Autumn 2023
Week 3

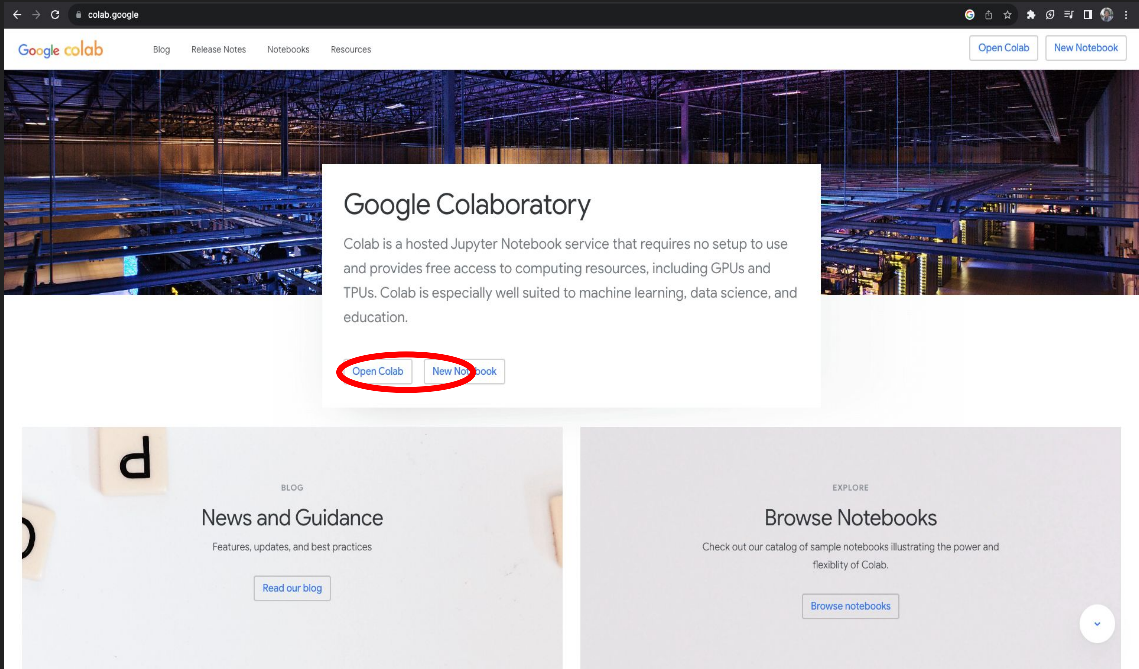
go.osu.edu/aiclubsignup

Plan for today

- We’re going to explain how to find a dataset
- How to import that dataset into your Google Colab environment
- How to use the built-in datasets functionality of PyTorch
- How to transform your data so that you can feed it into a model
- How to feed a deep learning model the data

Before we start…

- Open up Google Colab
- Select “New Notebook”
- Name the notebook whatever you want
- https://colab.research.google.com/



Google Colaboratory

Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education.

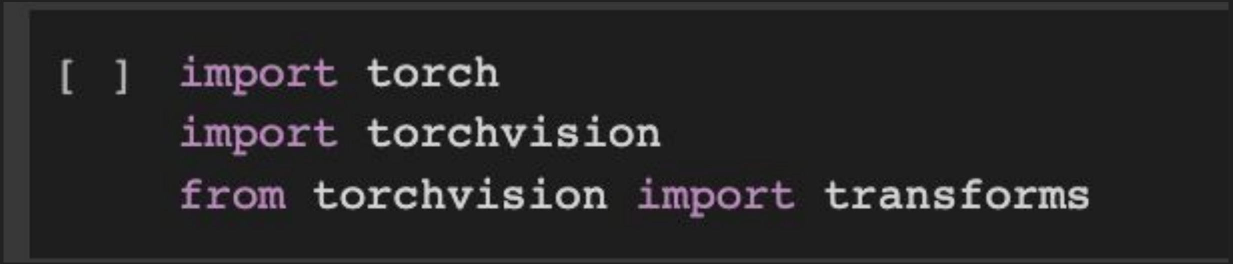
News and Guidance

Features, updates, and best practices

Browse Notebooks

Check out our catalog of sample notebooks illustrating the power and flexiblity of Colab.

