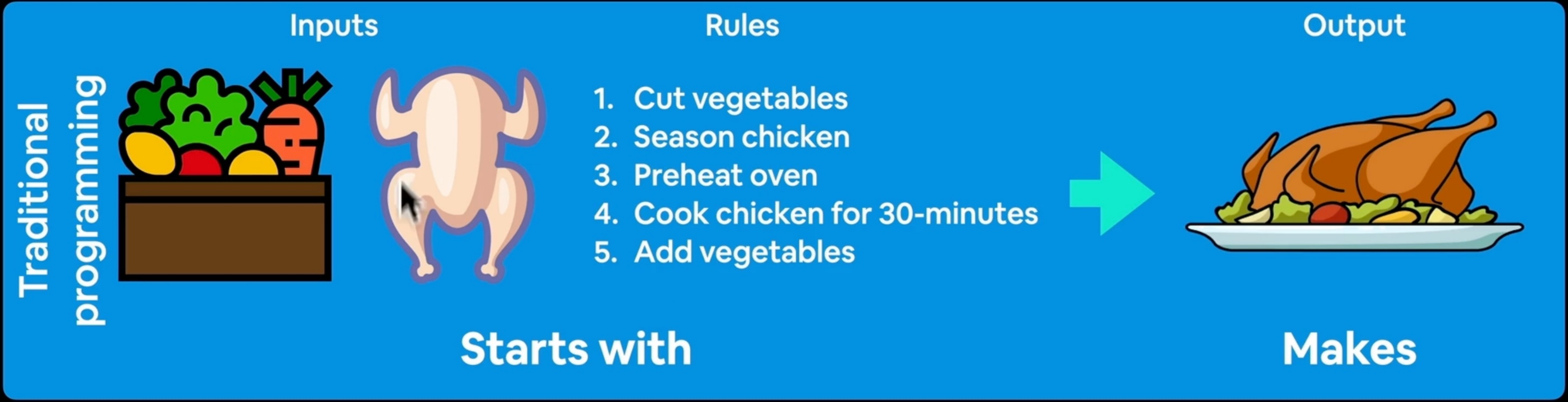


Traditional programming

Inputs

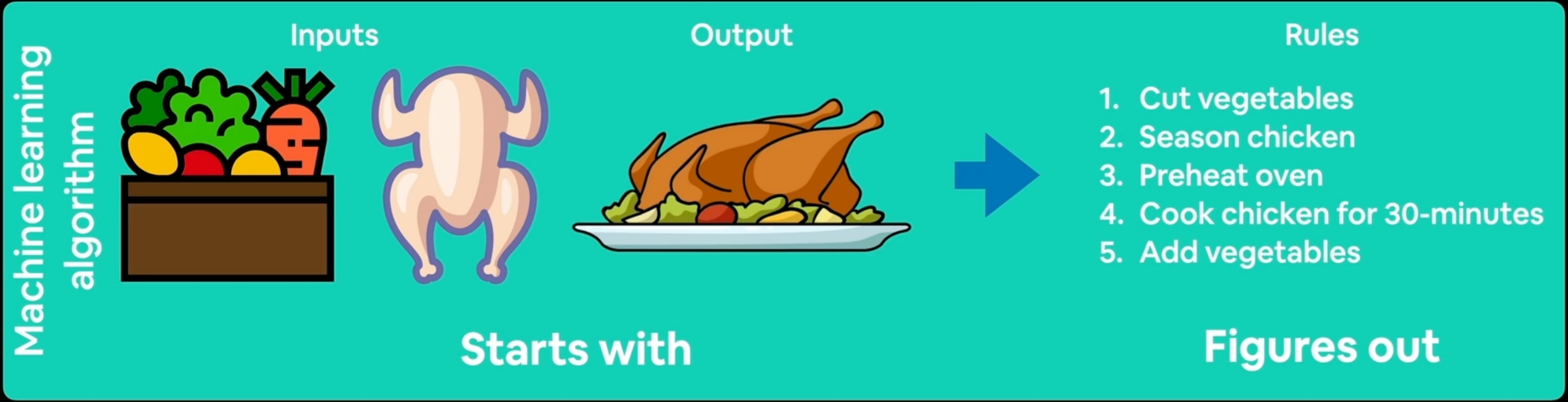
Rules

Output

