Green	99213	4	\$4,500		
8	Honda	Blue	45698	4	\$7,500		
9	Honda	Blue	54738	4	\$7,000		
10	Toyota	White	60000	4	\$6,250		
11	Nissan	White	31600	4	\$9,700		
12							



Structured data

Deep Learning



Daniel Bourke @mrdbourke · Nov 1
"How do I learn #machinelearning?"

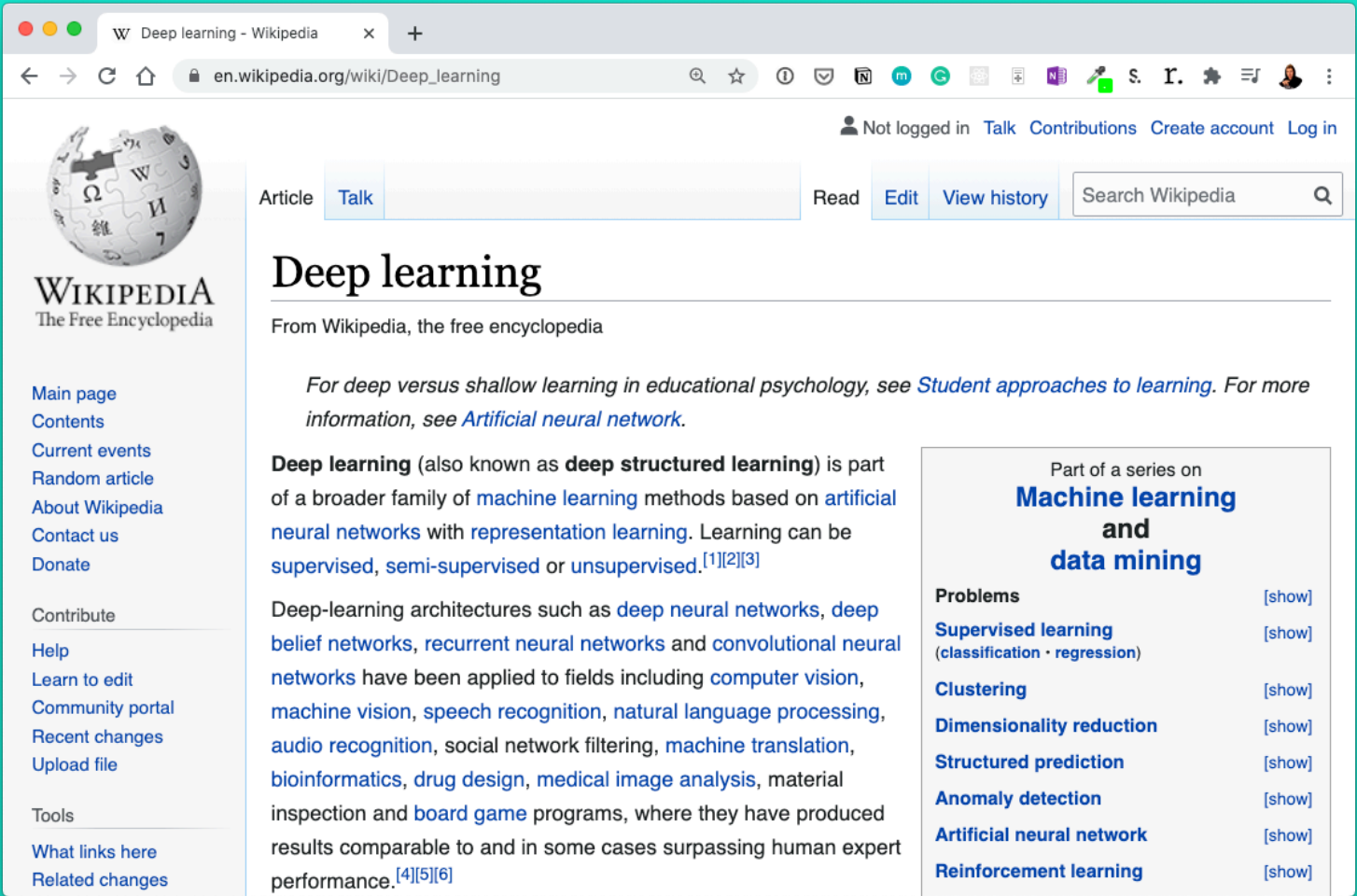
What you want to hear:

1. Learn Python
2. Learn Math/Stats/Probability
3. Learn software engineering
4. Build

What you need to do:

1. Google it
2. Go down the rabbit hole
3. Resurface in 6-9 months and reassess

See you on the other side.



Wikipedia - Deep learning

Deep learning

From Wikipedia, the free encyclopedia

For deep versus shallow learning in educational psychology, see Student approaches to learning. For more information, see Artificial neural network.

Deep learning (also known as **deep structured learning**) is part of a broader family of **machine learning** methods based on **artificial neural networks** with **representation learning**. Learning can be supervised, semi-supervised or unsupervised.^{[1][2][3]}

Deep-learning architectures such as **deep neural networks**, **deep belief networks**, **recurrent neural networks** and **convolutional neural networks** have been applied to fields including **computer vision**, **machine vision**, **speech recognition**, **natural language processing**, **audio recognition**, **social network filtering**, **machine translation**, **bioinformatics**, **drug design**, **medical image analysis**, **material inspection** and **board game programs**, where they have produced results comparable to and in some cases surpassing human expert performance.^{[4][5][6]}

Part of a series on
Machine learning and data mining

- Problems** [show]
- Supervised learning** (classification · regression) [show]
- Clustering** [show]
- Dimensionality reduction** [show]
- Structured prediction** [show]
- Anomaly detection** [show]
- Artificial neural network** [show]
- Reinforcement learning** [show]



