

Custom datasets with



Where can you get help?

- Follow along with the code



In the last notebook, [notebook 03](#), we looked at how to build computer vision models on an in-built dataset in PyTorch (FashionMNIST). The steps we took are similar across many different problems in machine learning.

Find a dataset, turn the dataset into numbers, build a model (or find an existing model) to find patterns in those numbers that can be used for prediction.

PyTorch has many built-in datasets used for a wide number of machine learning benchmarks, however, you'll often want to use your own custom dataset.

What is a custom dataset?

A custom dataset is a collection of data relating to a specific problem you're working on. In essence, a **custom dataset** can be comprised of almost anything.

For example, if we were building a food image classification app like [Nutrifit](#), our custom dataset might be images of food.

Or if we were trying to build a model to classify whether or not a text-based review on a website was positive or negative, our custom dataset might be examples of existing customer reviews and their ratings.

Or if we were trying to build a sound classification app, our custom dataset might be sound samples alongside their sample labels.

Or if we were trying to build a recommendation system for customers purchasing things on our website, our custom dataset might be examples of products other people have bought.

PyTorch Domain Libraries

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

"Are these reviews positive or negative?"

TorchVision

"If in doubt, run the code"

- Try it for yourself

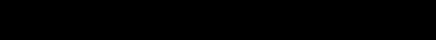


```
(transforms) -> None
One of the ways we can do this is by using the torchvision.transforms module.
torchvision.transforms contains many pre-built methods for formatting images, turning them into tensors and even manipulating them for data augmentation (the practice of altering data to make it more useful).
To get experience with torchvision.transforms, let's write some code to see how we can use it.
1. Resize the images using transforms.Resize.
2. Flip our images randomly on the horizontal using transforms.RandomHorizontalFlip.
3. Turn our images from a PIL image to a PyTorch tensor using transforms.ToTensor.
We can compile all of these steps using transforms.Compose.
```

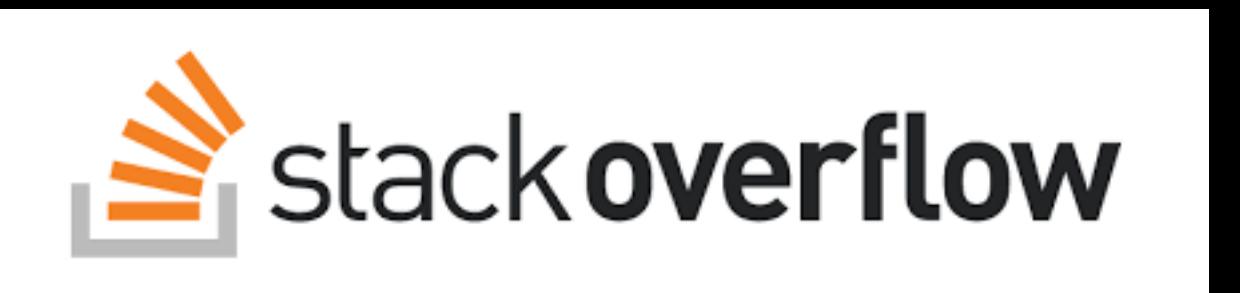
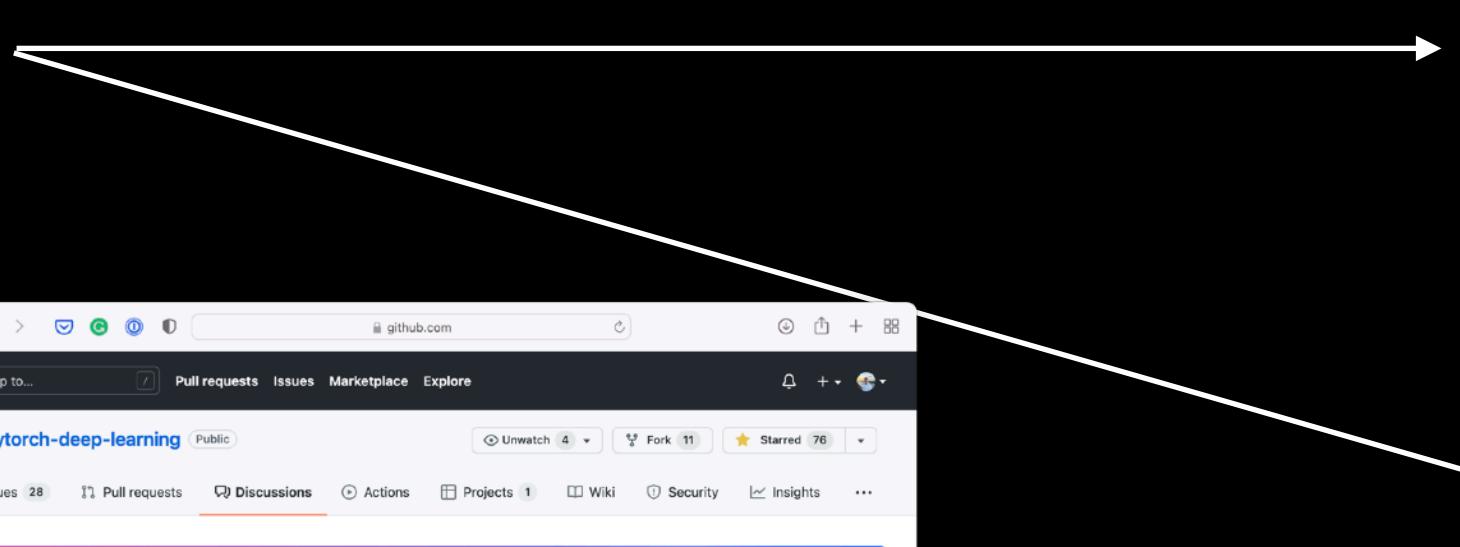
```
>>> transforms.Compose([
    >>> transforms.CenterCrop(256)
])
transforms.Compose([CenterCrop(256)])
```

