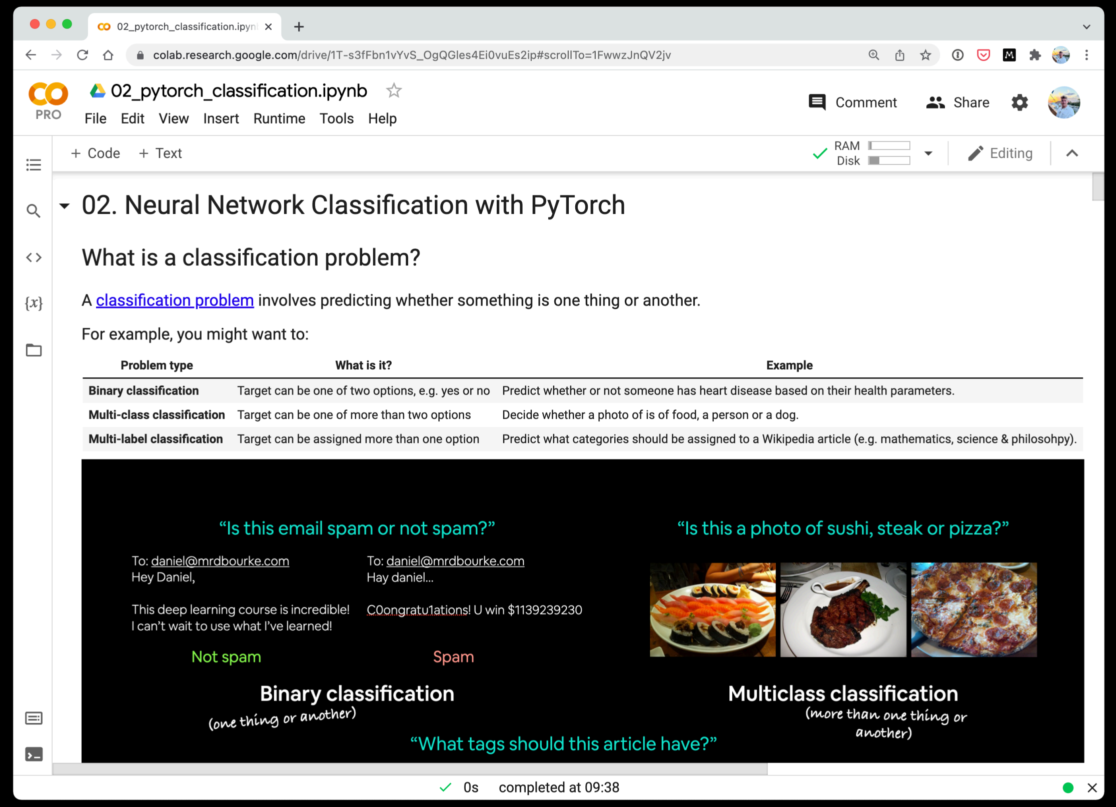
Neural Network Classification with

PyTorch

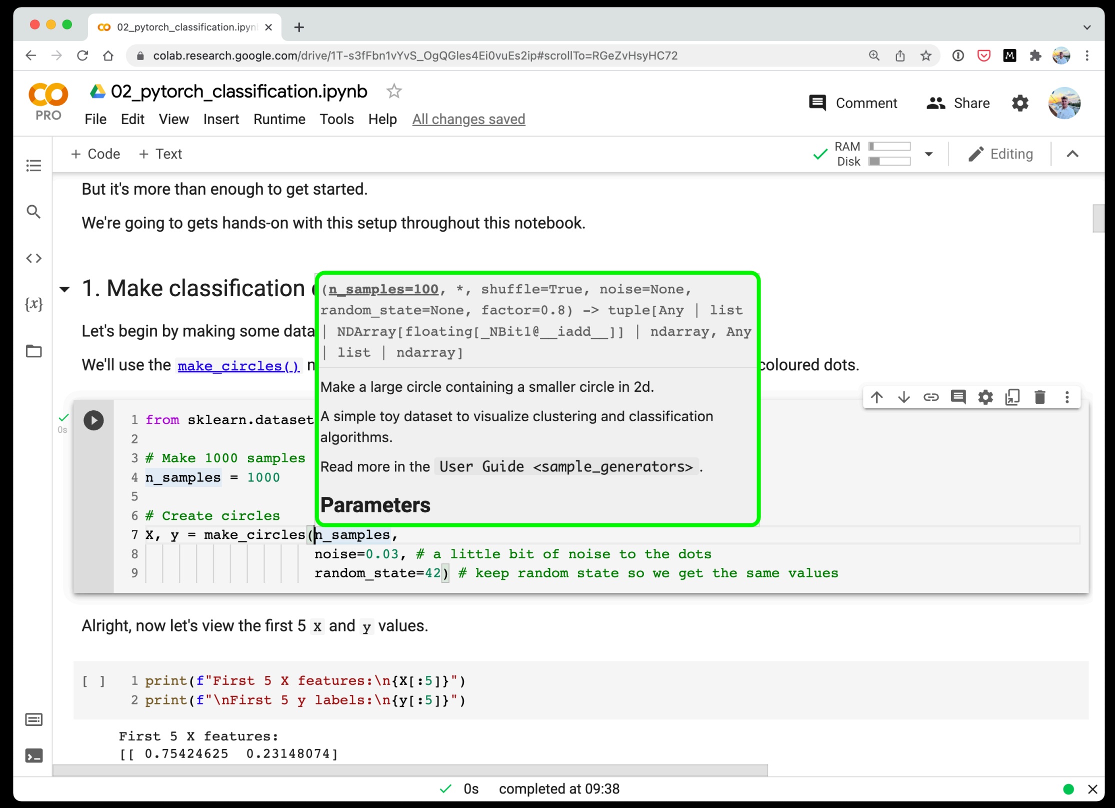
Where can you get help?

- Follow along with the code



"If in doubt, run the code"

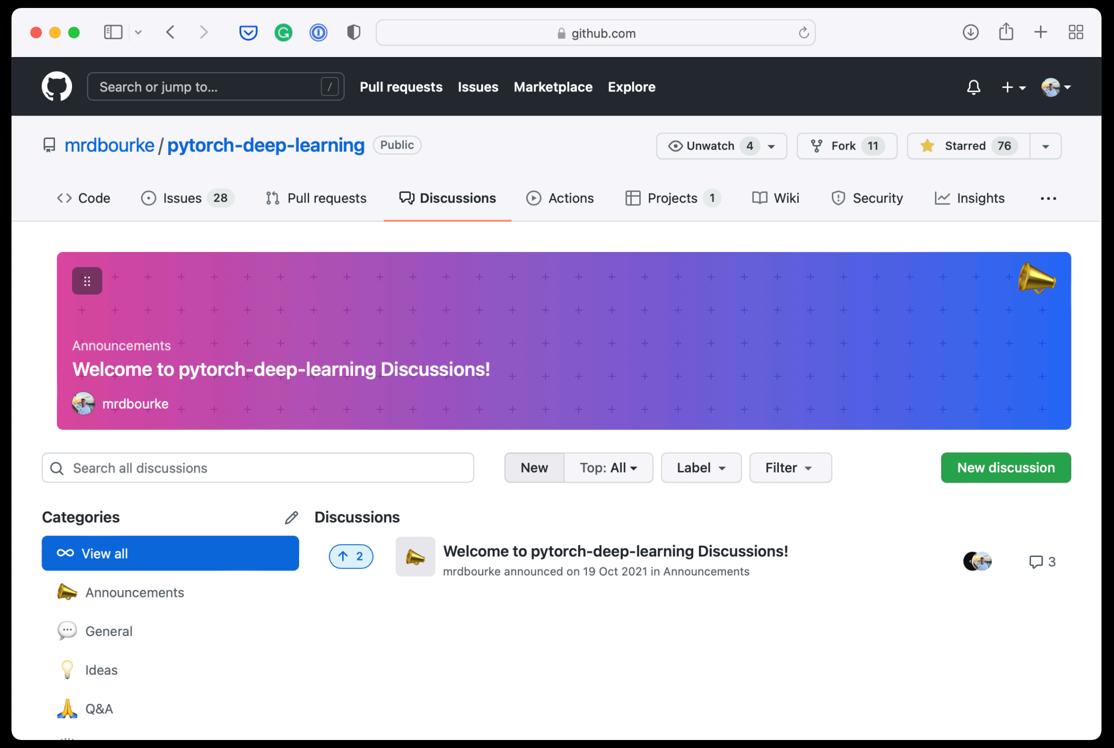
- Try it for yourself
- Press SHIFT + CMD + SPACE to read the docstring