Unstructured data

Machine Learning vs. Deep Learning

(common algorithms)

- Random forest
- Naive bayes
- Nearest neighbour
- Support vector machine
- ...many more

(since the advent of deep learning these are often referred to as "shallow algorithms")

- Neural networks
- Fully connected neural network
- Convolutional neural network
- Recurrent neural network
- Transformer
- ...many more

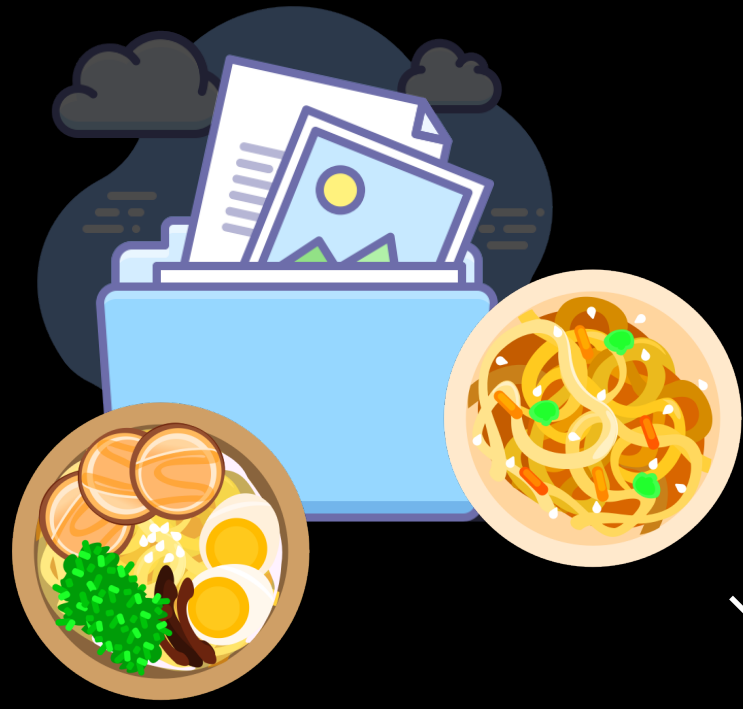
What we're focused on building
(with TensorFlow)

(depending how you represent your problem, many algorithms can be used for both)

Structured data ← **Unstructured data**

“What are neural networks?”

Neural Networks



(before data gets used with a neural network, it needs to be turned into numbers)

Daniel Bourke @mrdbourke · Nov 1
"How do I learn #machinelearning?"

What you want to hear:

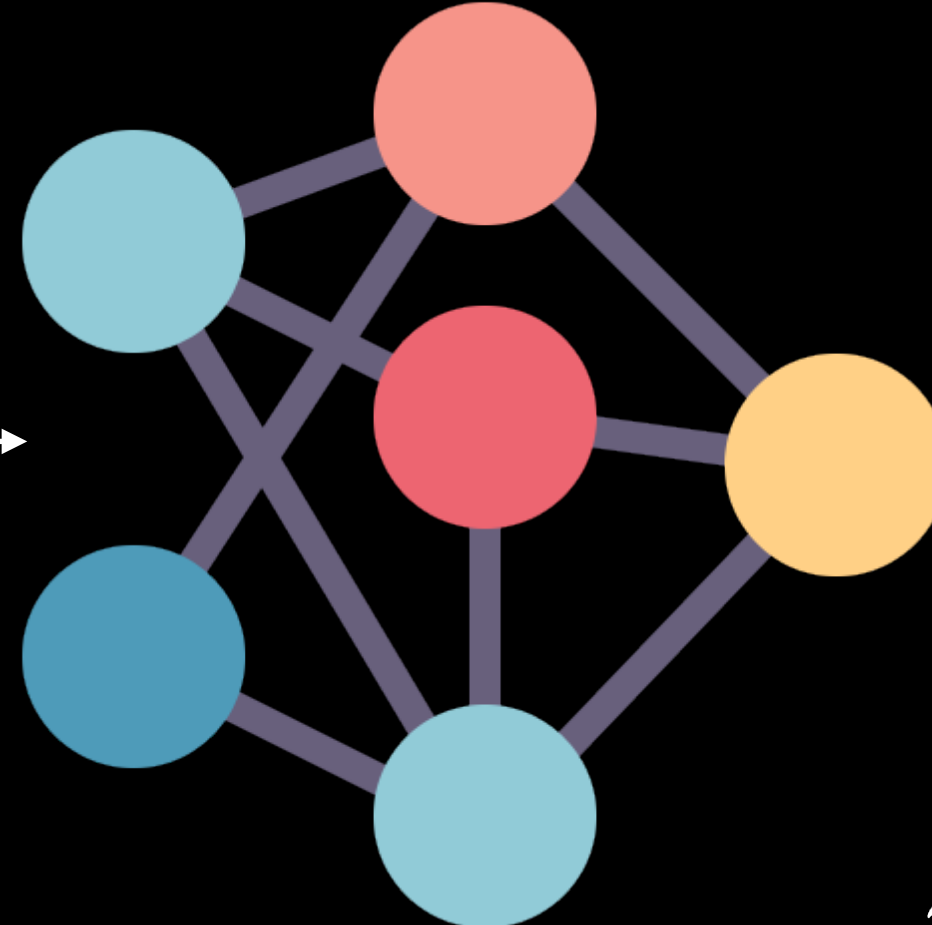
1. Learn Python
2. Learn Math/Stats/Probability
3. Learn software engineering
4. Build

What you need to do:

1. Google it
2. Go down the rabbit hole
3. Resurface in 6-9 months and reassess

See you on the other side.

