**Pytorch for deep learning and machine learning – freeCodeCamp – 10/18/2024**

[**https://pytorch.org/docs/stable/index.html**](https://pytorch.org/docs/stable/index.html)

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

* The computer does this part. How? Code and maths, we are going to be writing the code.

**What is machine learning?**

**Good comment by Yashaswi Kulshreshtha : I think you can use ML for literally as long as you can convert it into numbers and program it to find patterns. Literally it could be anything, any input or output from the universe.**

Why use machine learning (or deep learning)?

* Good reason: Why not?
* Better reason: For a complex problem, can you think of all the rules?

“If you can build a simple rule-based system that does not require machine learning, do that”.

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 adapts (‘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 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 models 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**

In machine learning, algorithm gradients boosted machines using XGBoost in structured data.

In unstructured data, algorithms are used in deep learning models.

What is a tensor?

A torch.Tensor is a multi-dimensional matrix containing elements of a single data type.

[**https://www.learnpytorch.io/00\_pytorch\_fundamentals/#exercises**](https://www.learnpytorch.io/00_pytorch_fundamentals/#exercises)

Where can you get help?

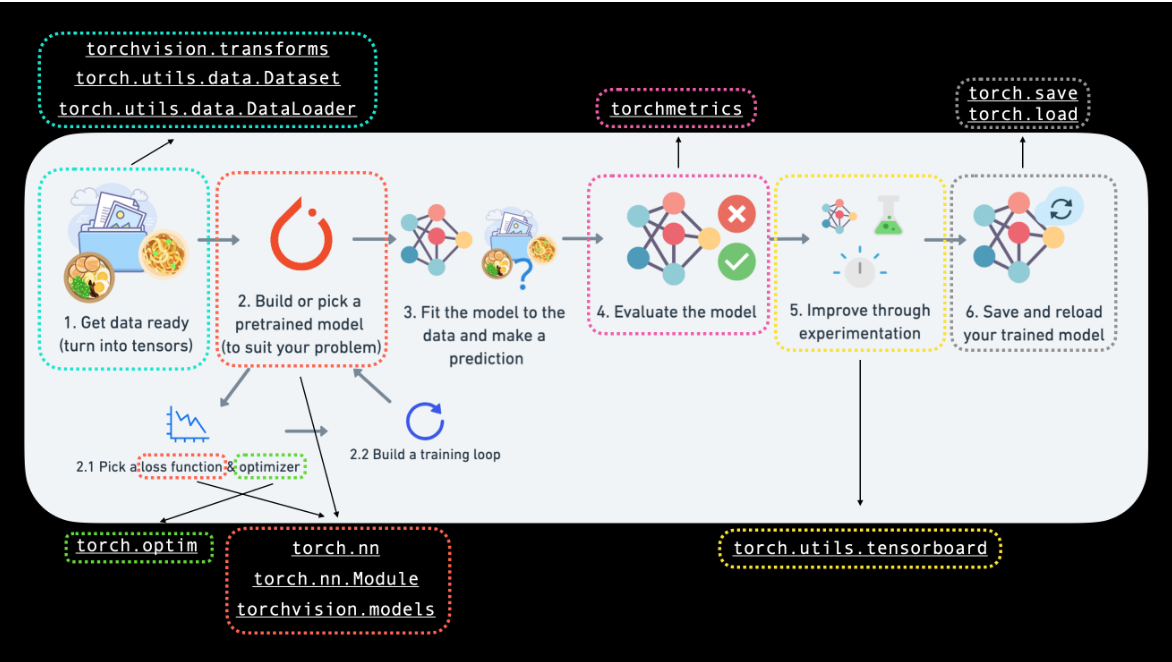
1. Follow along with the code
2. Try it for yourself
3. Press SHIFT+CMD+SPACE to read the docstring
4. Search for it
5. Try again
6. Ask

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

<https://www.learnpytorch.io/03_pytorch_computer_vision/>

1. Getting a vision dataset to work with using torchvision.datasets
2. Architecture of a convolution neural network (CNN) with pyTorch
3. An end-to-end multi-class image classification problem
4. Steps in modelling with CNNs in Pytorch
   1. Creating a CNN model with CNNs in Pytorch
   2. Picking a loss and optimizer
   3. Training a model
   4. Evaluating a model

What is a convolutional neural network (CNN)?