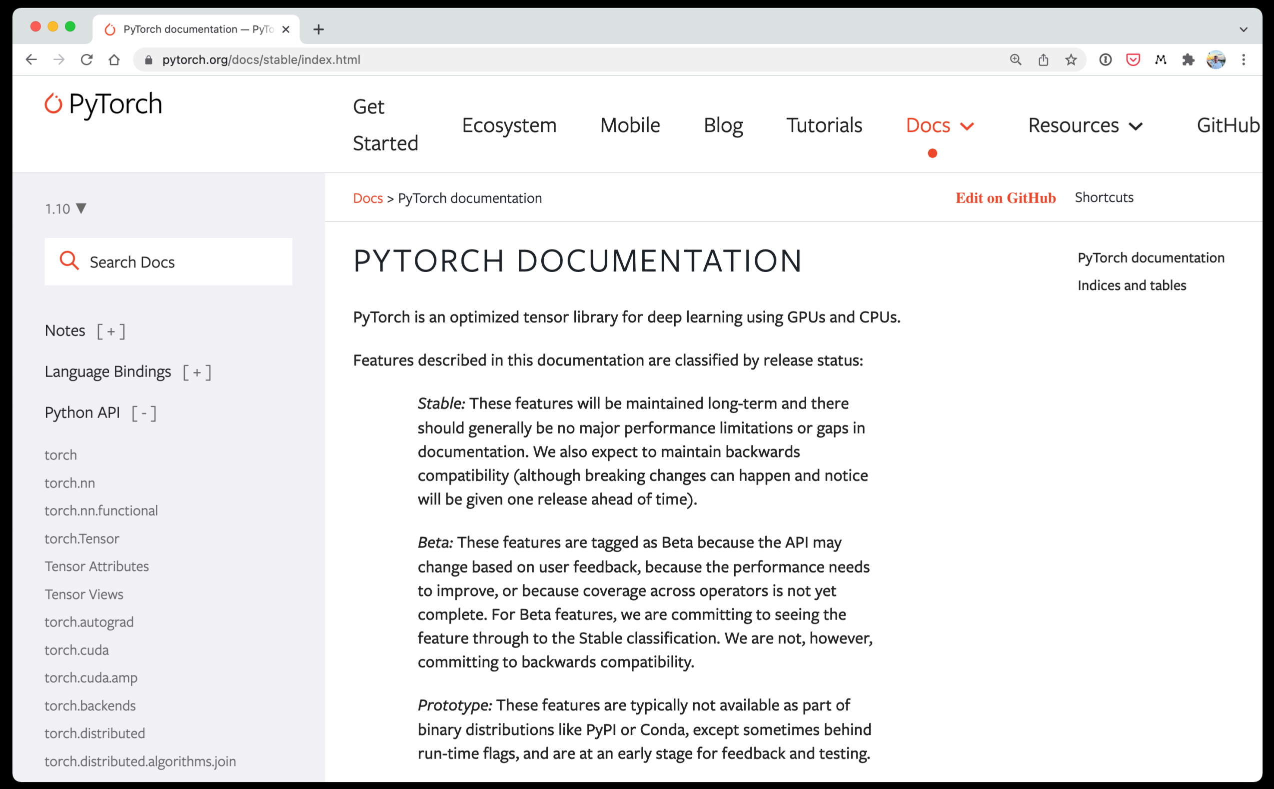


- Search for it

stackoverflow

- Try again
- Ask





https://www.github.com/mrdbourke/pytorch-deep-learning/discussions

“What is a classification problem?”

Example classification problems

"Is this email spam or not spam?"

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...

C0ongratu1ations! U win $1139239230

Spam

Binary classification

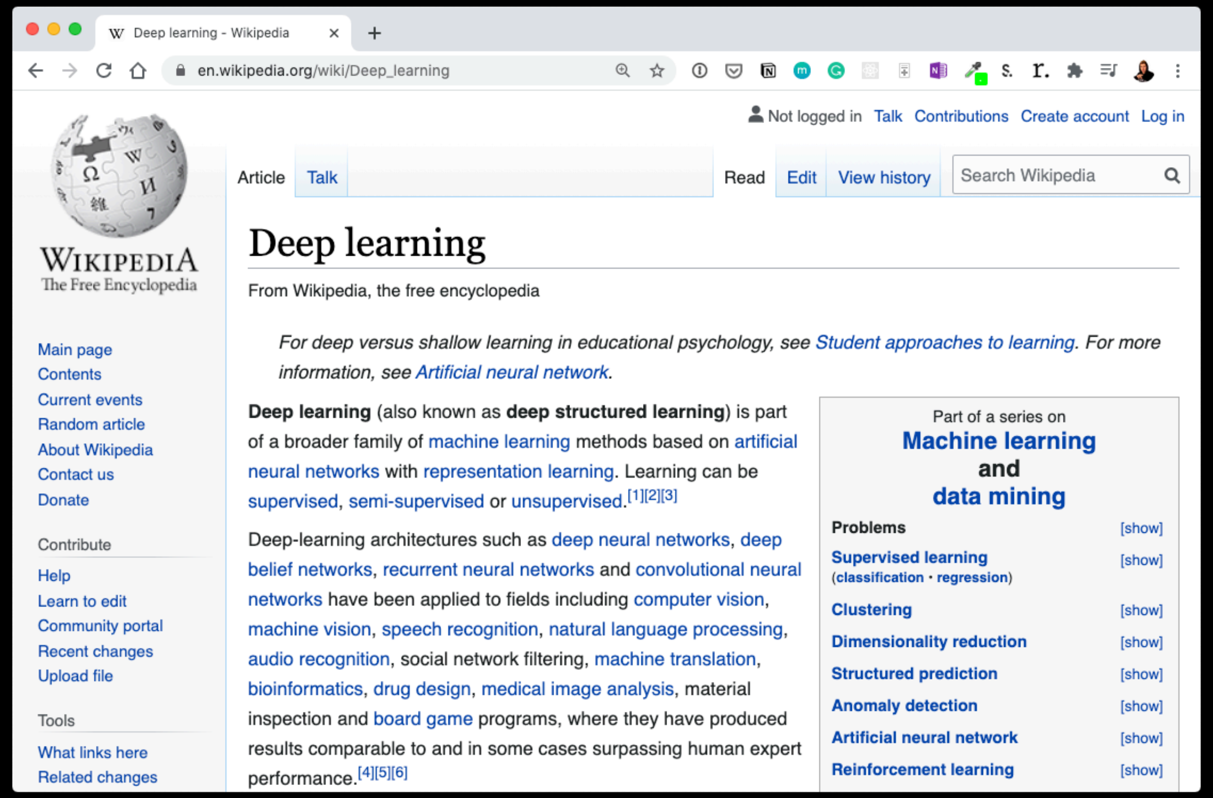
(one thing or another)

"Is this a photo of sushi, steak or pizza?"

Multiclass classification

(more than one thing or another)

"What tags should this article have?"



Machine learning
Representation learning
Artificial intelligence

(multiple label options per sample)

Multilabel classification

Binary vs. Multi-class Classification

Binary classification
(one thing or another)

Multiclass classification
(more than one thing or another)

What we're going to cover

(broadly)

- Architecture of a neural network classification model
- Input shapes and output shapes of a classification model (features and labels)
- Creating custom data to view, fit on and predict on
- Steps in modelling
  - Creating a model, setting a loss function and optimiser, creating a training loop, evaluating a model
- Saving and loading models
- Harnessing the power of non-linearity
- Different classification evaluation methods

(we'll be cooking up lots of code!)

How: 👩‍🍳 👩‍🔬

Classification inputs and outputs

W = 224
H = 224
C = 3

(c = colour channels, R, G, B)

224

224

Sushi 🍣
Steak 🥩
Pizza 🍕

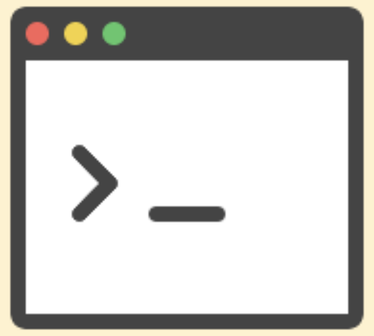
Actual output

```
[[0.31, 0.62, 0.44…],
[0.92, 0.03, 0.27…],
[0.25, 0.78, 0.07…],
…,
```

(normalized pixel values)

Numerical encoding

Inputs



Machine Learning Algorithm

(often already exists, if not, you can build one)

Outputs

🍣 🥩 🍕

```
[[0.97, 0.00, 0.03],
[0.81, 0.14, 0.05],
[0.03, 0.07, 0.90],
…,
```

Predicted output

(comes from looking at lots of these)

Input and output shapes

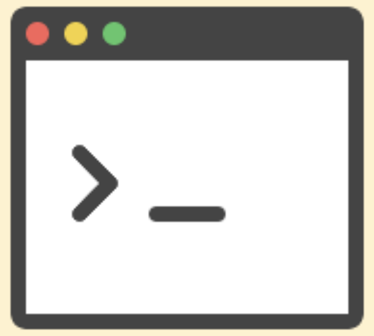
(for an image classification example)

224

224

[[0.31, 0.62, 0.44…],
[0.92, 0.03, 0.27…],
[0.25, 0.78, 0.07…],
…,

Inputs



Machine Learning Algorithm

Outputs

[0.97, 0.00, 0.03]

(prediction probabilities)

(gets represented as a tensor)

[batch_size, colour_channels, width, height]

Shape = [None, 3, 224, 224]
or
Shape = [32, 3, 224, 224]

(32 is a very common batch size)

Shape = [3]

These will vary depending on the problem you're working on.