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

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

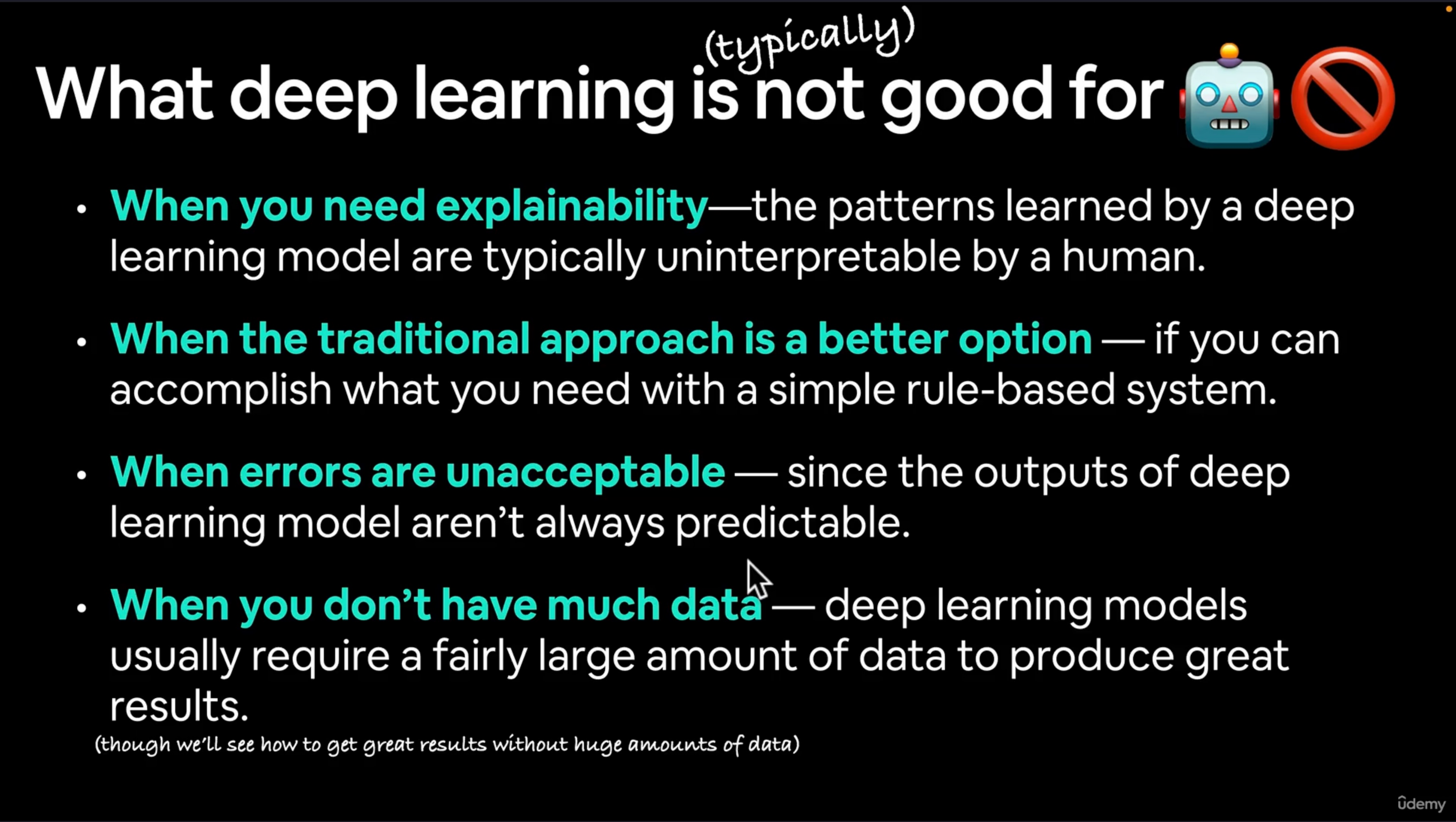
(maybe not very simple...)