Enter this into a code block

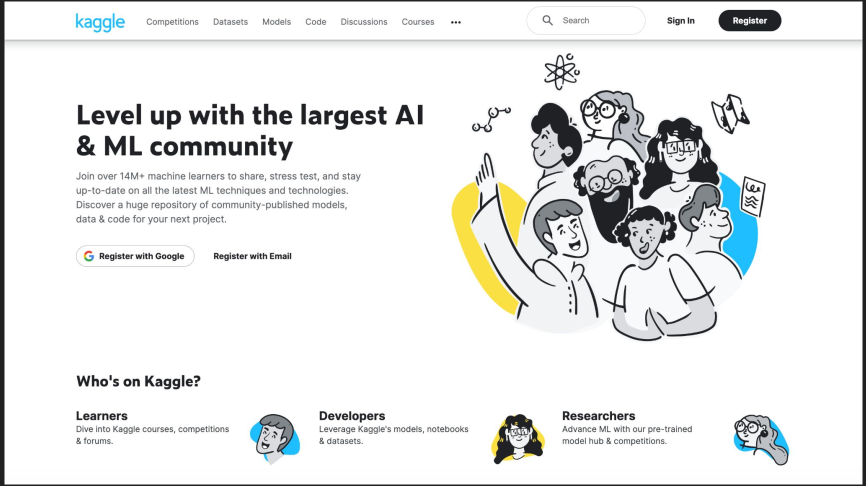


```
import torch
import torchvision
from torchvision import transforms
```

- This is installing libraries that we will need for today
    - PyTorch and torchvision (an accompanying library of PyTorch)

How to find a dataset for your project? (pt. 1)

- Kaggle is a popular machine learning community website that offers tons of datasets, pretrained models, tutorials, and etc.
- Here is the link: https://www.kaggle.com/
- If you don’t already have an account, you can make one for free!



Level up with the largest AI & ML community

Join over 14M+ machine learners to share, stress test, and stay up-to-date on all the latest ML techniques and technologies. Discover a huge repository of community-published models, data & code for your next project.

Who's on Kaggle?