(typical)

Architecture of a classification model

(we're going to be building lots of these)

Hyperparameter	Binary Classification	Multiclass classification
Input layer shape (`in_features`)	Same as number of features (e.g. 5 for age, sex, height, weight, smoking status in heart disease prediction)	Same as binary classification
Hidden layer(s)	Problem specific, minimum = 1, maximum = unlimited	Same as binary classification
Neurons per hidden layer	Problem specific, generally 10 to 512	Same as binary classification
Output layer shape (`out_features`)	1 (one class or the other)	1 per class (e.g. 3 for food, person or dog photo)
Hidden layer activation	Usually ReLU (rectified linear unit) but can be many others	Same as binary classification
Output activation	Sigmoid (`torch.sigmoid` in PyTorch)	Softmax (`torch.softmax` in PyTorch)
Loss function	Binary crossentropy (`torch.nn.BCELoss` in PyTorch)	Cross entropy (`torch.nn.CrossEntropyLoss` in PyTorch)
Optimizer	SGD (stochastic gradient descent), Adam (see `torch.optim` for more options)	Same as binary classification

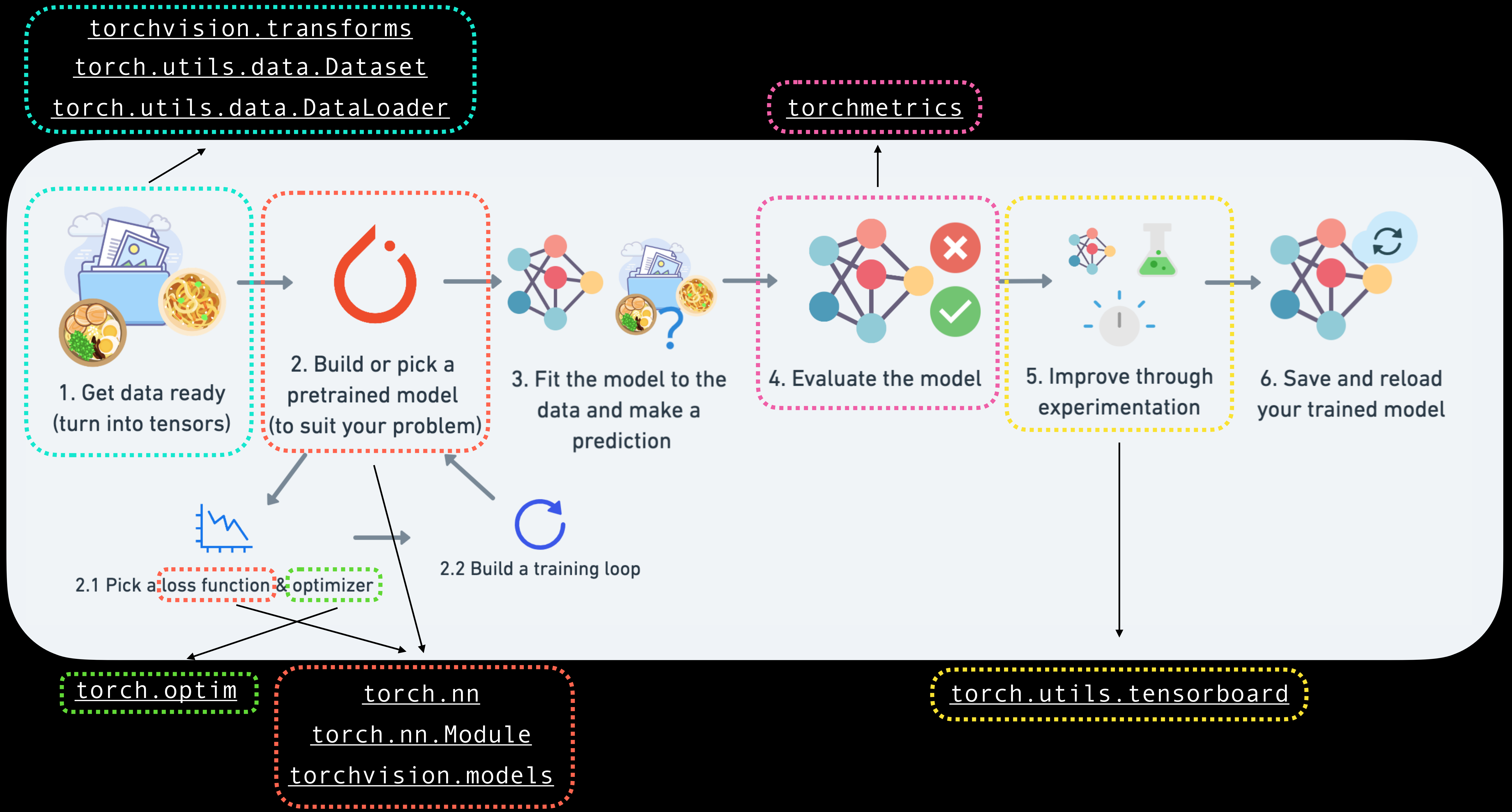
→

```python
# Create a model
model = nn.Sequential(
    nn.Linear(in_features=3, out_features=100),
    nn.Linear(in_features=100, out_features=100),
    nn.ReLU(),
    nn.Linear(in_features=100, out_features=3)
)

# Setup a loss function and optimizer
loss_fn = nn.BCEWithLogitsLoss()
optimizer = torch.optim.SGD(params=model.parameters(),
                            lr=0.001)

# Training code...

# Testing code...
```

→

Sushi 🍣
Steak 🥩
Pizza 🍕

Let’s code!



See more: https://pytorch.org/tutorials/beginner/ptcheat.html

Improving a model

(from a model's perspective)

```python
# Create a model
model = nn.Sequential(
    nn.Linear(in_features=3, out_features=100),
    nn.Linear(in_features=100, out_features=100),
    nn.ReLU(),
    nn.Linear(in_features=100, out_features=3)
)

# Setup a loss function and optimizer
loss_fn = nn.BCEWithLogitsLoss()
optimizer = torch.optim.SGD(params=model.parameters(),
                            lr=0.001)

# Training code...
epochs = 10

# Testing code...
```

Smaller model

```python
# Create a larger model
model = nn.Sequential(
    nn.Linear(in_features=3, out_features=128),
    nn.ReLU(),
    nn.Linear(in_features=128, out_features=256),
    nn.ReLU(),
    nn.Linear(in_features=256, out_features=128),
    nn.ReLU(),
    nn.Linear(in_features=128, out_features=3)
)

# Setup a loss function and optimizer
loss_fn = nn.BCEWithLogitsLoss()
optimizer = torch.optim.Adam(params=model.parameters(),
                             lr=0.0001)

# Training code...
epochs = 100

# Testing code...
```

Larger model

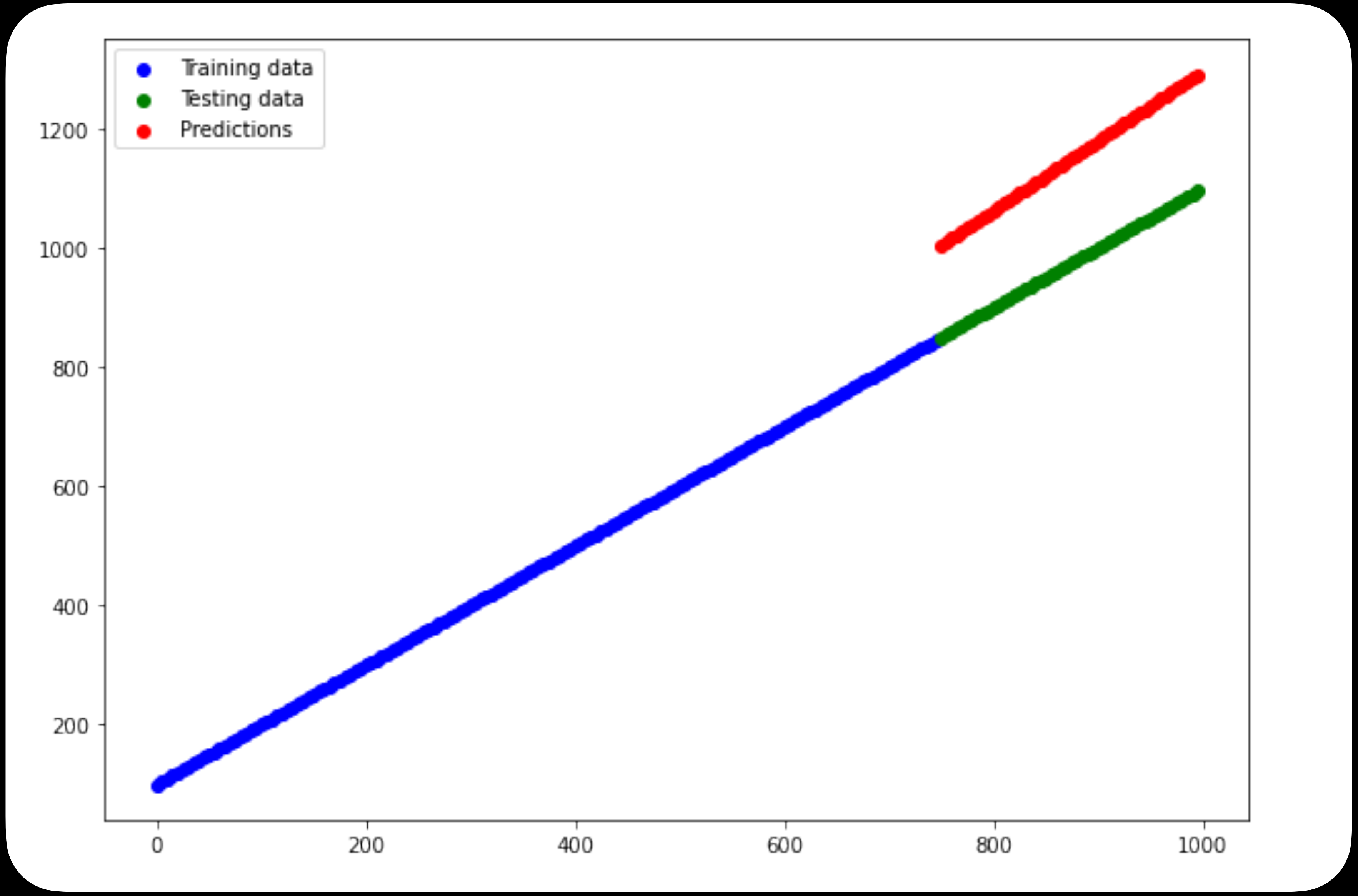
Common ways to improve a deep model:

