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.

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

(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

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

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

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

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

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.

What we're going to cover

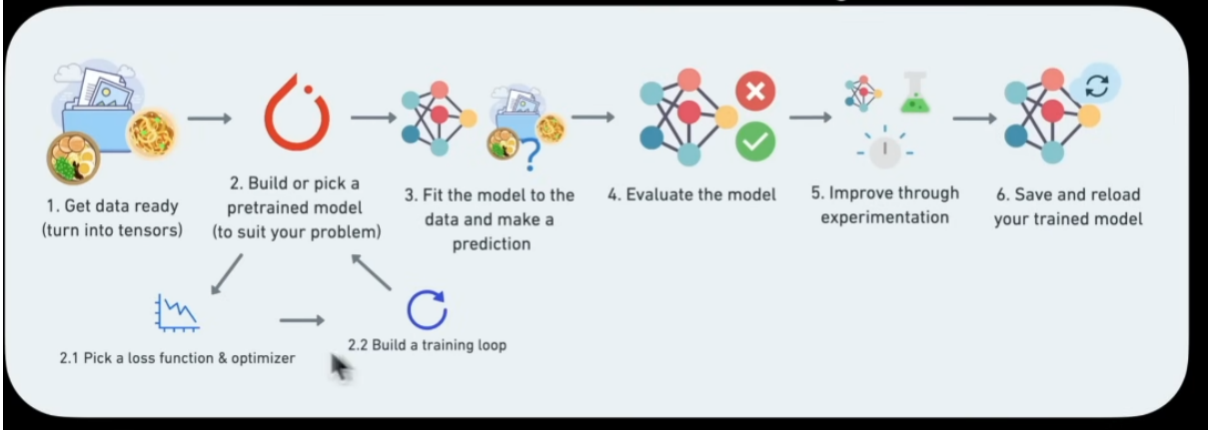
(broadly)

- Now:
  - PyTorch basics & fundamentals (dealing with tensors and tensor operations)
- Later:
  - Preprocessing data (getting it into tensors)
  - Building and using pretrained deep learning models
  - Fitting a model to the data (learning patterns)
  - Making predictions with a model (using patterns)
  - Evaluating model predictions
  - Saving and loading models
  - Using a trained model to make predictions on custom data

What we're going to cover

A PyTorch workflow

(one of many)



How to approach this course

```python
# 1. Construct a model class that subclasses nn.Module
class CircleModelV0(nn.Module):
    def __init__(self):
        super().__init__()
        # 2. Create 2 nn.Linear layers
        self.layer_1 = nn.Linear(in_features=2, out_features=5)
        self.layer_2 = nn.Linear(in_features=5, out_features=1)

    # 3. Define a forward method containing the forward pass computation
    def forward(self, x):
        # Pass the data through both layers
        return self.layer_2(self.layer_1(x))

# 4. Create an instance of the model and send it to target device
model_0 = CircleModelV0().to(device)
model_0
```

1. Code along

Motto #1: if in doubt, run the code!

Motto #2: Experiment, experiment, experiment!

2. Explore and experiment

Motto #3: Visualize, visualize, visualize!

3. Visualize what you don't understand

(including the "dumb" ones)

4. Ask questions

5. Do the exercises

6. Share your work

SUBSCRIBE