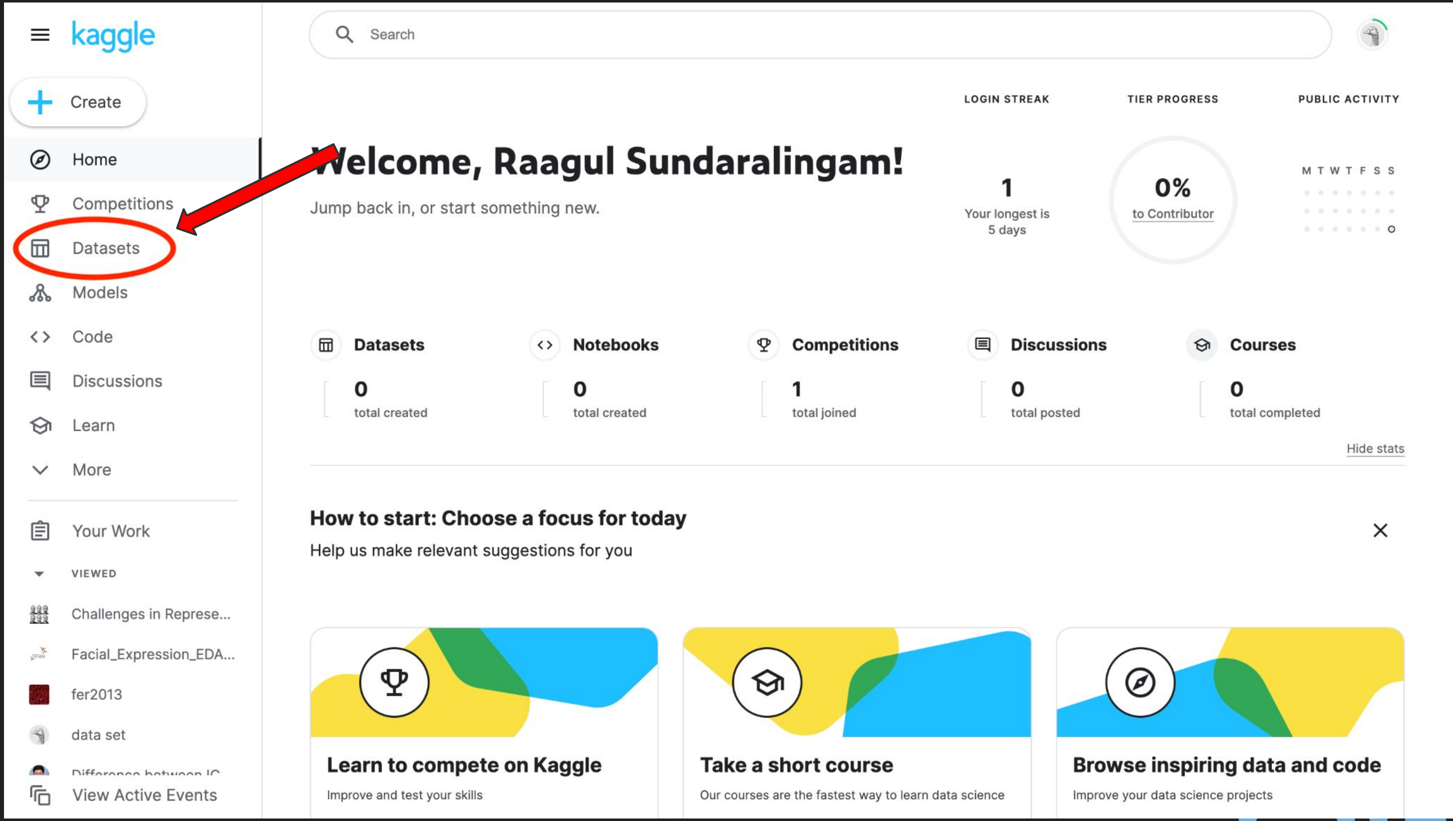
Learners
Dive into Kaggle courses, competitions & forums.

Developers
Leverage Kaggle's models, notebooks & datasets.

Researchers
Advance ML with our pre-trained model hub & competitions.

How to find a dataset for your project? (pt. 2)

Click on the “Datasets” option at the left hand side



Welcome, Raagul Sundaralingam!

Jump back in, or start something new.

1
Your longest is 5 days

0%
to Contributor

Datasets	Notebooks	Competitions	Discussions	Courses
0 total created	0 total created	1 total joined	0 total posted	0 total completed