$[[116, 78, 15],$
 $[117, 43, 96],$
 $[125, 87, 23],$
 $\dots,$



(choose the appropriate neural network for your problem)

$[[0.983, 0.004, 0.013],$
 $[0.110, 0.889, 0.001],$
 $[0.023, 0.027, 0.985],$
 $\dots,$

(a human can understand these)

Ramen,
Spaghetti

Not a diaster

"Hey Siri, what's
the weather
today?"



Inputs

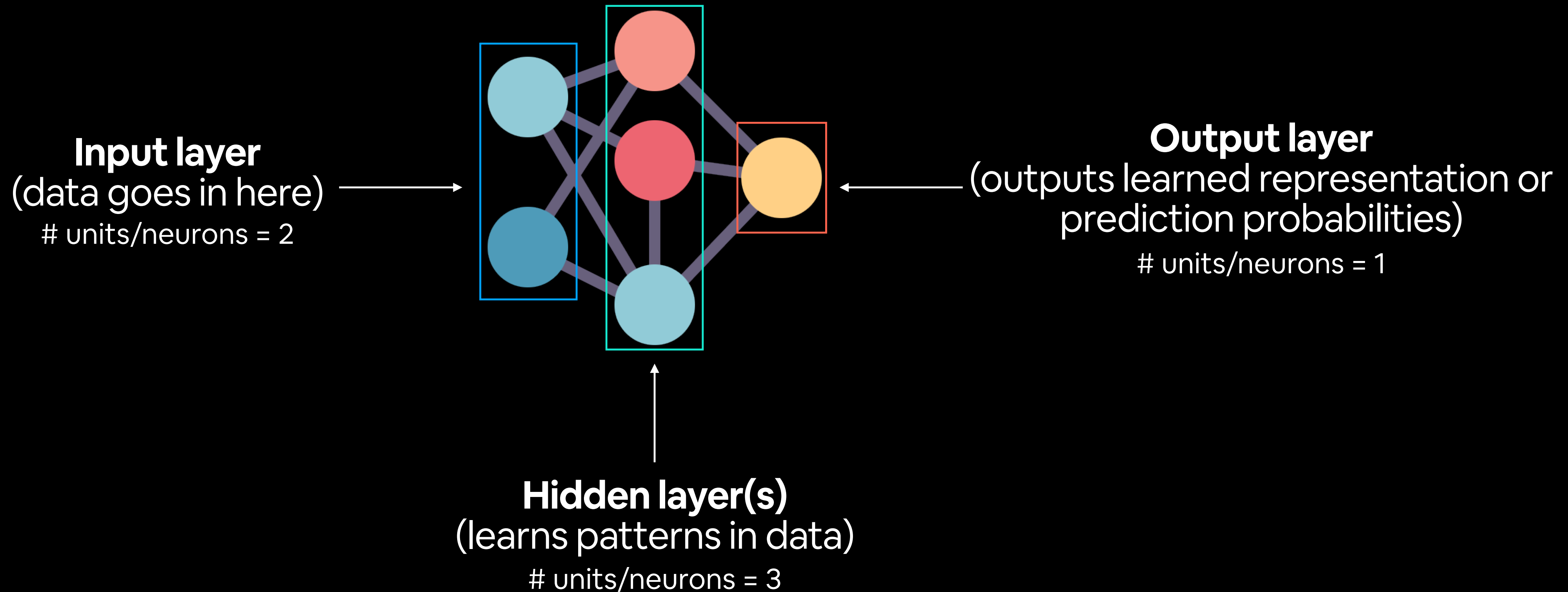
Numerical
encoding

Learns
representation
(patterns/features/weights)

Representation
outputs

Outputs

Anatomy of Neural Networks



Note: “patterns” is an arbitrary term, you’ll often hear “embedding”, “weights”, “feature representation”, “feature vectors” all referring to similar things.