Now we've got a composition of transforms, let's write a function to try them out on various images.

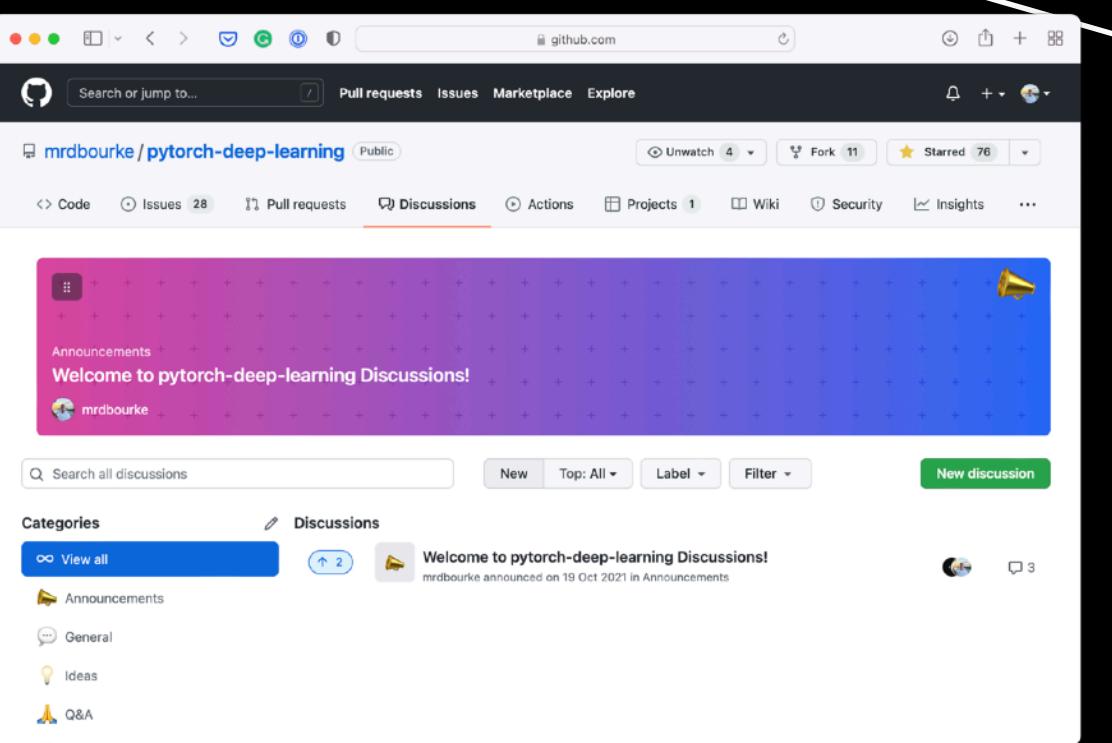
- Press SHIFT + CMD + SPACE to read the docstring