How to start: Choose a focus for today

Help us make relevant suggestions for you

Learn to compete on Kaggle

Improve and test your skills

Take a short course

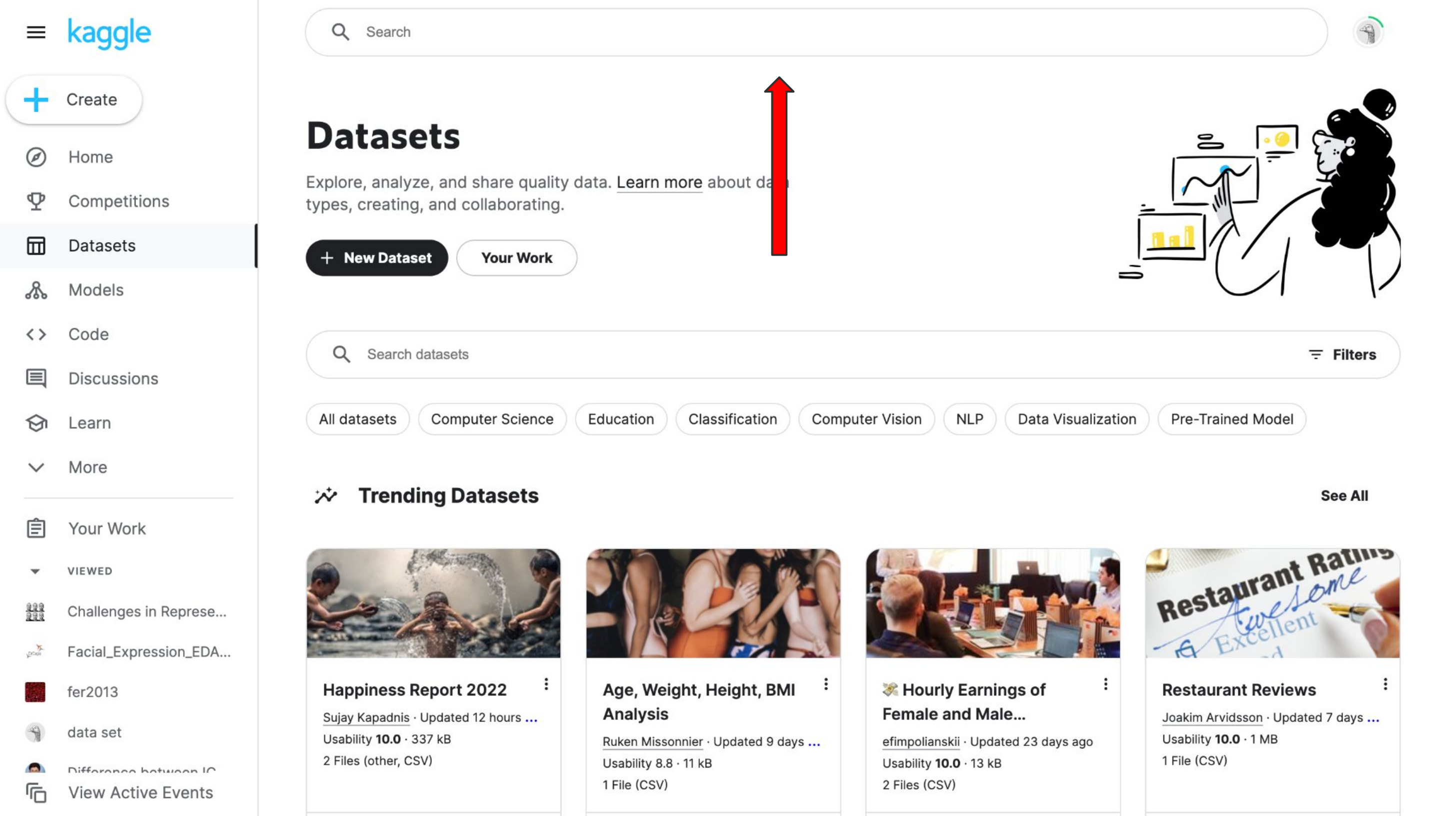
Our courses are the fastest way to learn data science

Browse inspiring data and code

Improve your data science projects

How to find a dataset for your project? (pt. 3)

Once you reach this screen, simply click on the search bar on the top and type in whatever topic you want data on (ex: housing prices)



Datasets

Explore, analyze, and share quality data. Learn more about data types, creating, and collaborating.

Trending Datasets

Happiness Report 2022
Sujay Kapadnis · Updated 12 hours ...
Usability **10.0** · 337 kB
2 Files (other, CSV)