Types of Learning



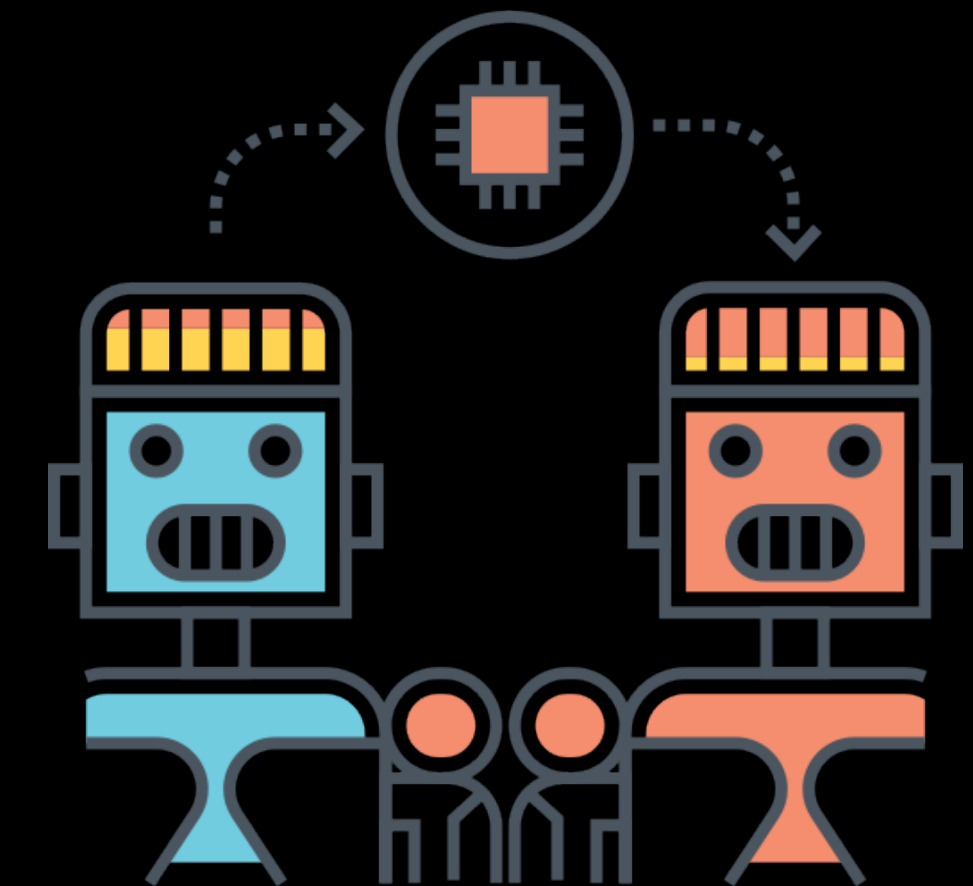
Supervised
Learning



Semi-supervised
Learning



Unsupervised
Learning

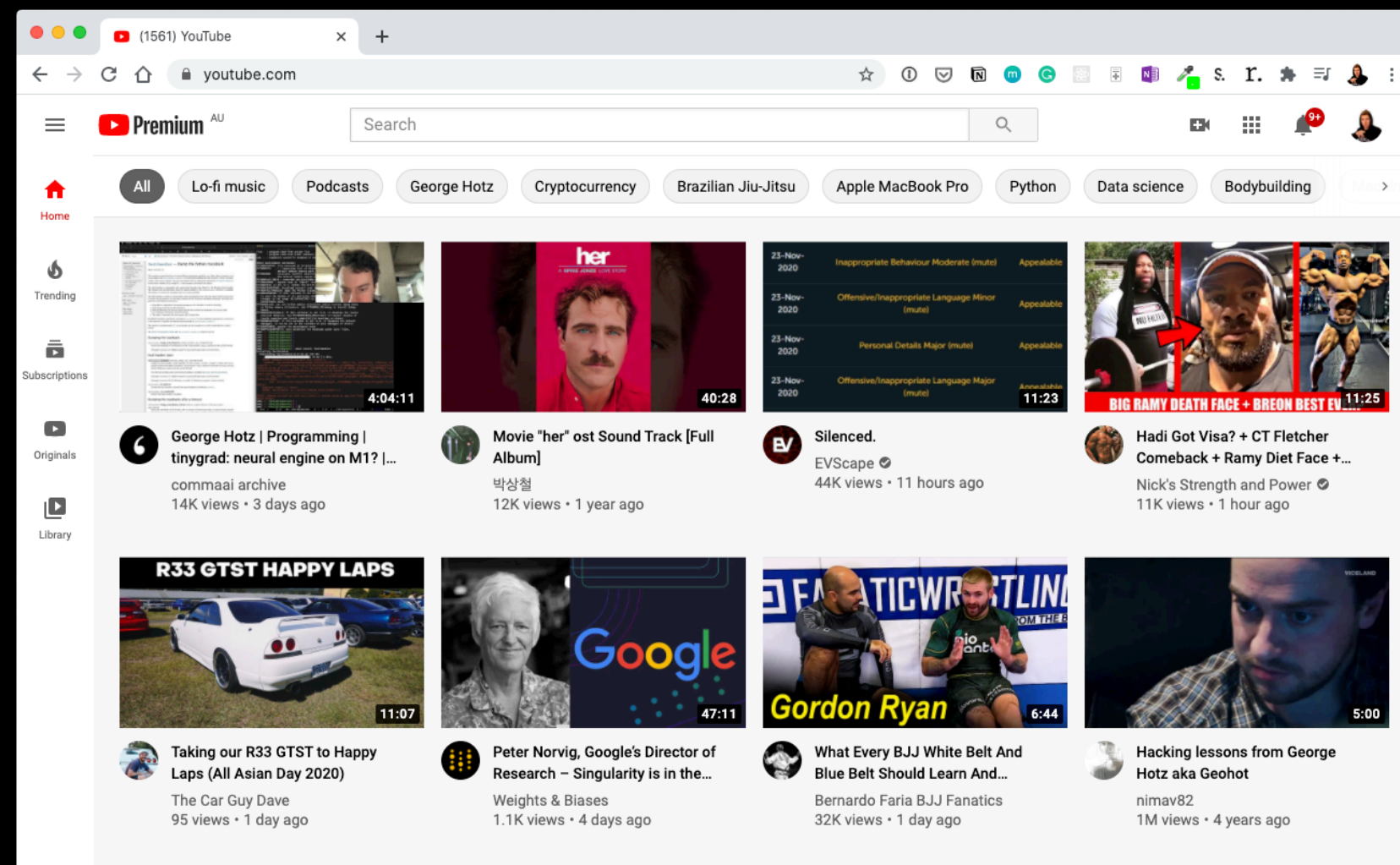


Transfer
Learning

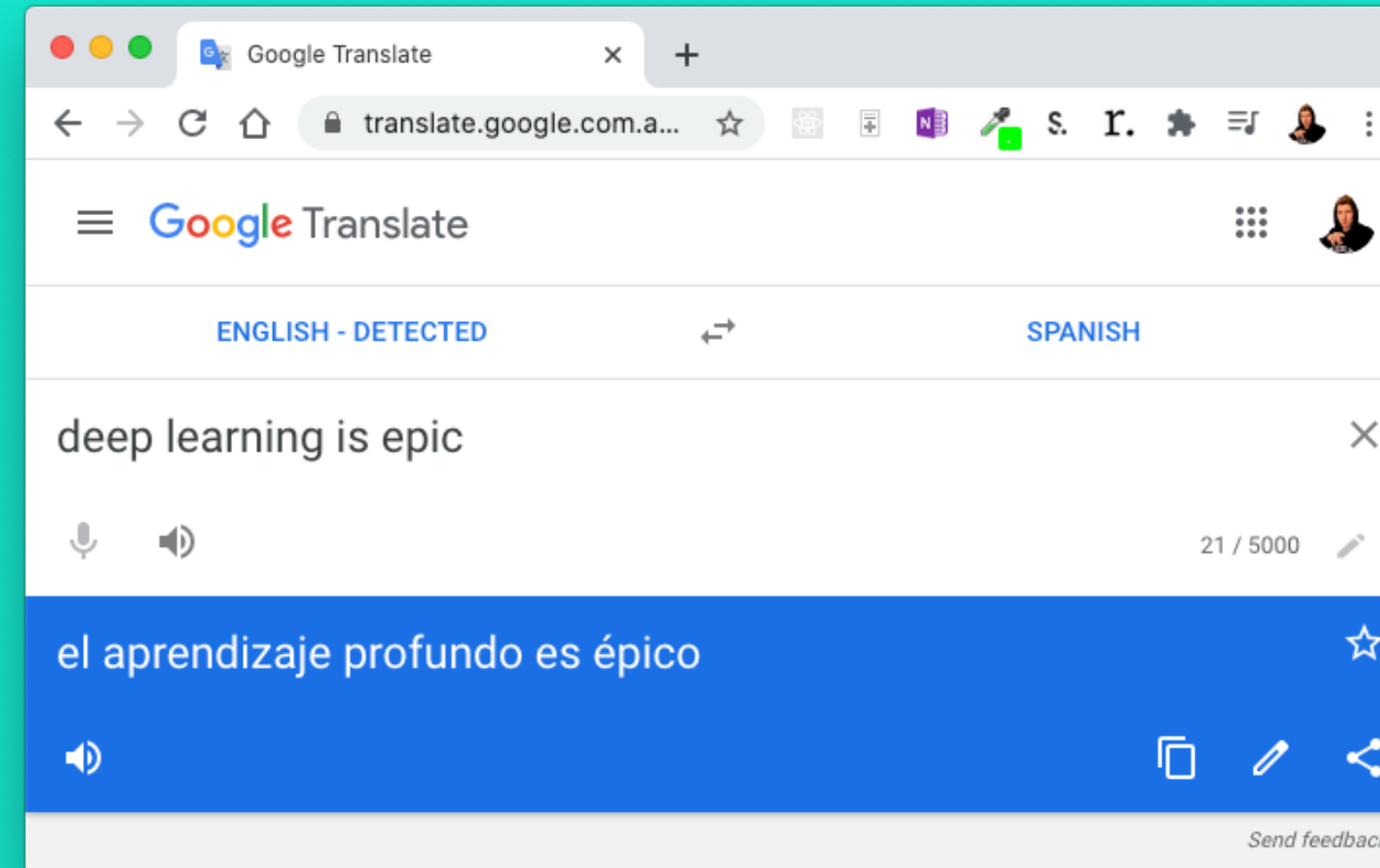
We'll be writing code to do these

**“What is deep learning actually
used for?”**

(some) Deep Learning Use Cases



Recommendation



Translation



“Hey Siri, who’s the biggest big dog of them all?”

Speech recognition



Computer Vision

To: daniel@mrdbourke.com
Hey Daniel,

This deep learning course is incredible!
I can’t wait to use what I’ve learned!