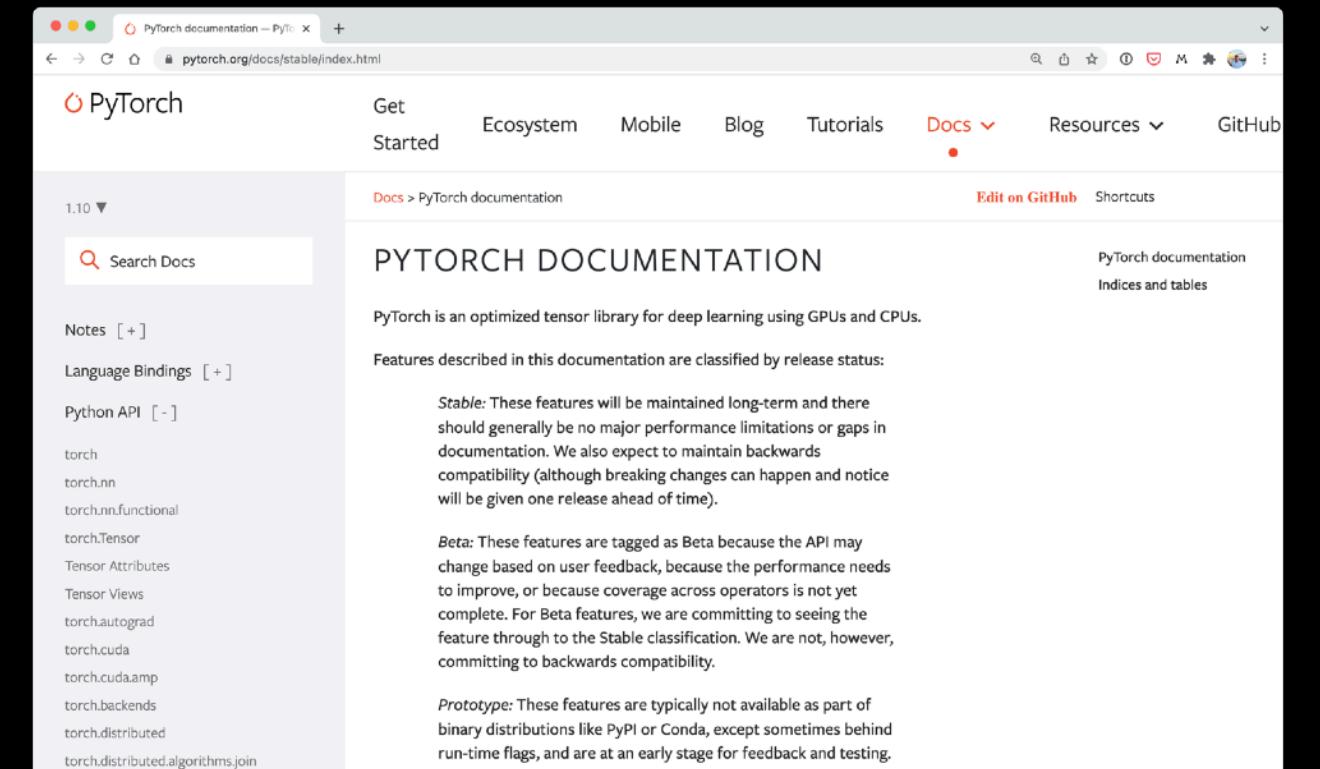
- Search for it



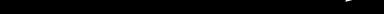
- Try again



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



- Ask



“What is a custom dataset?”

“I’ve got my own dataset, can I build a model with PyTorch to predict on it?”

Yes.

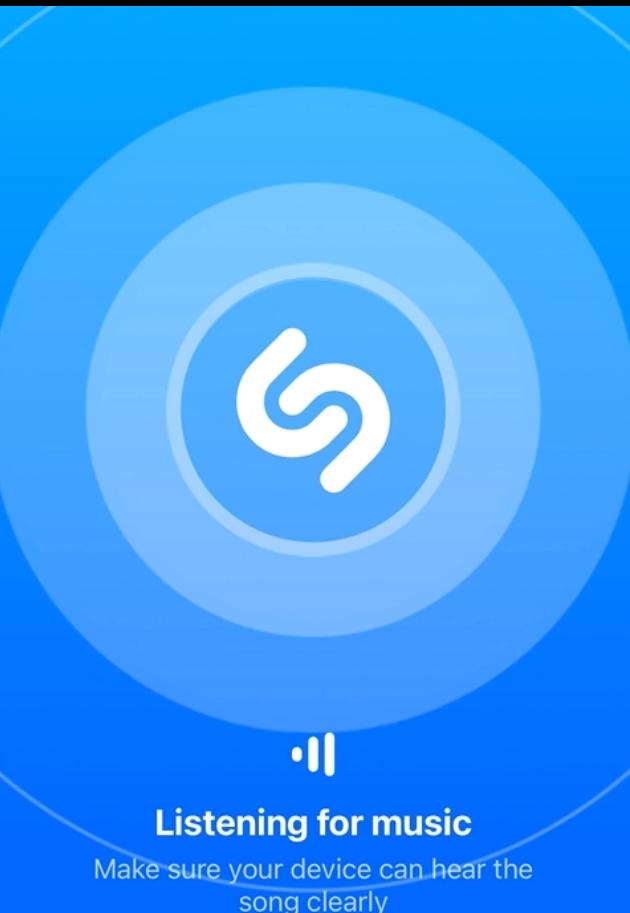
PyTorch Domain Libraries

“Is this a photo of pizza, steak or sushi?”



TorchVision

“What song is playing?”