- Adding layers
- Increase the number of hidden units
- Change/add activation functions
- Change the optimization function
- Change the learning rate
- Fitting for longer

(because you can alter each of these, they're hyperparameters)

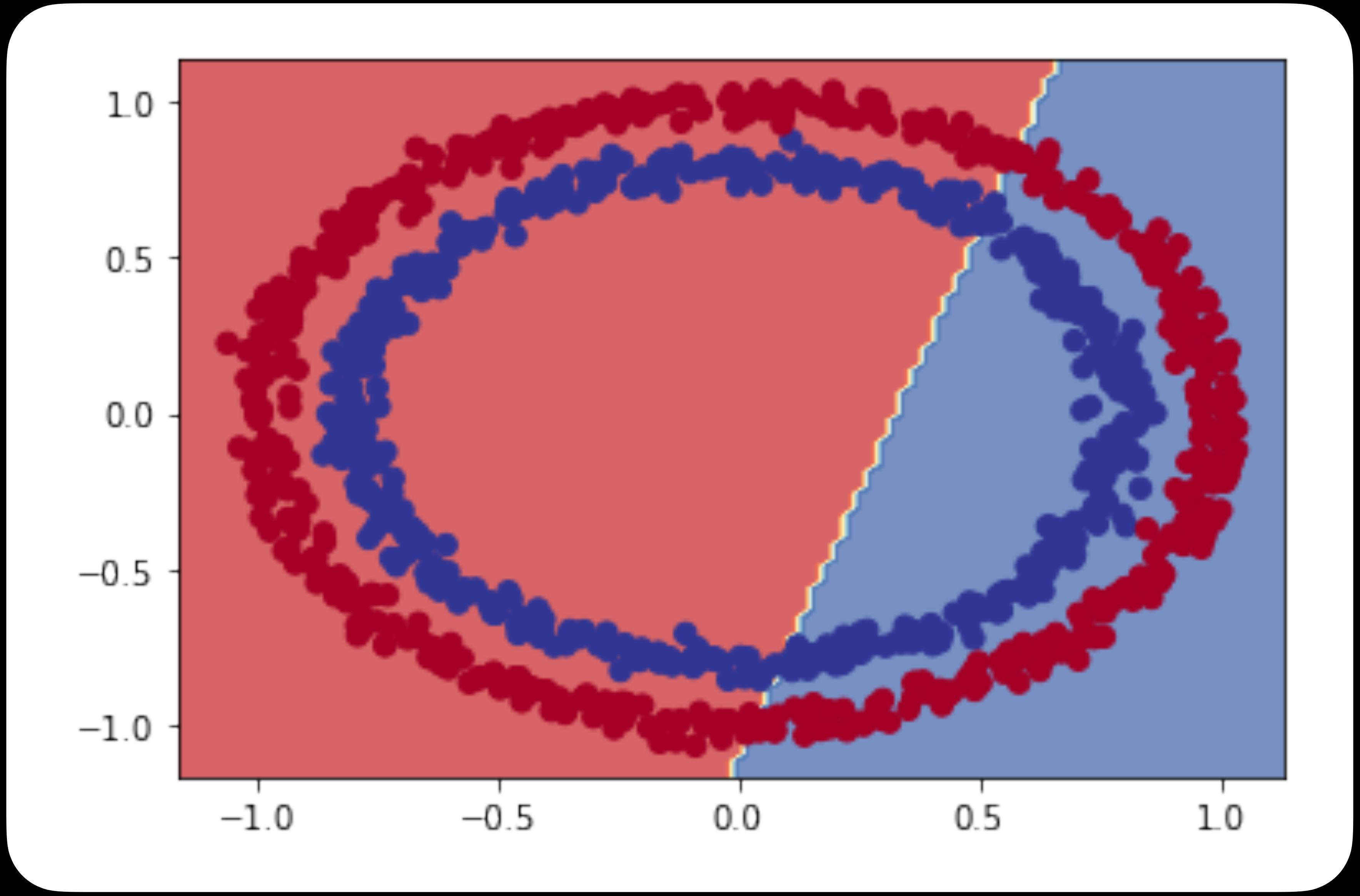
The missing piece: Non-linearity

🤔 "What could you draw if you had an unlimited amount of straight (linear) and non-straight (non-linear) lines?"



Linear data

(possible to model with straight lines)



Non-linear data

(not possible to model with straight lines)