Not spam

To: daniel@mrdbourke.com
Hay daniel...

C0ngratu1ations! U win \$1139239230

Spam

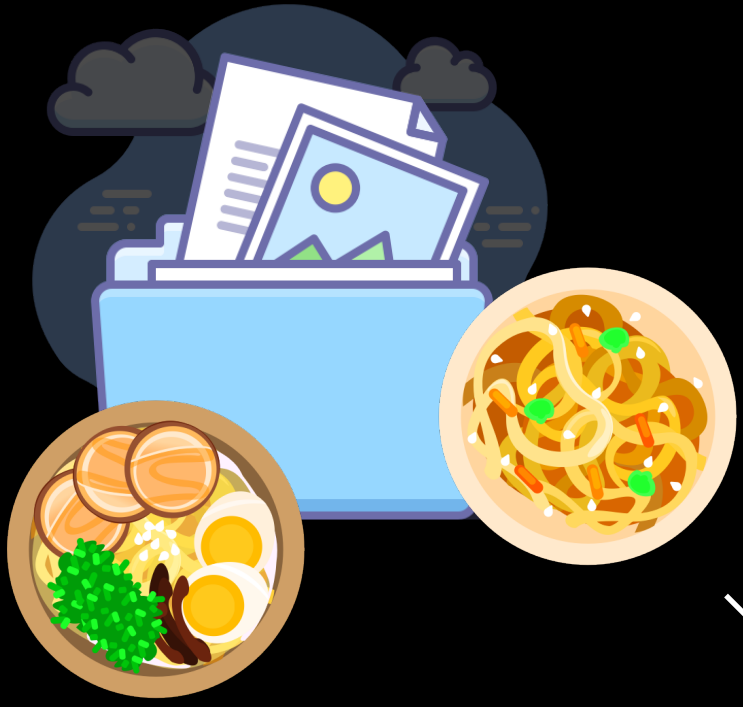
Natural Language Processing (NLP)

Sequence to sequence
(seq2seq)

Classification/regression

“What is a tensor?”

Neural Networks



Daniel Bourke @mrdbourke · Nov 1
"How do I learn #machinelearning?"

What you want to hear:

- 1. Learn Python
- 2. Learn Math/Stats/Probability
- 3. Learn software engineering
- 4. Build

What you need to do:

- 1. Google it
- 2. Go down the rabbit hole
- 3. Resurface in 6-9 months and reassess

See you on the other side.