Different domain
libraries contain data
loading functions for
different data sources

TorchAudio

“Are these reviews positive or negative?”

 martinhk Today at 9:20 PM
@mrdbourke Just started your tensorflow course a few days ago! I took quite a few ML/DL courses online and I would like to say it's by far the best deep learning course I have ever had! I like your way of teaching difficult topics and I learnt a lot more coding along with you!



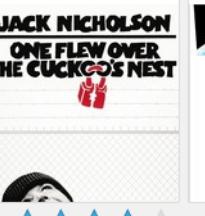
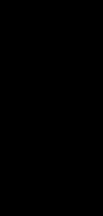
 iwatts Yesterday at 10:29 PM
Thanks to ZTM I've landed a job - in the UK working for a Global Marketing company as a Senior Analytics Developer. Thank you @Andrei Neagoie and @mrdbourke Couldn't have got there without you.



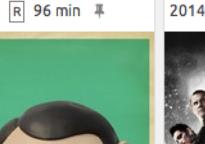
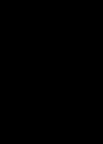
TorchText

“How do we recommend similar products?”

top picks see more
based on your ratings, MovieLens recommends these movies

Band of Brothers	Casablanca	One Flew Over the Cuckoo's Nest	The Lives of Others	Sunset Boulevard	The Third Man	Path
						
2001 [R] 705 min	1942 [PG] 102 min	1975 [R] 133 min	2006 [R] 137 min	1950 [NR] 110 min	1949 [NR] 104 min	1957

recent releases see more
movies released in last 90 days that you haven't rated

Cantinflas	Felony	What If	Frank	Sin City: A Dame to Kill For	If I Stay	Are We Next
						
2014 [PG] 105 min	2014	2014 [PG-13] 102 min	2014 [R] 96 min	2014 [R] 102 min	2014 [PG-13] 106 min	2014

TorchRec

Source: movielens.org

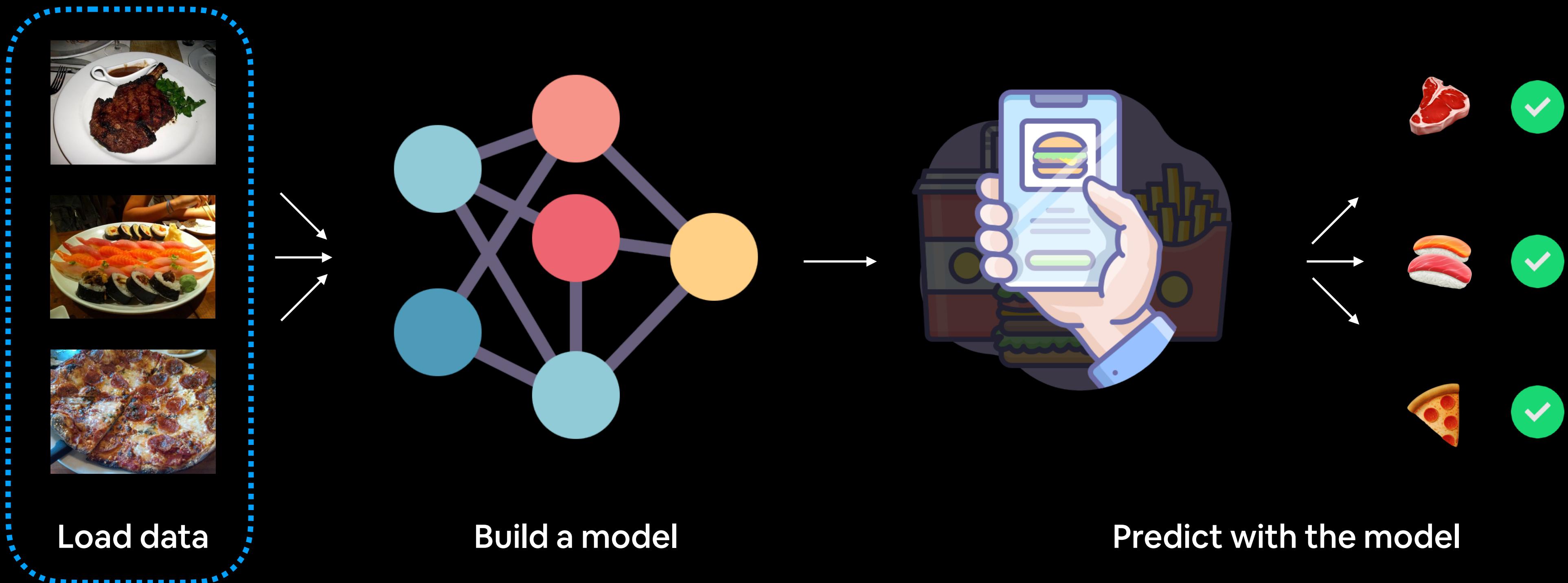
PyTorch Domain Libraries

Problem Space	Pre-built Datasets and Functions
Vision	<code>torchvision.datasets</code>
Text	<code>torchtext.datasets</code>
Audio	<code>torchaudio.datasets</code>
Recommendation system	<code>torchrec.datasets</code>
Bonus	<code>TorchData*</code>