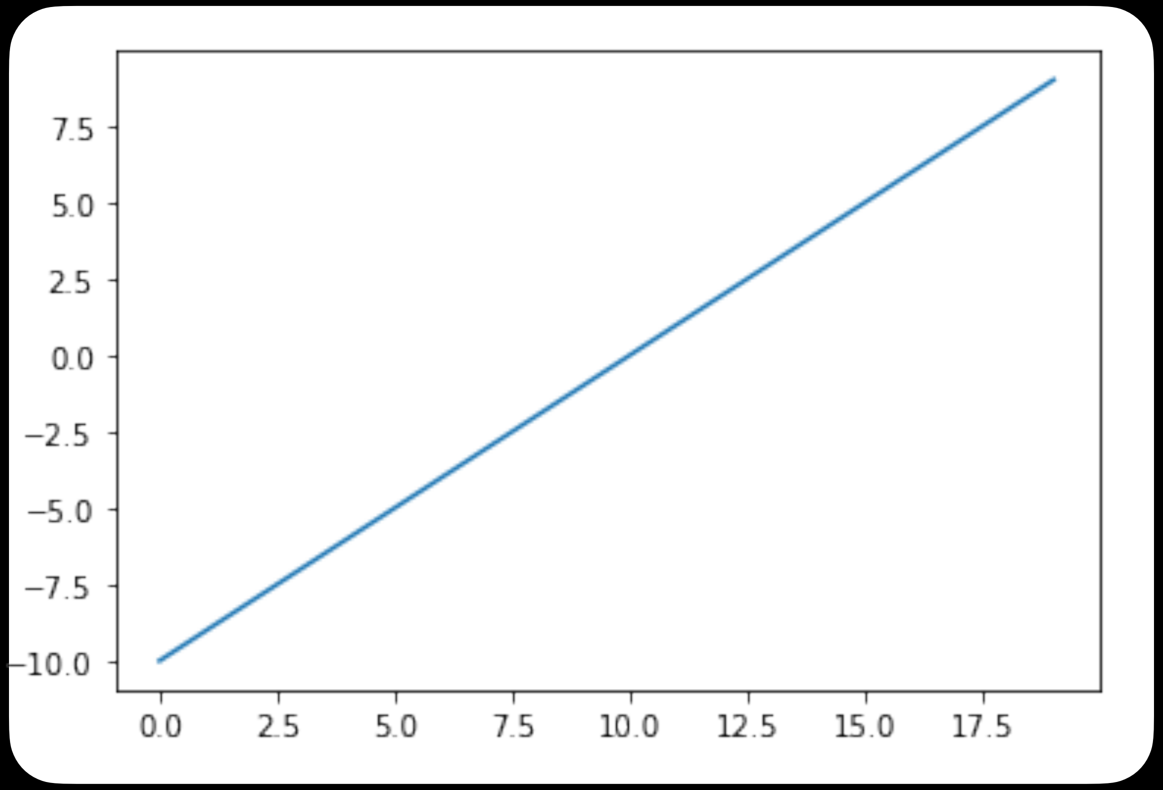
The missing piece: Non-linearity

`A = torch.arange(-10, 10)`

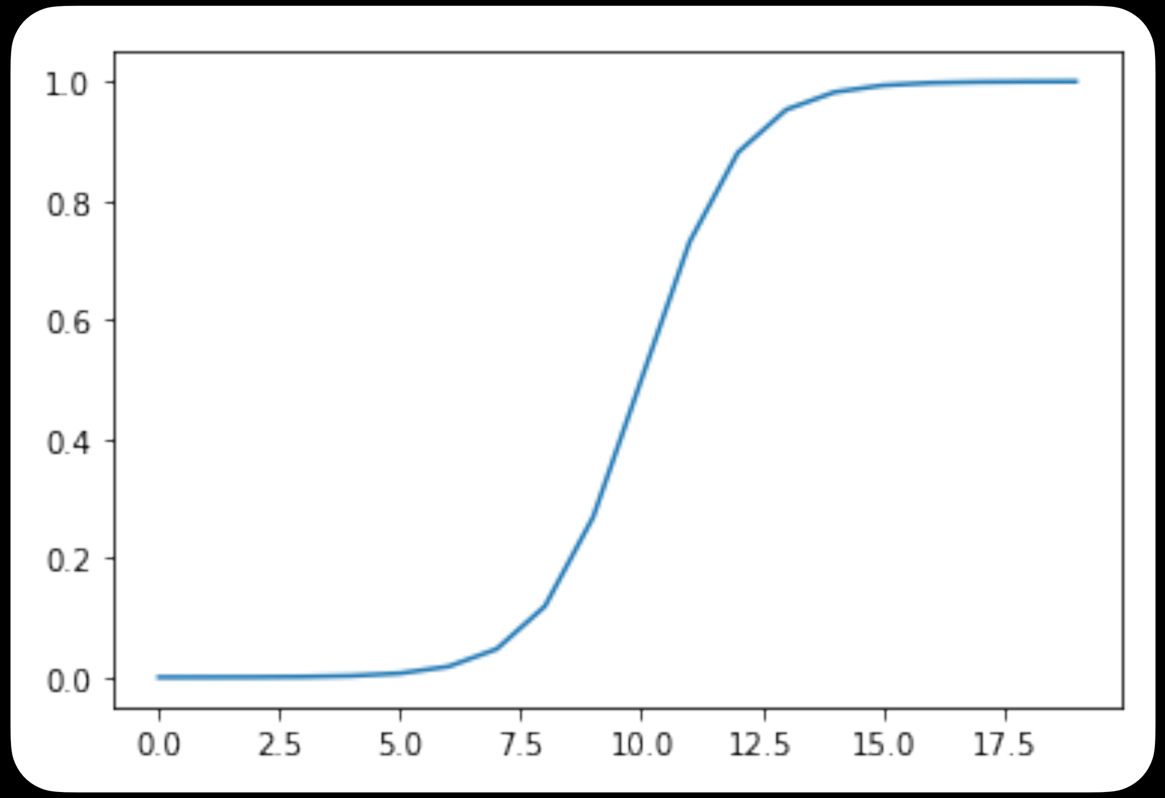
`plt.plot(A)`

`plt.plot(torch.sigmoid(A))`

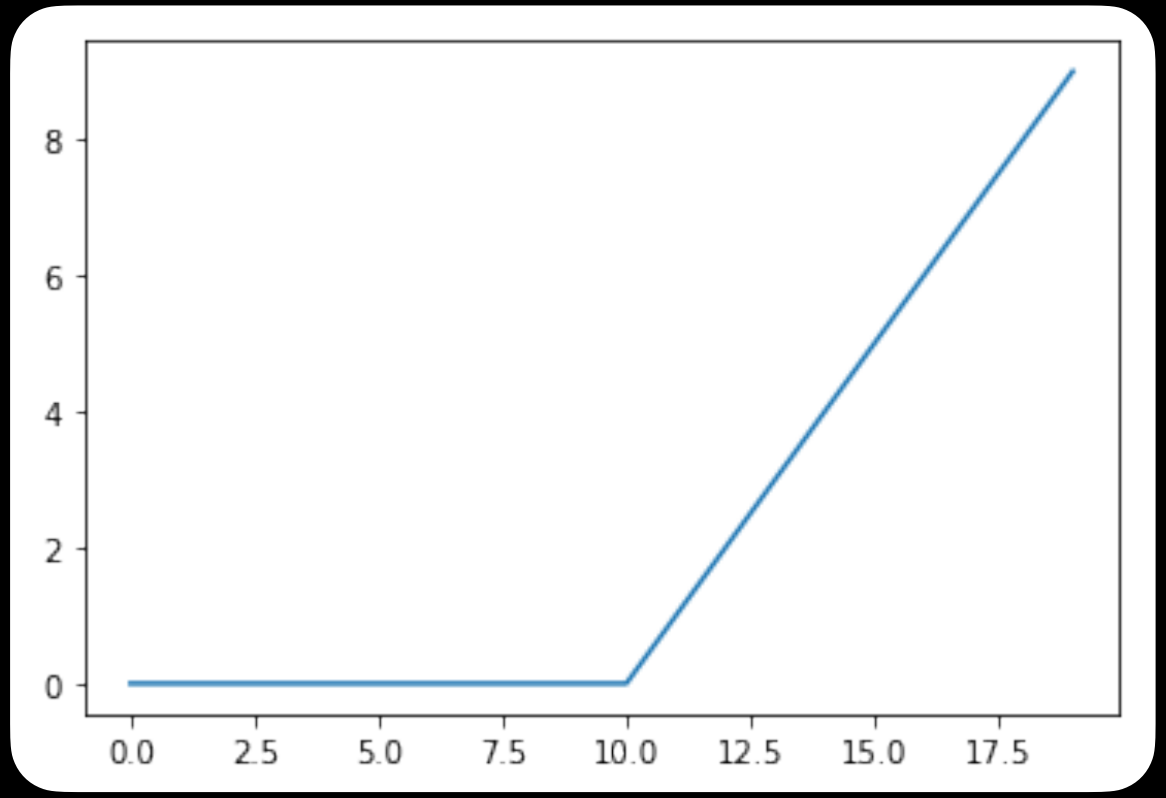
`plt.plot(torch.relu(A))`



Linear activation
(same as original values)



Sigmoid activation
(non-linear)

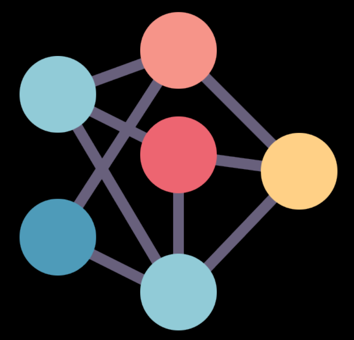


ReLU activation
(non-linear)

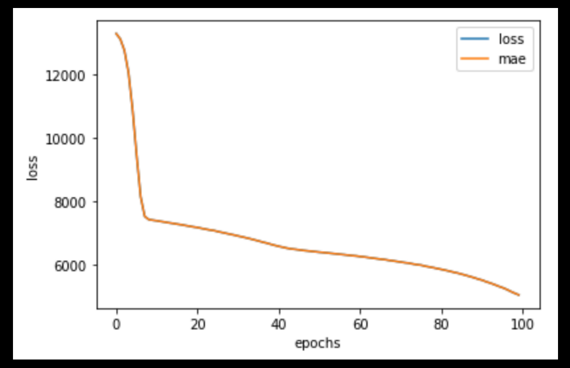
The machine learning explorer's motto

"Visualize, visualize, visualize"

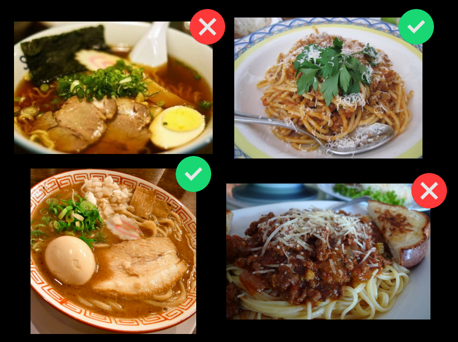
Data



Model



Training



Predictions

It's a good idea to visualize these as often as possible.

The machine learning practitioner's motto

"Experiment, experiment, experiment"

(try lots of things and see what tastes good)

Steps in modelling with PyTorch

```python
# Create a model
model = nn.Sequential(
    nn.Linear(in_features=3, out_features=100),
    nn.Linear(in_features=100, out_features=100),
    nn.ReLU(),
    nn.Linear(in_features=100, out_features=3)
)

# Setup a loss function and optimizer
loss_fn = nn.BCEWithLogitsLoss()
optimizer = torch.optim.SGD(params=model.parameters(),
                            lr=0.001)

# Training code...

# Testing code...
```

1. Construct or import a pretrained model relevant to your problem
2. Prepare the loss function, optimizer and training loop
    - Loss — how wrong your model's predictions are compared to the truth labels (you want to minimise this).
    - Optimizer — how your model should update its internal patterns to better its predictions.
3. Fit the model to the training data so it can discover patterns
    - Epochs — how many times the model will go through all of the training examples.
4. Evaluate the model on the test data (how reliable are our model's predictions?)