(before data gets used with an algorithm, it needs to be turned into numbers)

```
[[116, 78, 15],  
[117, 43, 96],  
[125, 87, 23],  
... ,
```



(choose the appropriate neural network for your problem)

These are tensors!

```
[[0.983, 0.004, 0.013],  
[0.110, 0.889, 0.001],  
[0.023, 0.027, 0.985],  
... ,
```

(a human can understand these)

Ramen,
Spaghetti

Not a diaster

"Hey Siri, what's
the weather
today?"

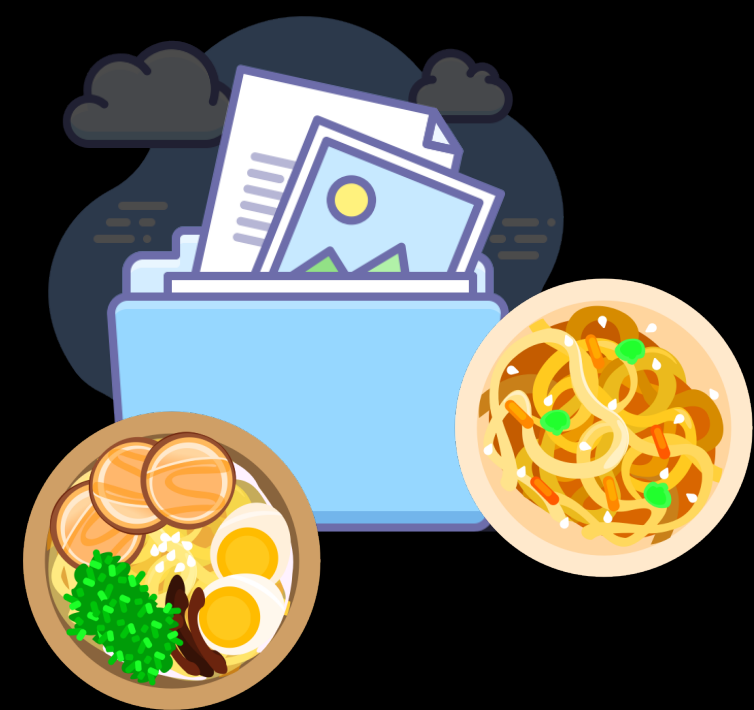
Inputs

Numerical
encoding

Learns
representation
(patterns/features/weights)

Representation
outputs

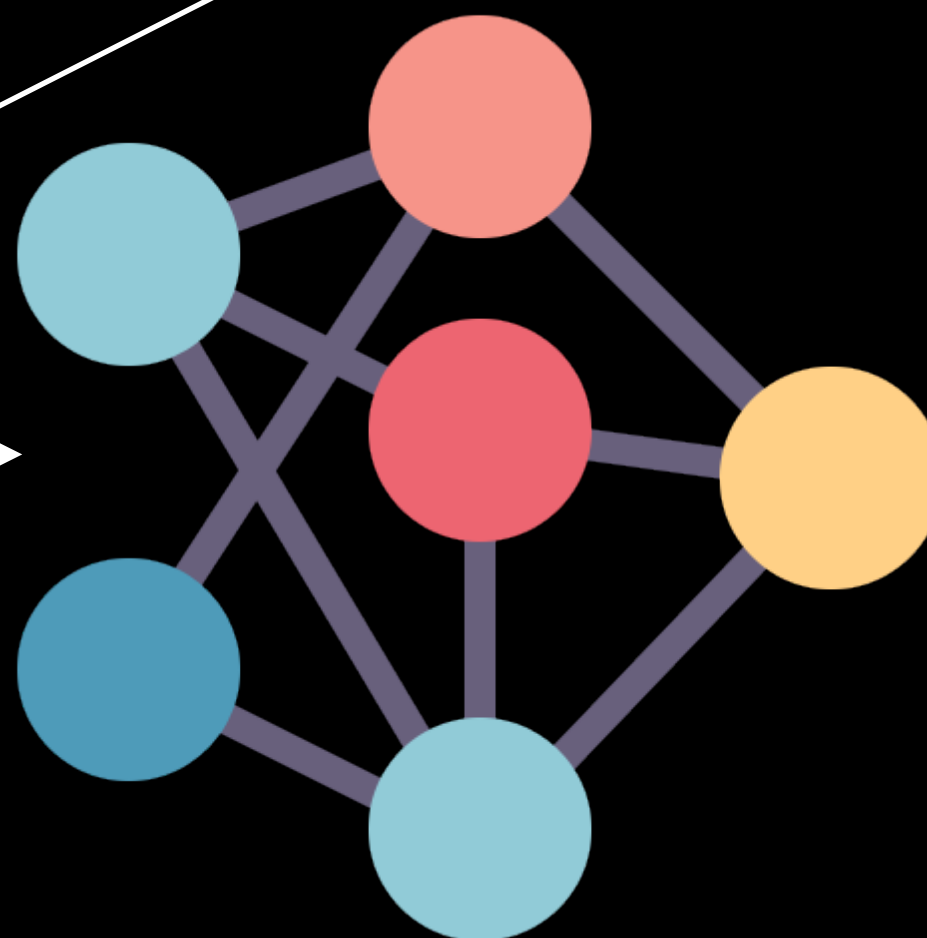
Outputs



Inputs

```
[[116, 78, 15],  
 [117, 43, 96],  
 [125, 87, 23],  
 ... ,
```

Numerical
encoding



Learns
representation
(patterns/features/weights)

```
[[0.983, 0.004, 0.013],  
 [0.110, 0.889, 0.001],  
 [0.023, 0.027, 0.985],  
 ... ,
```