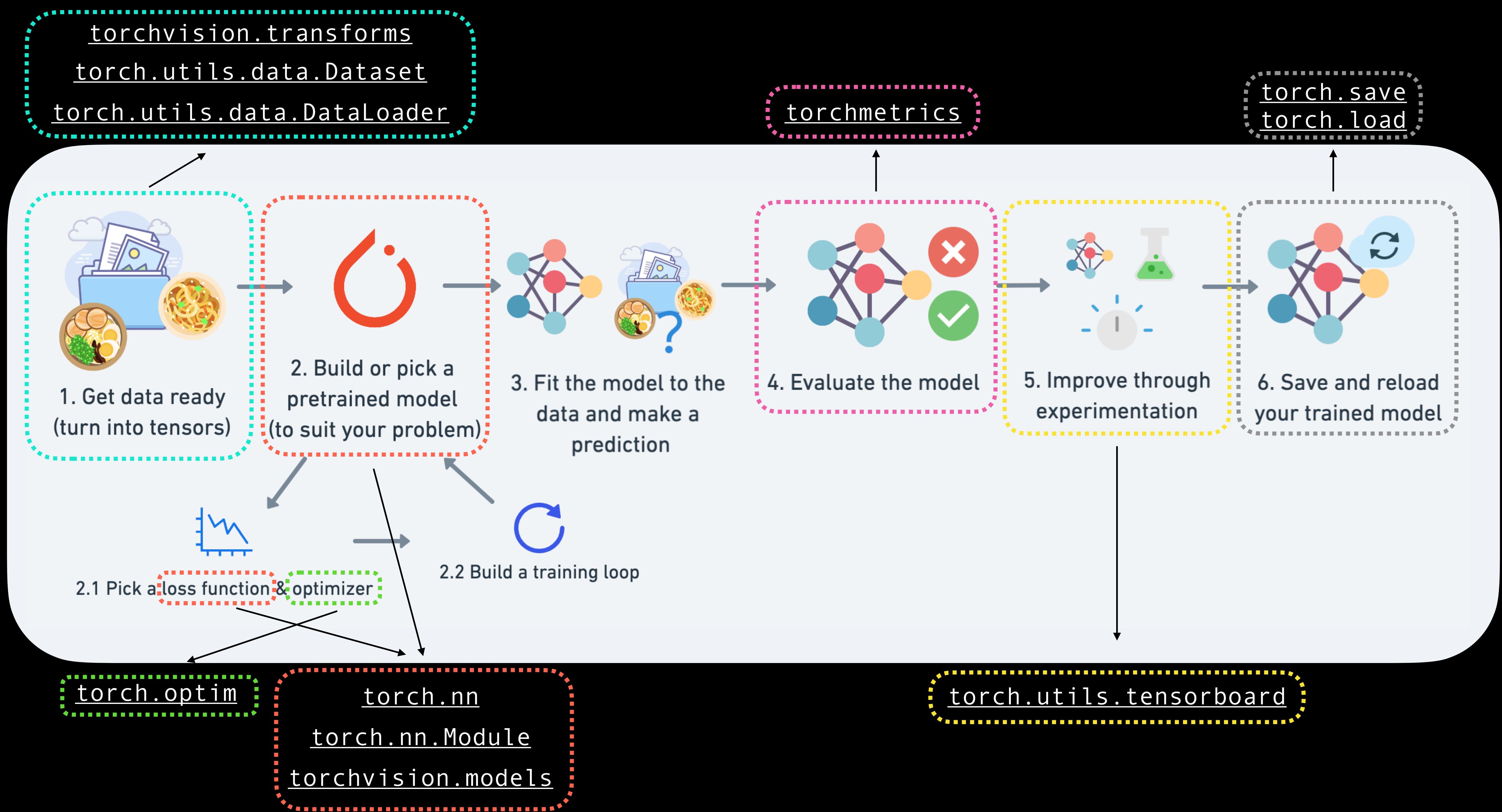
*TorchData contains many different helper functions for loading data and is currently in beta as of April 2022.

What we're going to build

FoodVision Mini



We're going to write code to load images of food (our own custom dataset for Foodvision Mini)



What we're going to cover

(broadly)

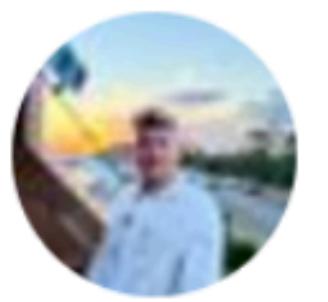
- Getting a **custom dataset** with PyTorch
- **Becoming one with the data** (preparing and visualising)
- **Transforming data** for use with a model
- **Loading custom data** with pre-built functions and custom functions
- Building **FoodVision Mini** to classify 🍕🥩🍣 images
- Comparing models with and without **data augmentation**
- **Making predictions** on custom data

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

How:



Let's code!