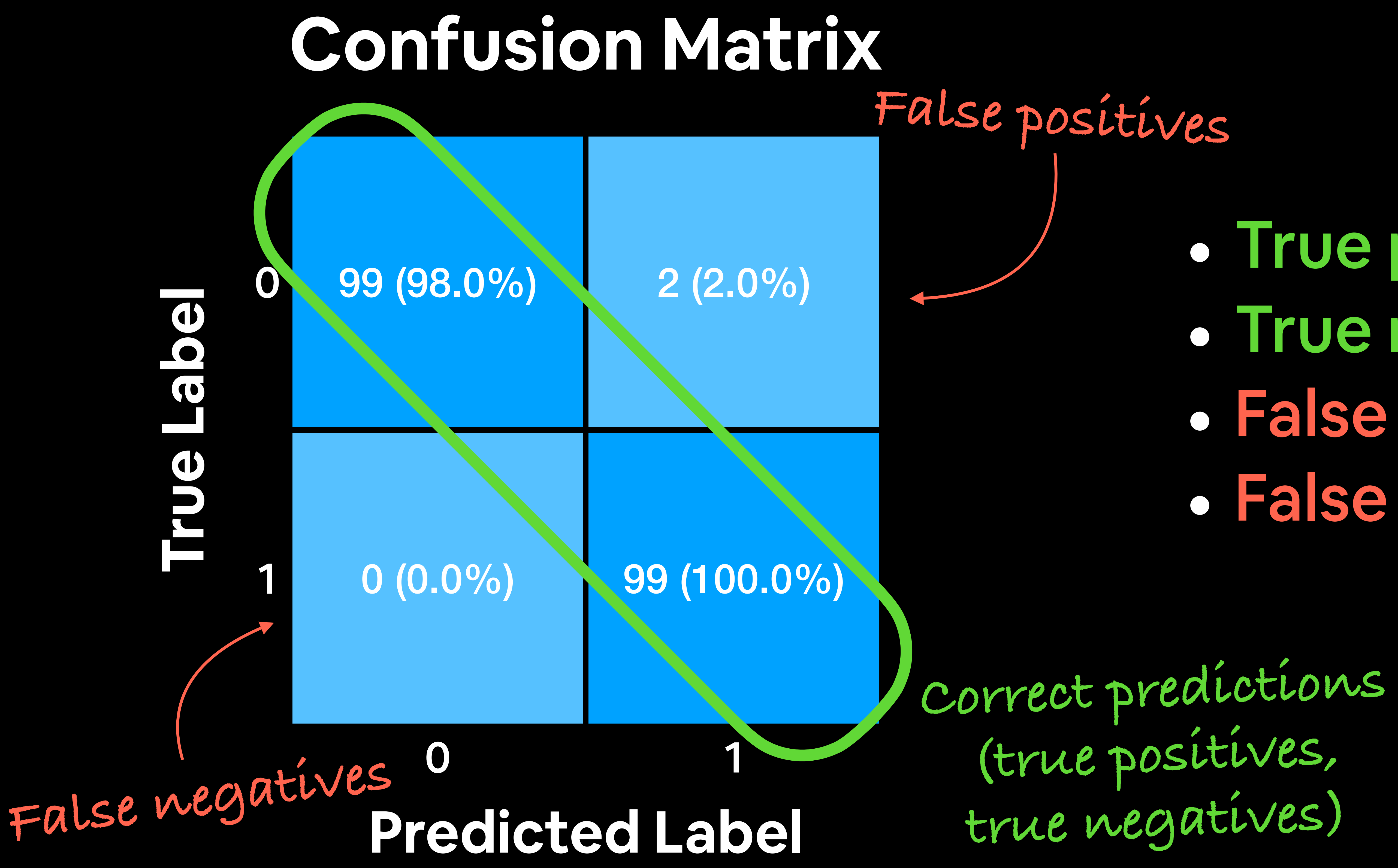
(some common)

Classification evaluation methods

Key: **tp** = True Positive, **tn** = True Negative, **fp** = False Positive, **fn** = False Negative

Metric Name	Metric Forumla	Code	When to use
Accuracy	$\textbf{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$	`torchmetrics.Accuracy()` or `sklearn.metrics.accuracy_score()`	Default metric for classification problems. Not the best for imbalanced classes.
Precision	$\textbf{Precision} = \frac{tp}{tp + fp}$	`torchmetrics.Precision()` or `sklearn.metrics.precision_score()`	Higher precision leads to less false positives.
Recall	$\textbf{Recall} = \frac{tp}{tp + fn}$	`torchmetrics.Recall()` or `sklearn.metrics.recall_score()`	Higher recall leads to less false negatives.
F1-score	$\textbf{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$	`torchmetrics.F1Score()` or `sklearn.metrics.f1_score()`	Combination of precision and recall, usually a good overall metric for a classification model.
Confusion matrix	NA	`torchmetrics.ConfusionMatrix()`	When comparing predictions to truth labels to see where model gets confused. Can be hard to use with large numbers of classes.

Anatomy of a confusion matrix



- True positive = model predicts 1 when truth is 1
- True negative = model predicts 0 when truth is 0
- False positive = model predicts 1 when truth is 0
- False negative = model predicts 0 when truth is 1

Three datasets

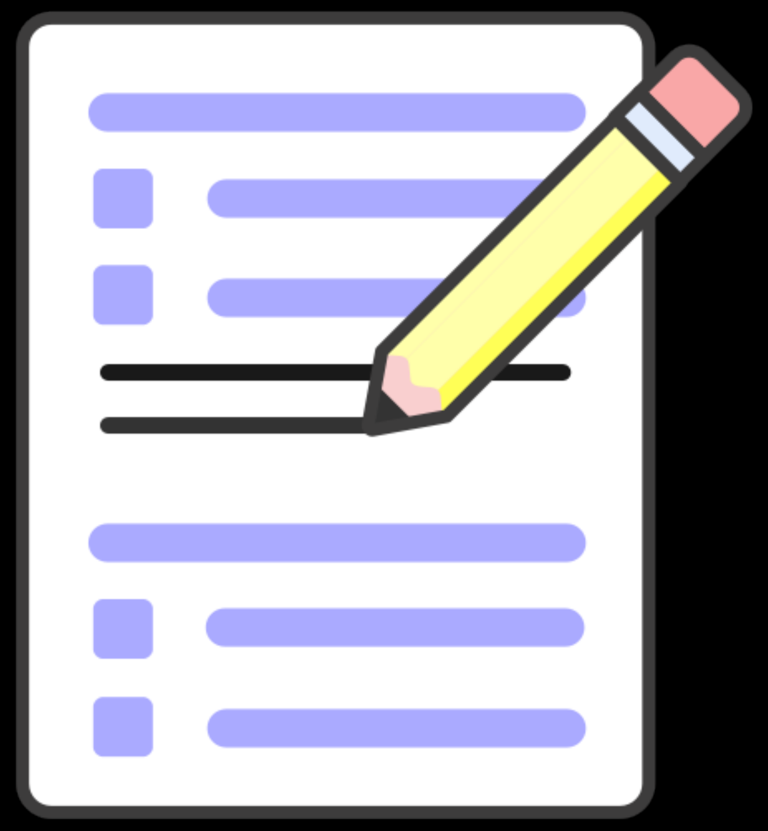
(possibly the most important concept in machine learning...)

Model learns patterns from here

Course materials
(training set)

Practice exam
(validation set)

Tune model patterns



Final exam
(test set)