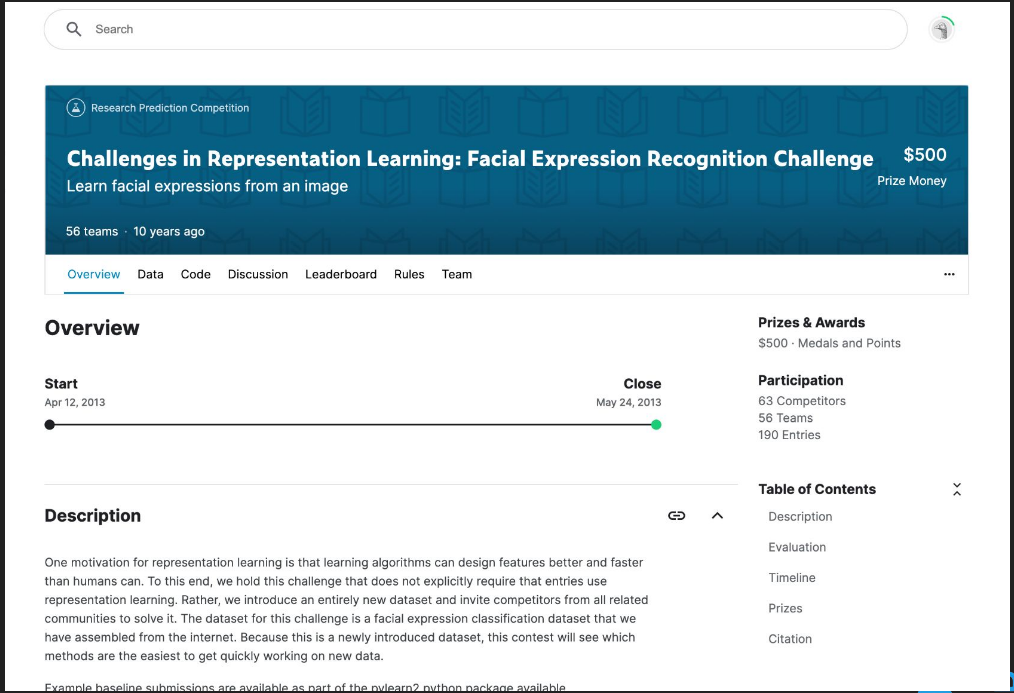
Age, Weight, Height, BMI Analysis
Ruken Missonnier · Updated 9 days ...
Usability 8.8 · 11 kB
1 File (CSV)

Hourly Earnings of Female and Male...
efimpolianskii · Updated 23 days ago
Usability **10.0** · 13 kB
2 Files (CSV)

Restaurant Reviews
Joakim Arvidsson · Updated 7 days ...
Usability **10.0** · 1 MB
1 File (CSV)

Dataset for emotion recognition

- For the emotion recognition project, we will be using the FER2013 dataset.
- There’s plenty of datasets out there for this project (a lot of them are better too), but we chose FER2013 because the PyTorch Datasets class already has support for FER2013. (**More on this later…**)
- Here is the link to the official FER2013 dataset on Kaggle: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge



Research Prediction Competition

Challenges in Representation Learning: Facial Expression Recognition Challenge

Learn facial expressions from an image

$500
Prize Money

56 teams · 10 years ago

Overview

Start
Apr 12, 2013

Close
May 24, 2013

Prizes & Awards
$500 · Medals and Points

Participation
63 Competitors
56 Teams
190 Entries

Table of Contents
Description
Evaluation
Timeline
Prizes
Citation

Description

One motivation for representation learning is that learning algorithms can design features better and faster than humans can. To this end, we hold this challenge that does not explicitly require that entries use representation learning. Rather, we introduce an entirely new dataset and invite competitors from all related communities to solve it. The dataset for this challenge is a facial expression classification dataset that we have assembled from the internet. Because this is a newly introduced dataset, this contest will see which methods are the easiest to get quickly working on new data.

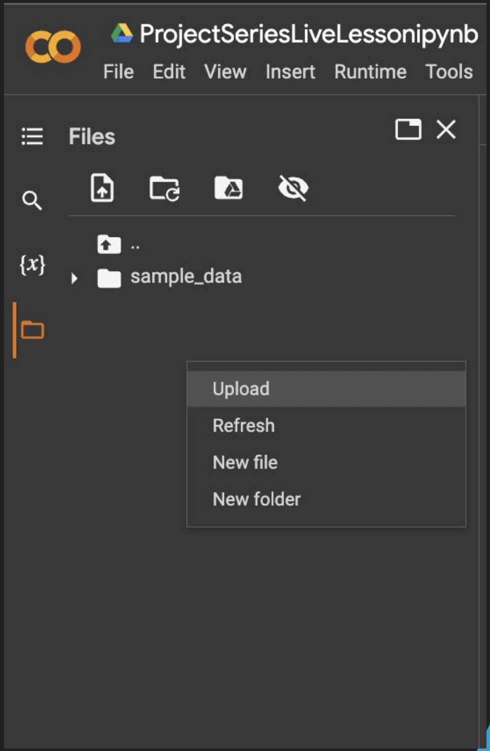
Example baseline submissions are available as part of the pylearn2 python package available