Daniel Bourke
@mrdbourke

...

“If I had 8 hours to build a machine learning model, I’d spend the first 6 hours preparing my dataset.”

- Abraham Lossfunction

12:35 PM · Nov 4, 2021 · Twitter Web App

Source: [@mrdbourke Twitter](#)

Standard image classification data format

Your own data format
will depend on what
you're working

```
pizza_steak_sushi/ # <- overall dataset folder
  train/ # <- training images
    pizza/ # <- class name as folder name
      image01.jpeg
      image02.jpeg
      ...
    steak/
      image24.jpeg
      image25.jpeg
      ...
    sushi/
      image37.jpeg
      ...
  test/ # <- testing images
    pizza/
      image101.jpeg
      image102.jpeg
      ...
    steak/
      image154.jpeg
      image155.jpeg
      ...
    sushi/
      image167.jpeg
      ...
```

The premise remains:
write code to get your
data into tensors for
use with PyTorch

What is data augmentation?

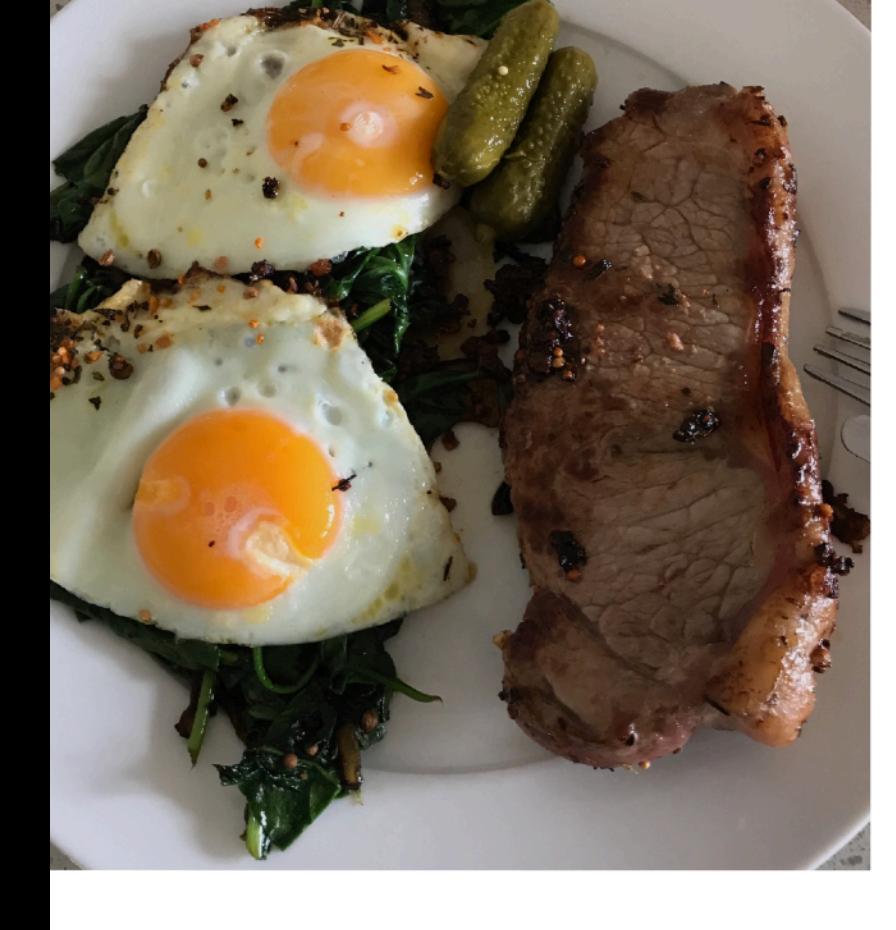
Looking at the same image but from different perspective(s)*. To artificially increase the diversity of a dataset.



Original



Rotate



Shift



Zoom

*Note: There are many more different kinds of data augmentation such as, cropping, replacing, shearing. This slide only demonstrates a few.

PyTorch State of the Art Recipe

Research comes out often on how best to train models, state-of-the-art (SOTA) methods are always changing).

The screenshot shows a web browser window displaying a PyTorch blog post. The title of the post is "How to Train State-Of-The-Art Models Using TorchVision's Latest Primitives". The author is listed as "by Vasilis Vryniotis". The text discusses the release of TorchVision v0.11, which includes numerous new primitives, models, and training recipe improvements, allowing for state-of-the-art (SOTA) results. The post aims to enable researchers to reproduce papers and conduct research more easily by using common building blocks. It also provides tools for Applied ML practitioners to train their models on their own data using the same SOTA techniques as in research. The final goal is to refresh pre-trained weights and offer better off-the-shelf models to users.