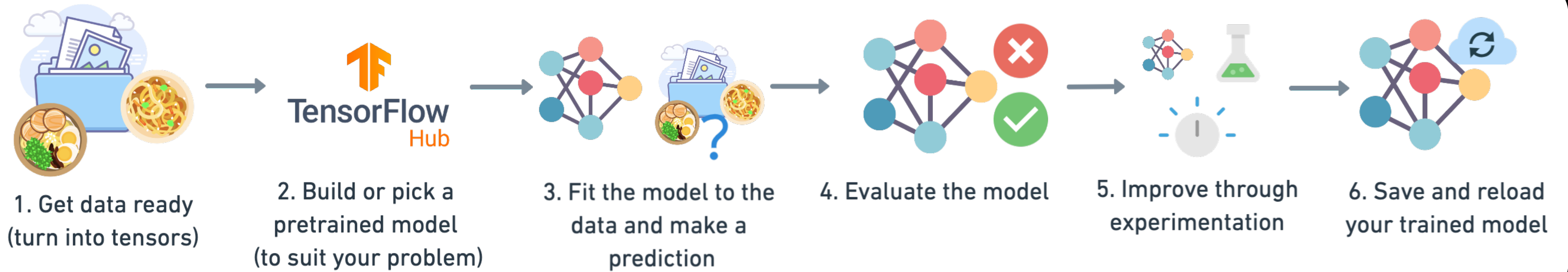
Representation
outputs

Ramen,
Spaghetti

Outputs

These are tensors!

What we're going to cover



A TensorFlow workflow

**“How should I approach
this course?”**

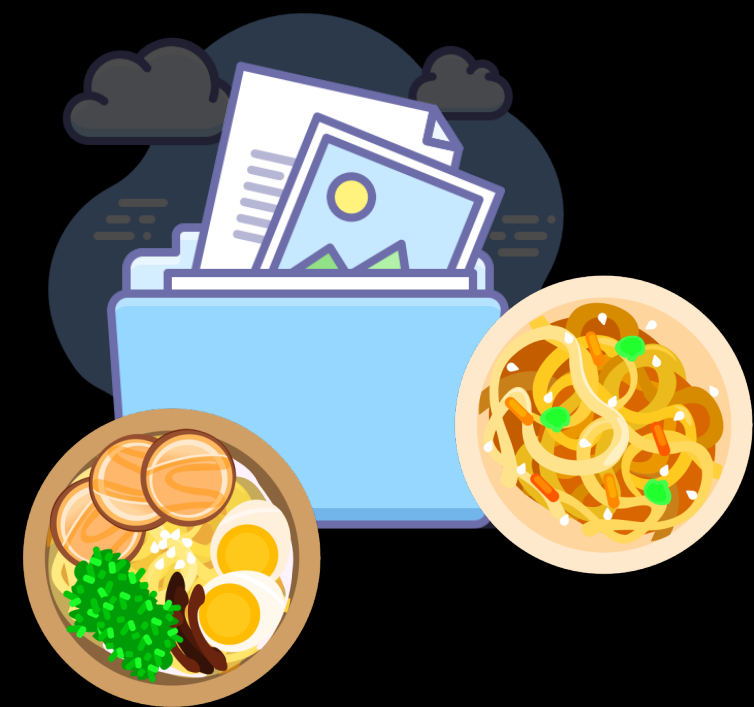
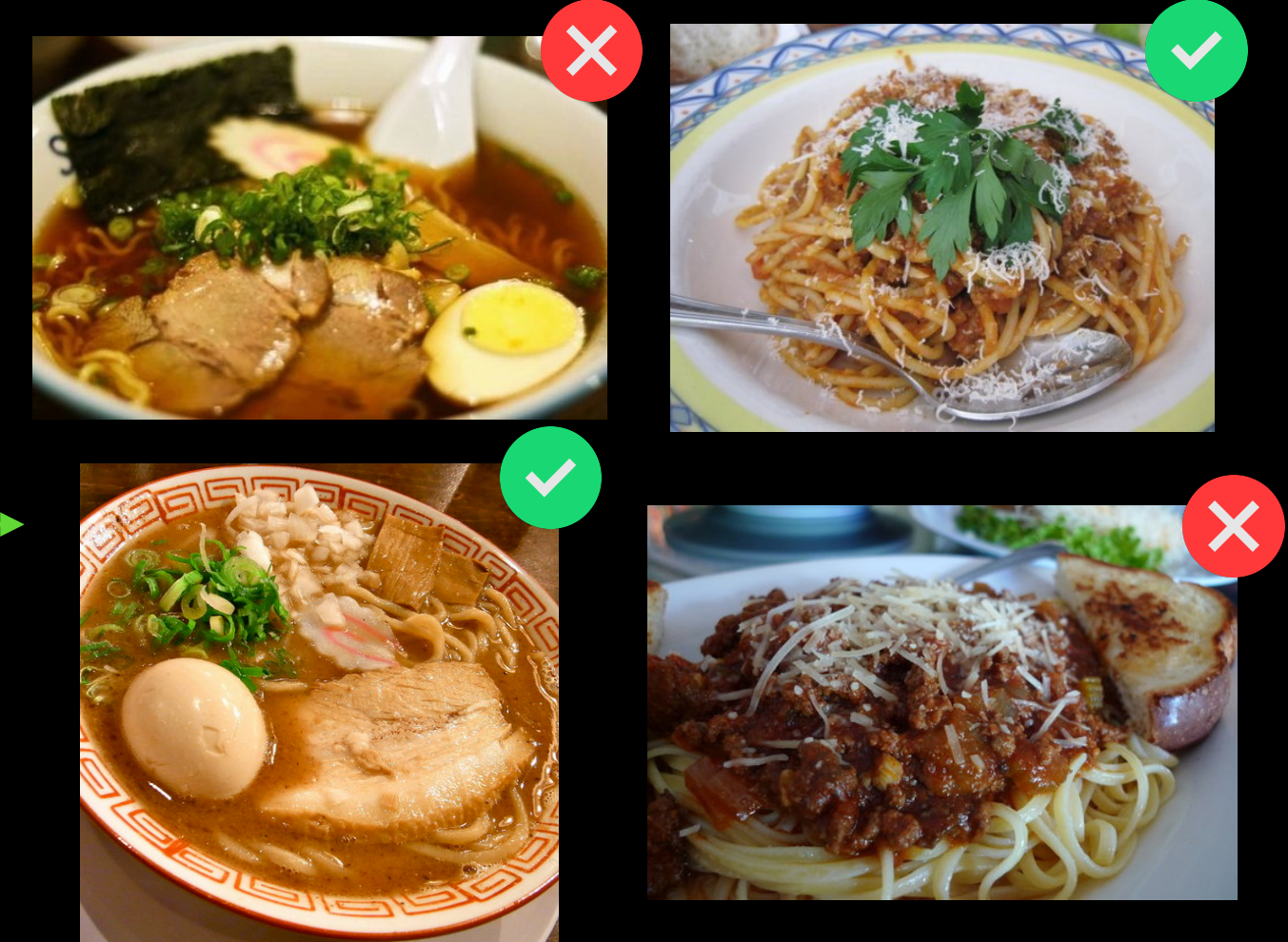
How to approach this course