See if the model is ready for the wild

Generalization

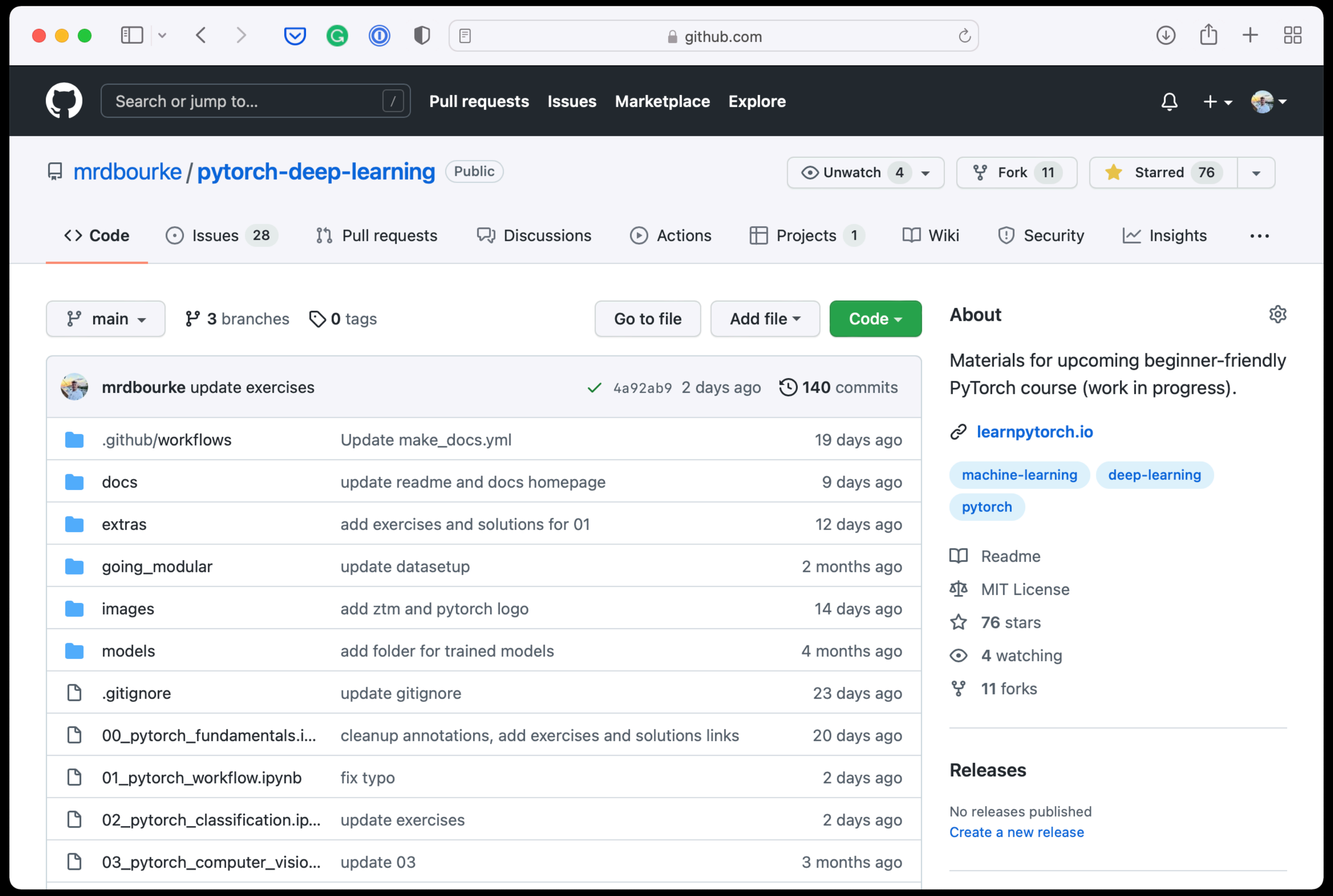
The ability for a machine learning model to perform well on data it hasn't seen before.

1. Initialise with random weights (only at beginning)
2. Show examples
Coat
Ankle boot
Shirt
Sneaker
[[0.092, 0.210, 0.415],
[0.778, 0.929, 0.030],
[0.019, 0.182, 0.555],
...,
Coat
Ankle boot
Shirt
Sneaker
[[116, 78, 15],
[117, 43, 96],
[125, 87, 23],
...,
[[0.983, 0.004, 0.013],
[0.110, 0.889, 0.001],
[0.023, 0.027, 0.985],
...,
Coat,
Ankle boot,
Shirt,
Sandal
3. Update representation outputs (weights & biases)
4. Repeat with more examples
Inputs
Numerical encoding
Learns representation (patterns/features/weights)
Representation outputs
Outputs

Resources

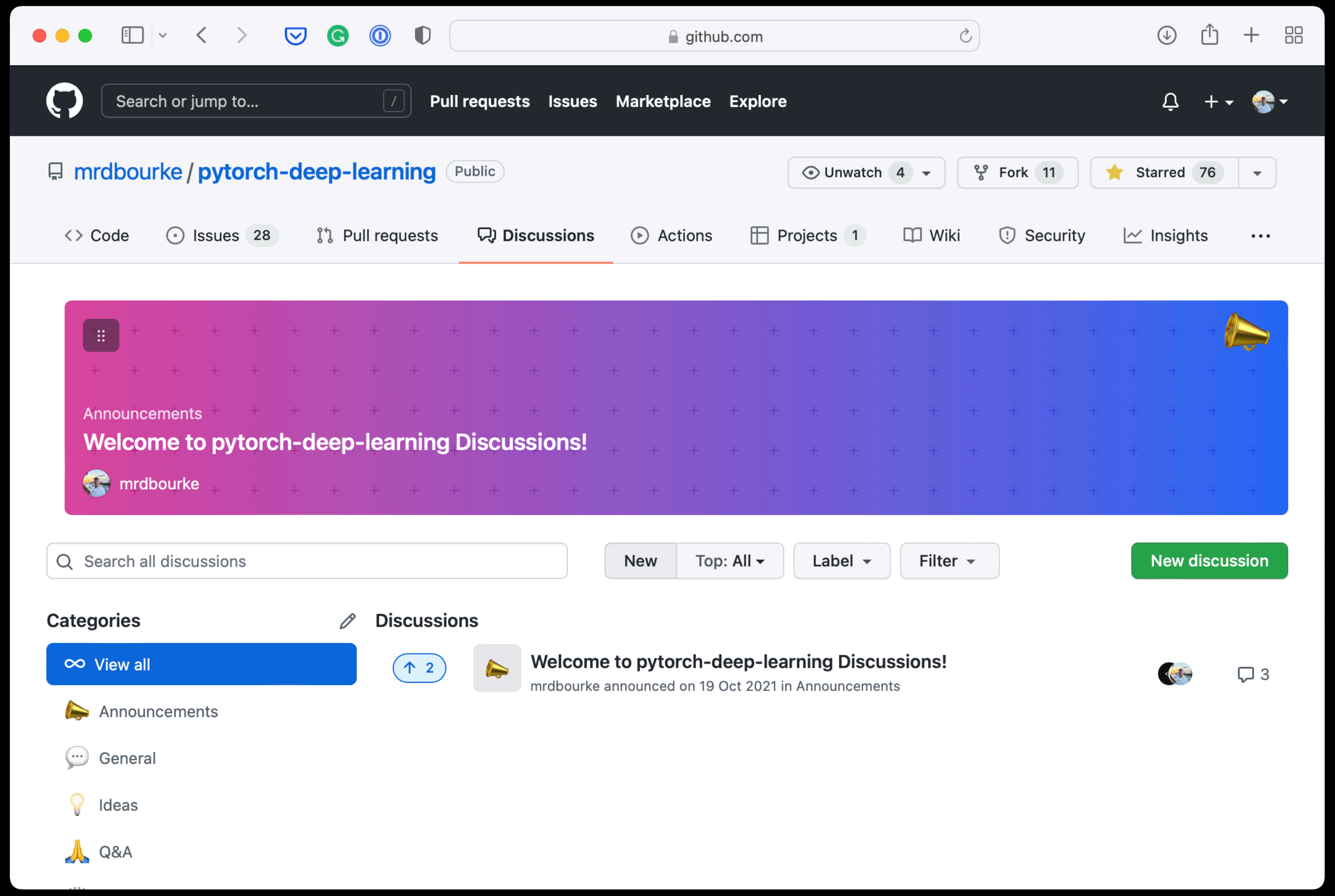
This course

Course materials



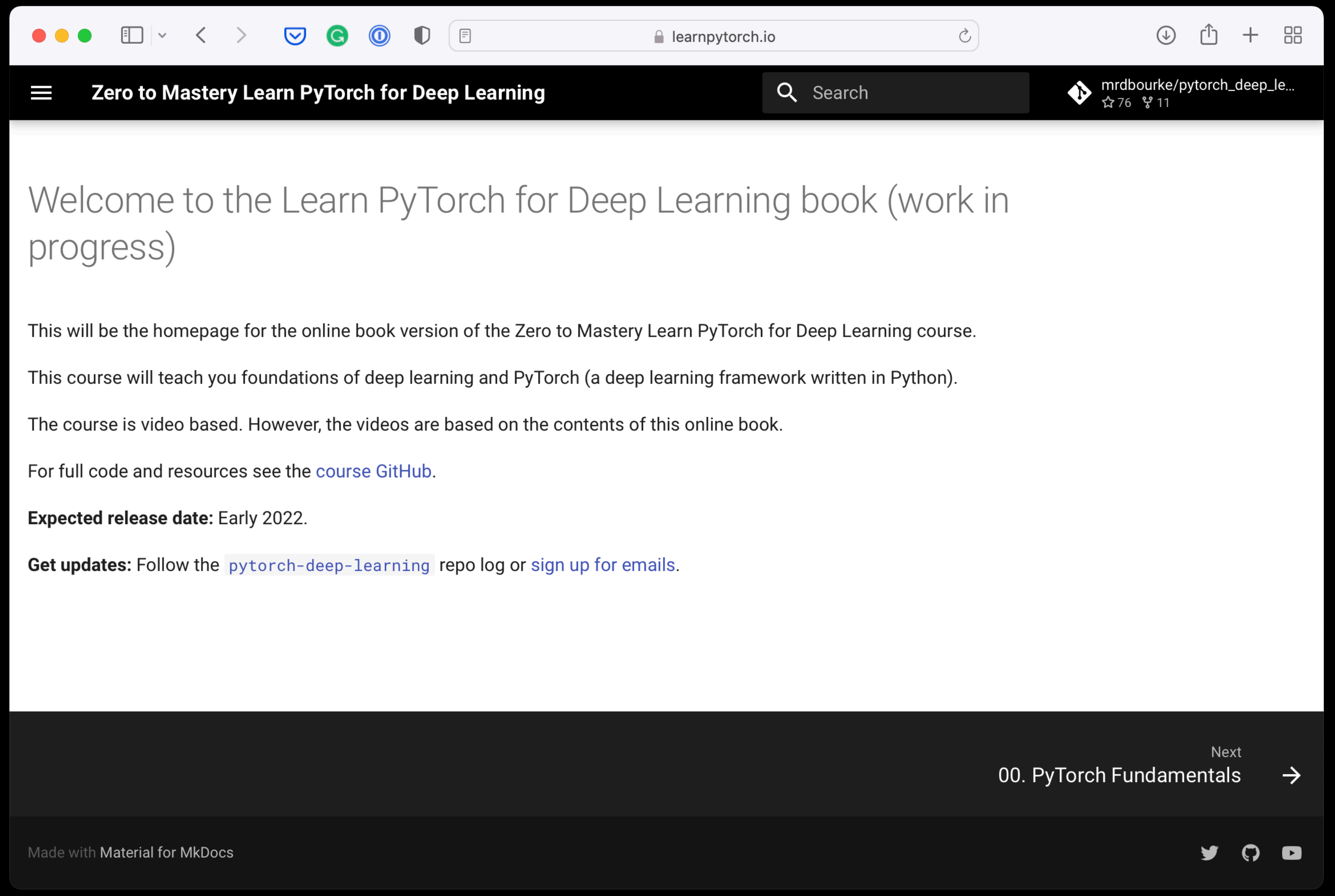
https://www.github.com/mrdbourke/pytorch-deep-learning