| **Topic** | **Contents** |
| --- | --- |
| Computer vision libraries in Pytorch | Pytorch has a bunch of built-in helpful computer vision libraries, let’s check them out. |
| Load data | To practise computer vision, we’ll start with some images of different pieces of clothing from FashioMNIST. |
| Prepare Data | We have got some images, let’s load them in with a PyTorch DataLoader so we can use them with our training loop. |
| Model 0: Building a baseline model | Create a multi-class classification model to learn patterns in the data, we’ll also choose a loss function, optimizer, and build a training loop. |
| Making predictions and evaluating model 0 | Let’s make some predictions with our baseline model and evaluate them. |
| Setup device agnostic code for future models | It’s best practice to write device-agnostic code, so let’s set it up. |
| Model 1: Adding non-linearity | Experimenting is a large part of machine learning, let’s try and improve upon our baseline model by adding non-linear layers. |
| Model 2: Convolutional Neural Network (CNN) | Time to get computer vision specific and introduce the powerful convolutional neural network architecture |
| Comparing our models | We’ve built three different models, let’s compare them. |
| Evaluating our best model | Let’s make some predictions on random images and evaluate our best model. |
| Making a confusion matrix | A confusion matrix is a great way to evaluate a classification model, let’s see how we can make one. |
| Saving and loading the best performing model | Since we might want to use our model for later, let’s save it and make sure it loads back in correctly. |

| **Pytorch Module** | What does it do? |
| --- | --- |
| torchvision | Contains datasets, model architectures and image transformations often used for computer vision problems. |
| torchvision.datasets | Here you’ll find many example computer vision datasets for a range of problems from image classification, object detection, image captioning, video classification and more. It also contains a series of base classes for making custom datasets. |
| torchvision.models | This module contains well-performing and commonly used computer vision model architectures implemented in PyTorch, you can use these with your own problems |
| torchvision.transforms | Often images need to be transformed (turned into numbers/processed/ augmented) before being used with a model, common image transformations are found here. |
| torch.utils.data.Dataset | Base dataset class for PyTorch |
| Torch.utils.data.DataLoader | Creates a Python iterable over a dataset(created with torch.utils.data.Dataset) |

[**https://poloclub.github.io/cnn-explainer/**](https://poloclub.github.io/cnn-explainer/)

What is data augmentation?

Looking at the same image but from a different perspective. To artificially increase the diversity of a dataset.

How to deal with overfitting?

<https://pytorch.org/docs/stable/index.html>

| Method to improve a model(Reduce Overfitting) | What does it do? |
| --- | --- |
| Get more data | Gives a model more of a chance to learn patterns between samples (e.g., if a model is performing poorly on images of pizza, show it more images of pizza. |
| Data augmentation | Increase the diversity of your training dataset without collecting more data. Increased diversity forces a model to learn more generalisation patterns. |
| Better data | Not all data samples are created equally. Removing poor samples from or adding better samples to your dataset can improve your model’s performance. |
| User transfer learning | Take a model’s pre-learned patterns from one problem and tweak to suit your own problem. For example, take a model trained on pictures of cars to recognise pictures of trucks. |
| Simplify your model | If the current model is already overfitting the training data, it may be too complicated of a model. This means it’s learning the patterns of the data too well and is not able to generalise well to unseen data.One way to simplify a model is to reduce the number of layers it uses or to reduce the number of hidden units in each layer. |
| Use learning rate decay | The idea here is to slowly decrease the learning rate as a model trains. |
| Use early stopping | Early stopping stops model training “before” it begins to overfit. As in, say the model’s loss has stopped decreasing for the past 10 epochs, you may want to stop the model training here and go with the model weights that have the lowest loss. |
| Add more layers/units to your model | If your model is underfitting, it may not have enough capability to learn the required patterns/weights/representations of the data to be productive. One way to add more predictive power to your model is to increase the number of hidden layers/units within those layers. |
| Tweak the learning rate | Perhaps your model’s learning rate is too high to begin with. And it’s trying to update it’s weights each epoch too much, in turn not learning anything. In this case, you might lower the learning rate and see what happens. |
| Train for longer | Sometimes a model just needs more time to learn representations of data. If you find in your smaller experiments your model is not learning anything. Perhaps leaving it train for a more epochs may result in better performance |
| Use transfer learning | Take a model’s pre-learned from one problem and tweak them to suit your own problem |
| Use less regularisation | Perhaps your model is underfitting because you are trying to prevent overfitting too much. Holding back on regularisation techniques can help your model fit the data better. |