- Write code (lots of it, follow along, let's make mistakes together)
 - Motto #1: "If in doubt, run the code"
- Explore & experiment
 - Motto #2: "Experiment, experiment, experiment"
 - Motto #3: "Visualize, visualize, visualize" (recreate things in ways you can understand them)
- Ask questions (including the "dumb" ones) 🤔
- Do the exercises (try them yourself before looking at the solutions) 🔧
 - This course doesn't cover everything, if you want to learn more on something, look it up
- Share your work
- 🚫 Avoid:
 - Overthinking the process
 - The "I can't learn it" mentality (that's bullsh*t)

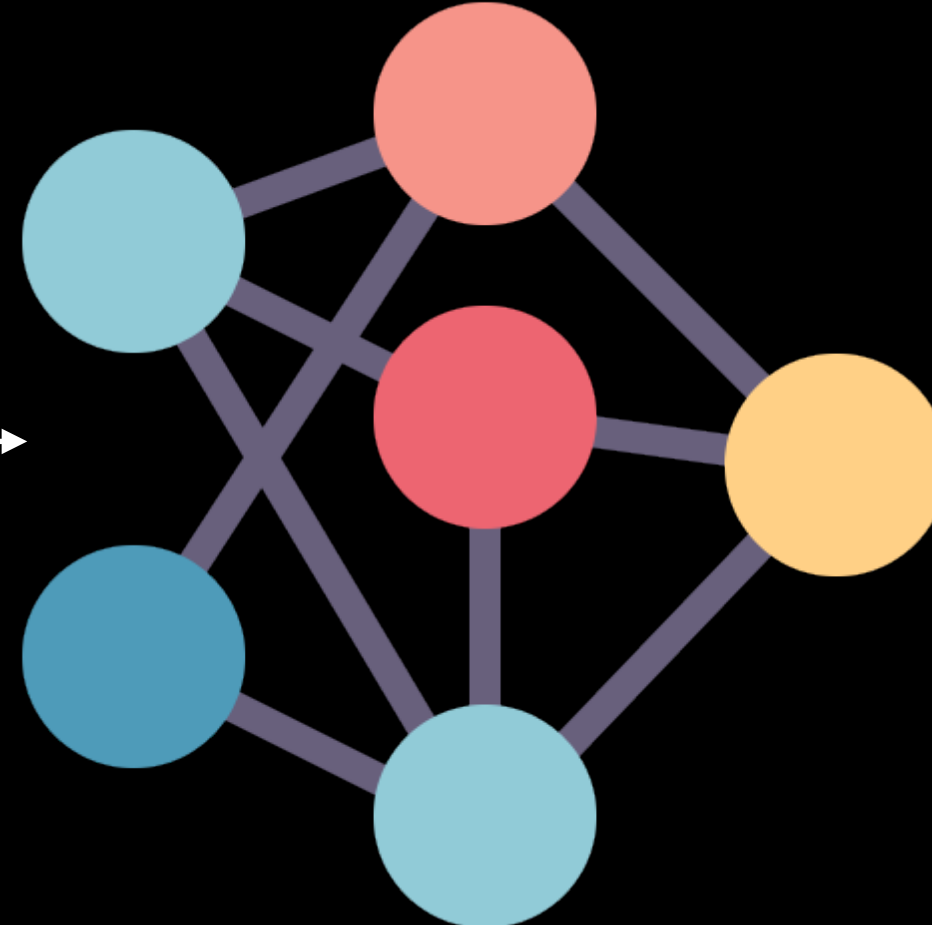
1. Initialise with random weights (only at beginning)

$[[0.092, 0.210, 0.415],$
 $[0.778, 0.929, 0.030],$
 $[0.019, 0.182, 0.555],$
 $\dots,$

2. Show examples



$[[116, 78, 15],$
 $[117, 43, 96],$
 $[125, 87, 23],$
 $\dots,$



$[[0.983, 0.004, 0.013],$
 $[0.110, 0.889, 0.001],$
 $[0.023, 0.027, 0.985],$
 $\dots,$

Ramen,
Spaghetti

3. Update representation outputs

4. Repeat with more examples

Inputs

Numerical
encoding

Learns
representation
(patterns/features/weights)

Representation
outputs

Outputs