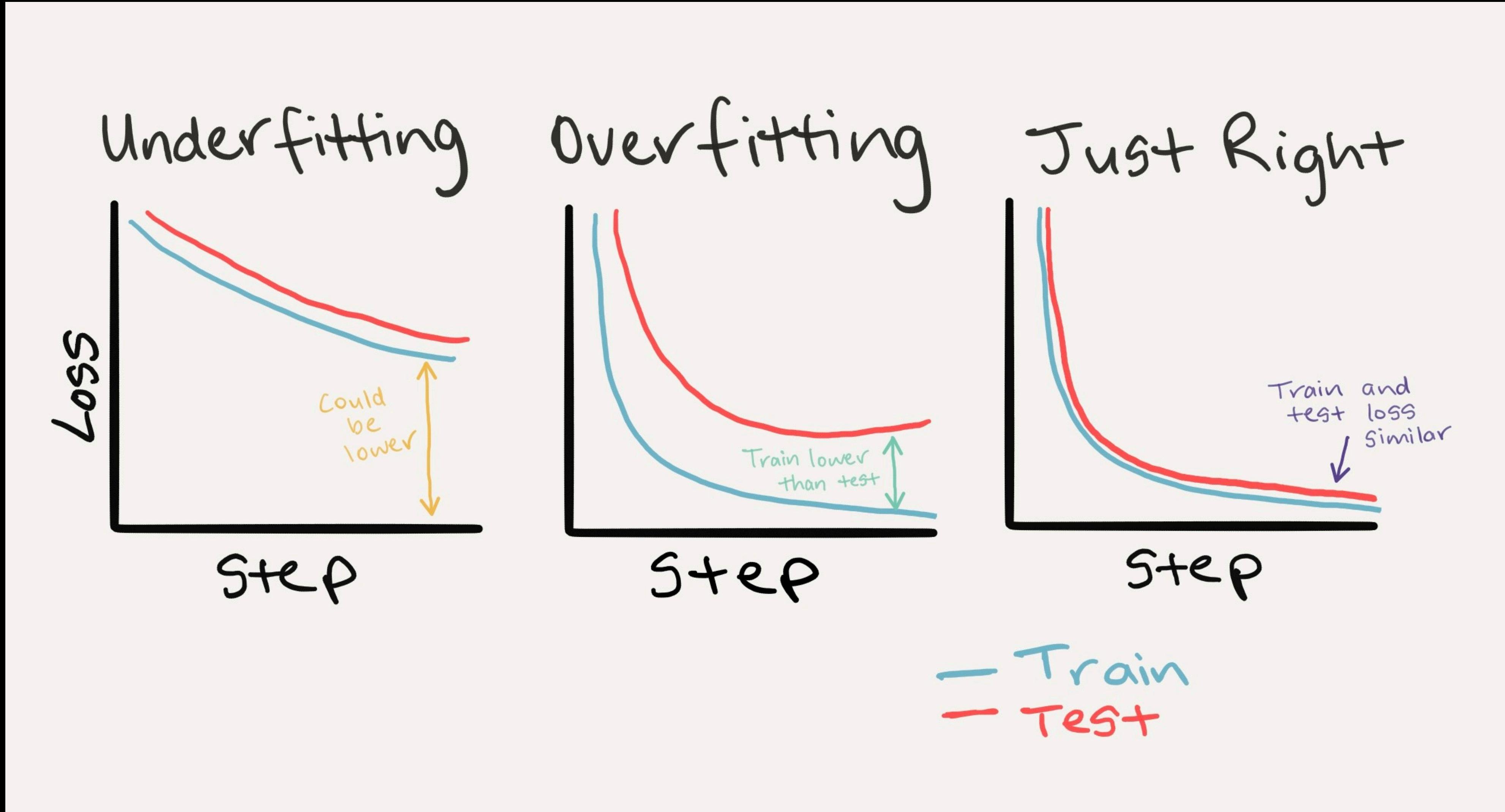
A few weeks ago, TorchVision v0.11 was released packed with numerous new primitives, models and training recipe improvements which allowed achieving state-of-the-art (SOTA) results. The project was dubbed “[TorchVision with Batteries Included](#)” and aimed to modernize our library. We wanted to enable researchers to reproduce papers and conduct research more easily by using common building blocks. Moreover, we aspired to provide the necessary tools to Applied ML practitioners to train their models on their own data using the same SOTA techniques as in research. Finally, we wanted to refresh our pre-trained weights and offer better off-the-shelf models to our users, hoping that they would build better applications.