“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 🤖✅

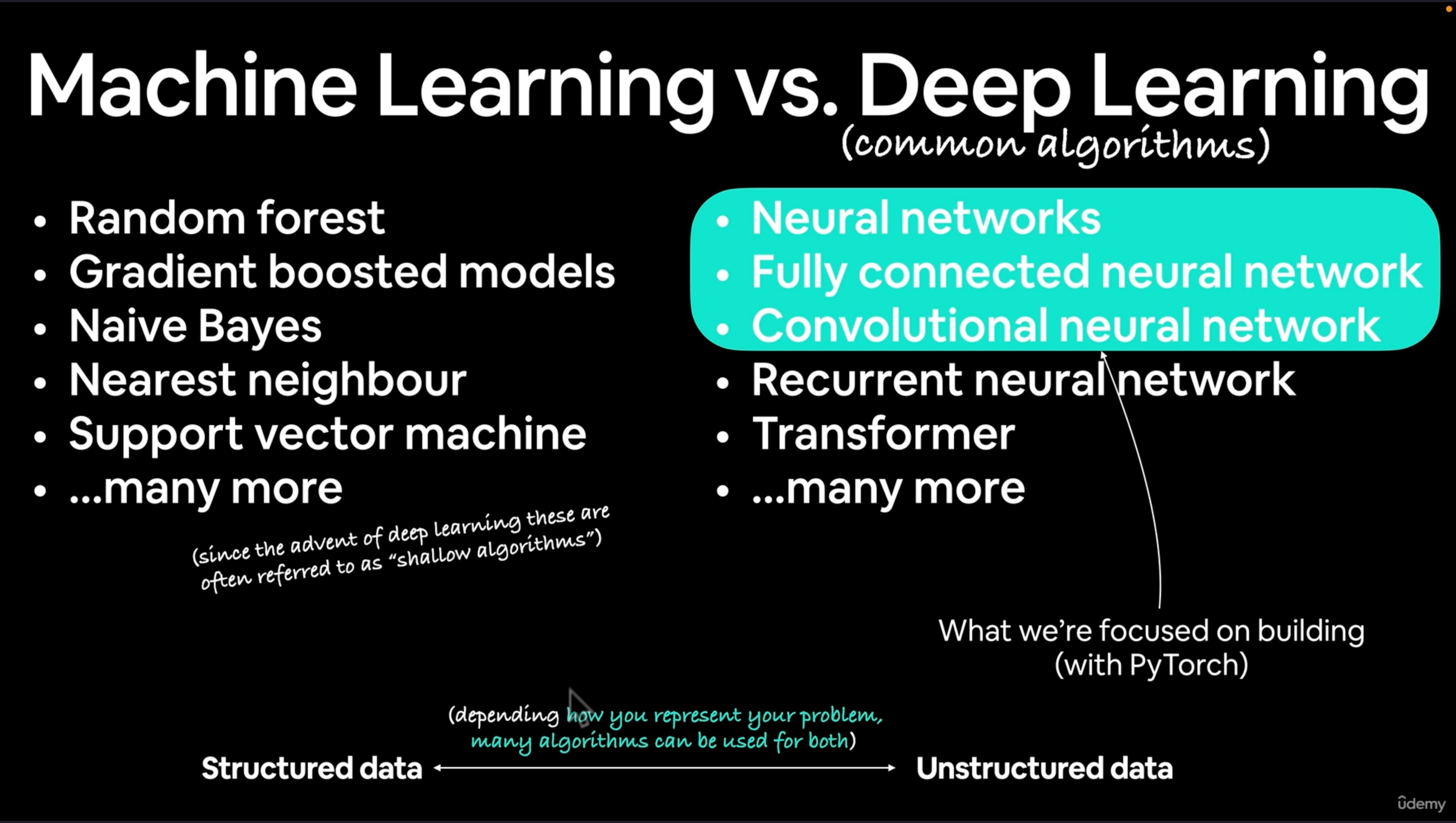
- **Problems with long lists of rules**—when the traditional approach fails, machine learning/deep learning may help.
- **Continually changing environments**—deep learning can adapt (‘learn’) to new scenarios.
- **Discovering insights within large collections of data**—can you imagine trying to hand-craft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