Course Q&A



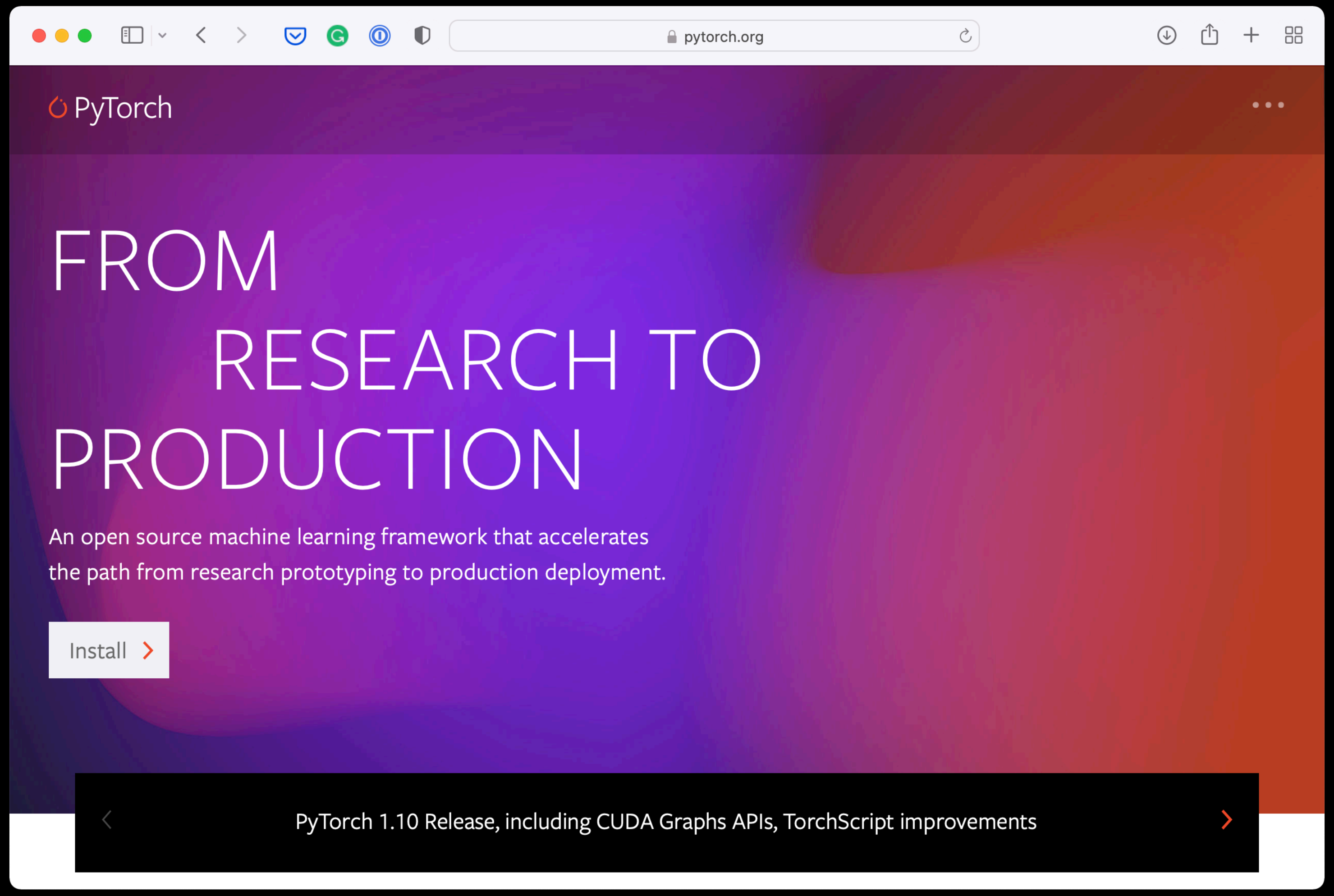
https://www.github.com/mrdbourke/pytorch-deep-learning/discussions

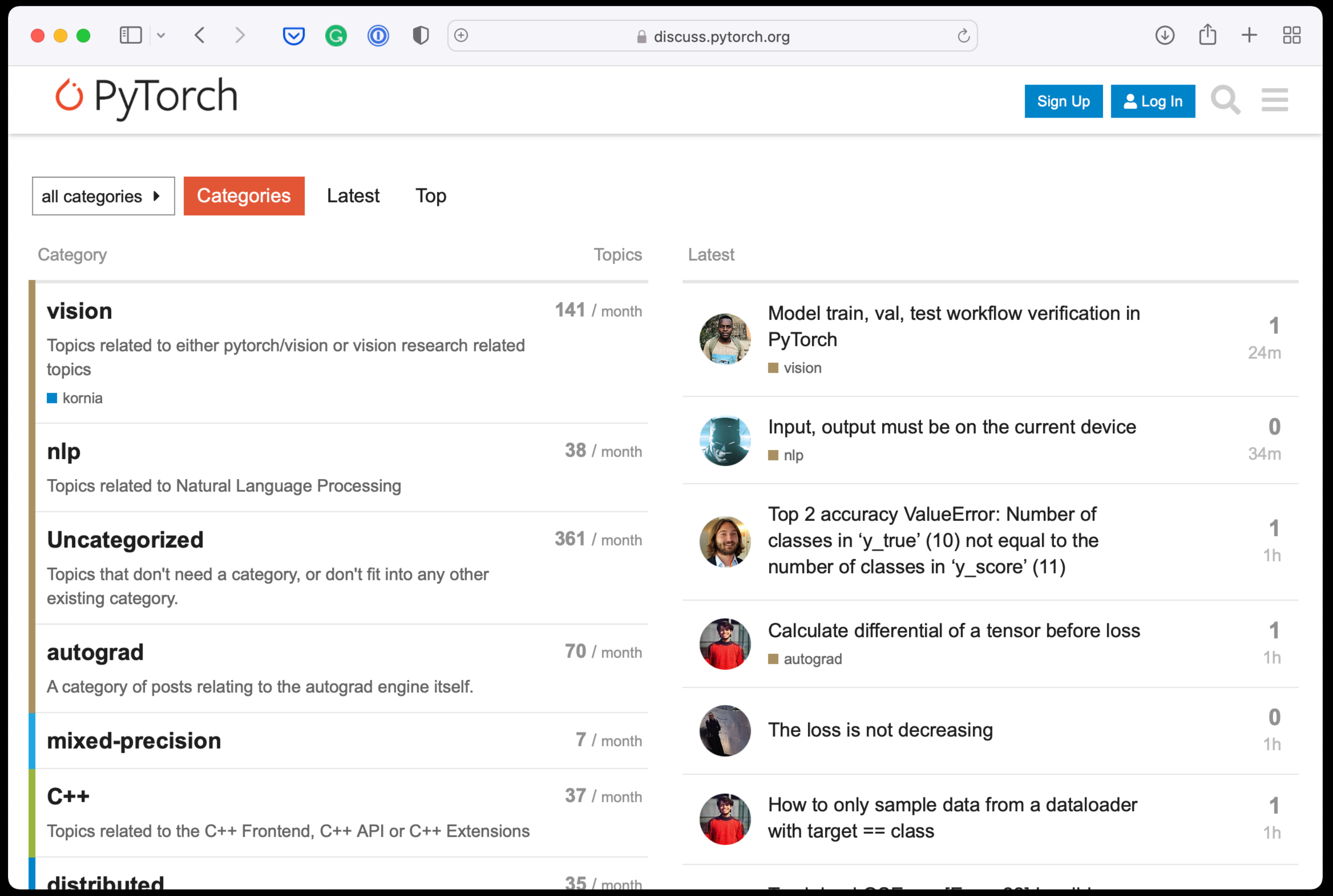
Course online book



https://learnpytorch.io

PyTorch website & forums





All things PyTorch