Though there is still much work to be done, we wanted to share with you some exciting results from the above work. We will showcase how one can use the new tools included in TorchVision to achieve state-of-the-art results on a highly competitive and well-studied architecture such as ResNet50 [1]. We will share the exact recipe used to improve our baseline by over 4.7 accuracy points to reach a final top-1 accuracy of 80.9% and share the journey for deriving the new training process. Moreover, we will

Source: Training state-of-the-art computer vision models with `torchvision` from the [PyTorch blog](#).

LOSS curves

(a way to evaluate your model's performance over time)



*There are more combinations of these, to see them check out [Google's Interpreting Loss Curves guide](#).

Dealing with overfitting

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 (e.g. take your photos of pizza and randomly rotate them 30°). 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.

Use transfer learning

Take a model's pre-learned patterns from one problem and tweak them 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 isn't able to generalize 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. This is akin to reaching for a coin at the back of a couch. The closer you get, the smaller your steps. The same with the learning rate, the closer you get to convergence, the smaller you'll want your weight updates to be.

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 (this number is arbitrary), you may want to stop the model training here and go with the model weights that had the lowest loss (10 epochs prior).

Dealing with underfitting

Method to improve a model (reduce underfitting)

What does it do?

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 predictive. 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 its 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 isn't learning anything, perhaps leaving it train for a more epochs may result in better performance.

Use transfer learning

Take a model's pre-learned patterns from one problem and tweak them to suit your own problem. For example, take a model trained on pictures of cars to recognise pictures of trucks.

Use less regularization

Perhaps your model is underfitting because you're trying to prevent overfitting too much. Holding back on regularization techniques can help your model fit the data better.

Predicting on custom data

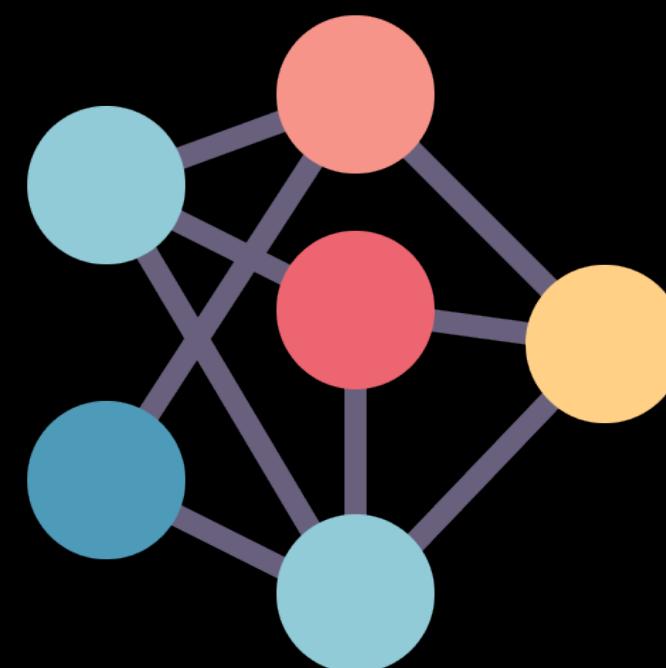
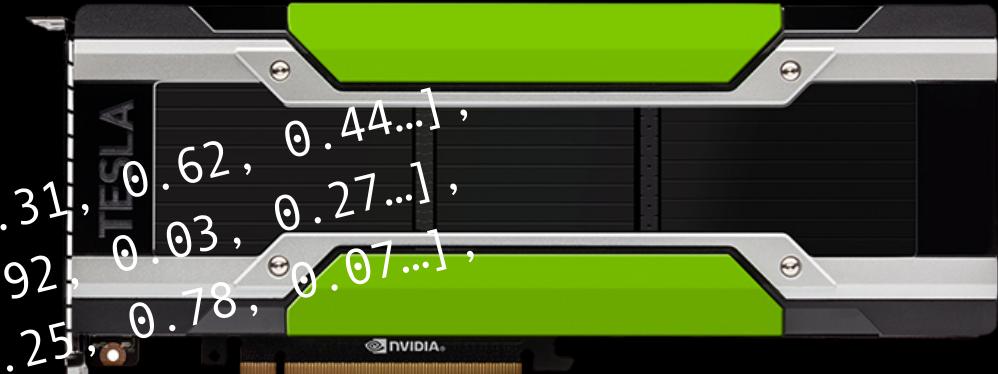
(3 things to make sure of...)

Custom image



→
[[0.31, 0.62, 0.44...],
[0.92, 0.03, 0.27...],
[0.25, 0.78, 0.07...],
...,
torch.float32

Is the model on the GPU?



Original

Shape = [64, 64, 3]

Add batch dimension & rearrange if needed

Shape = [None, 64, 64, 3] (NHWC)
Shape = [None, 3, 64, 64] (NCHW)

Same as model input

1. Data in right datatype

2. Data on same device as model

3. Data in correct shape