(common algorithms)

- Random forest
- Gradient boosted models
- Naive Bayes
- Nearest neighbour
- Support vector machine
- ...many more

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

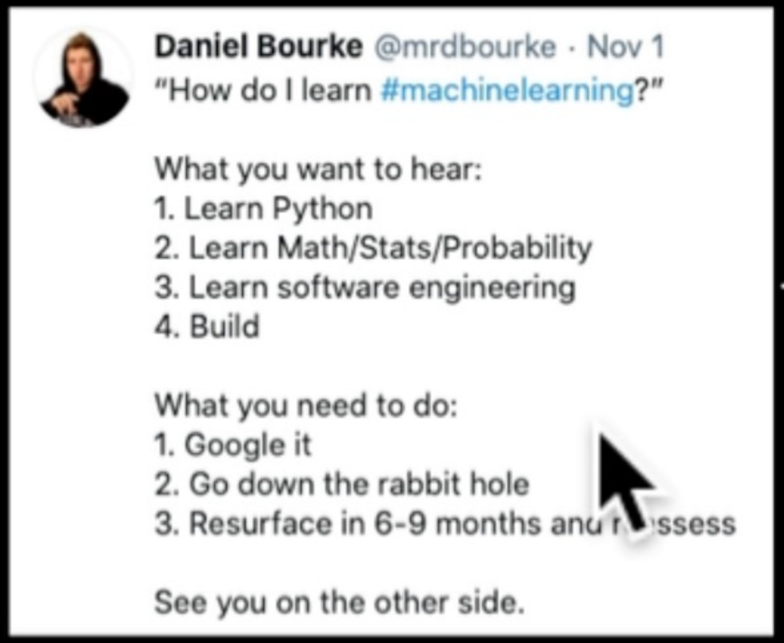
- Neural networks
- Fully connected neural network
- Convolutional neural network
- Recurrent neural network
- Transformer
- ...many more

What we're focused on building (with PyTorch)

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

Structured data ⟷ Unstructured data

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

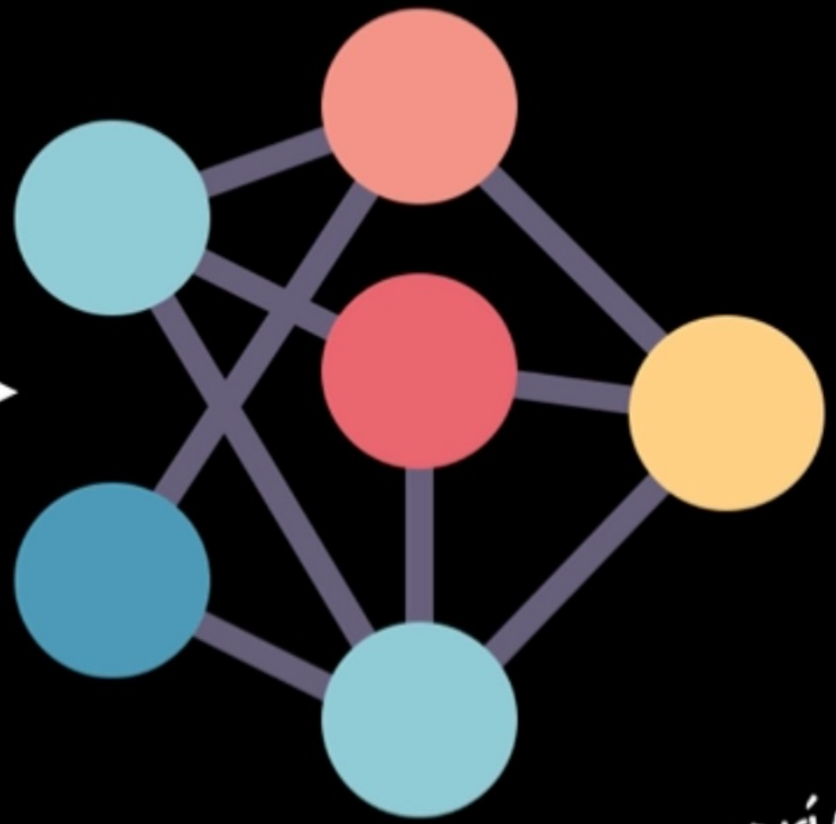
What you need to do:
1. Google it
2. Go down the rabbit hole
3. Resurface in 6-9 months and reassess

See you on the other side.

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

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

Each of these nodes is called a "hidden unit" or "neuron".



(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],
...,
```

(a human can understand these)

Ramen, Spaghetti

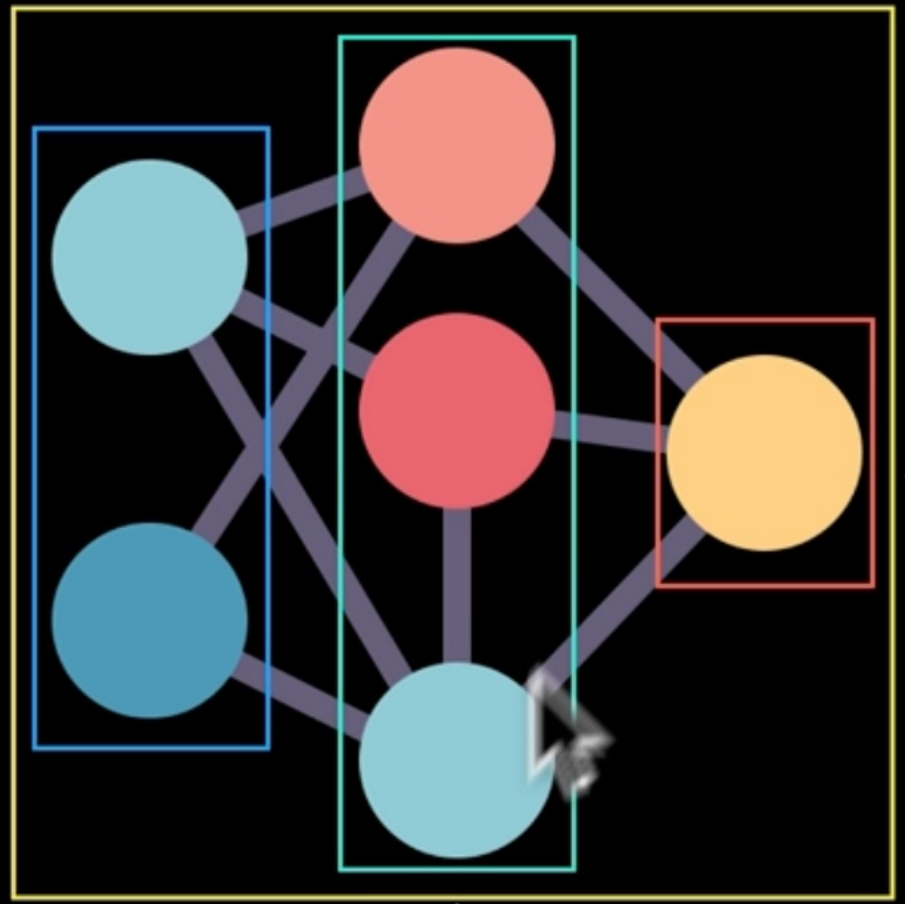
Not a diaster

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

Inputs → **Numerical encoding** → **Learns representation (patterns/features/weights)** → **Representation outputs** → **Outputs**

Anatomy of Neural Networks

Overall architecture



Input layer
(data goes in here)
# units/neurons = 2

Output layer
(outputs learned representation or prediction probabilities)
# units/neurons = 1

Hidden layer(s)
(learns patterns in data)
# units/neurons = 3

Each layer is usually combination of linear (straight line) and/or non-linear (not-straight line) functions

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

Unsupervised & Self-supervised Learning

Transfer Learning

We'll be writing code to do these,
but the style of code can be adopted across learning paradigms.