Downloading the FER2013 Dataset into Google Colab

- Click on the “Data” option under the banner of the FER2013 link
- Scroll down and click on the “Download All” option (should be in a black button)
- It should’ve downloaded as a zip file called “challenges-in-representation-learning-facial-expression-recognition-challenge.zip”

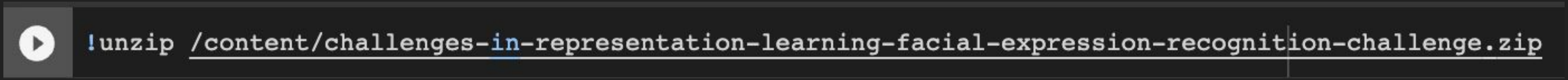
Downloading the FER2013 Dataset into Google Colab

- Click on the 📁 icon on the left side bar.
- Then right click in the file area and click "Upload" (as shown in the picture to the right)
- Then upload the zip file that we downloaded in the previous slide



Downloading the FER2013 Dataset into Google Colab

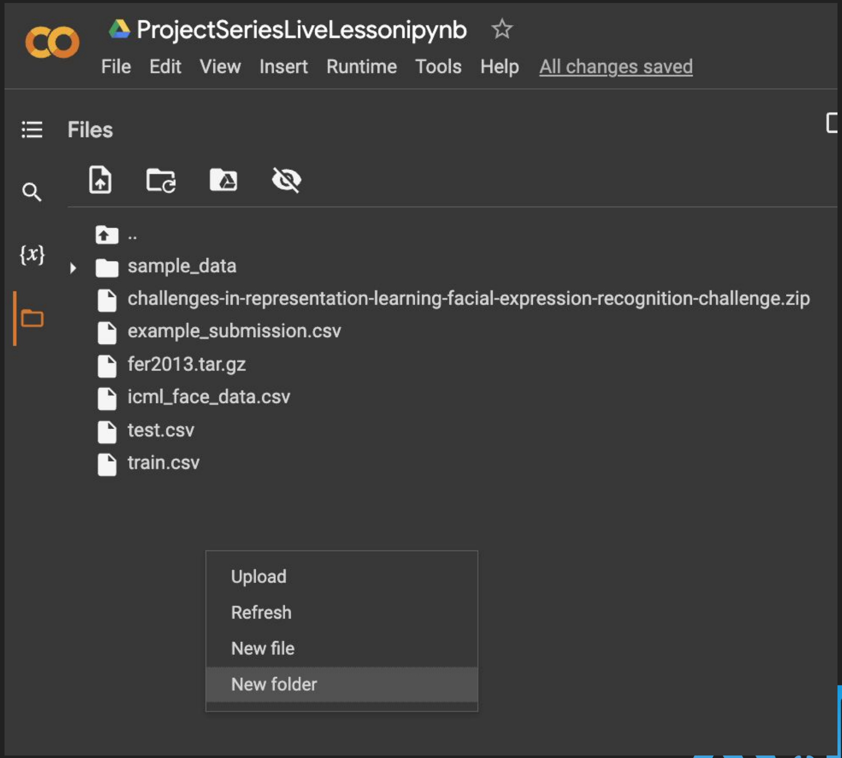
- Create a new code block with the button that says “ + Code ” at the top.
- Type in this line into that code block and click run (if your zip file is called something else, replace the “challenges-in-representation-learning-facial-expression-recognition-challenge” part with the name of your zip file
- Then wait a minute or two for the files to show up on your colab files.



```
!unzip /content/challenges-in-representation-learning-facial-expression-recognition-challenge.zip
```

Downloading the FER2013 Dataset into Google Colab

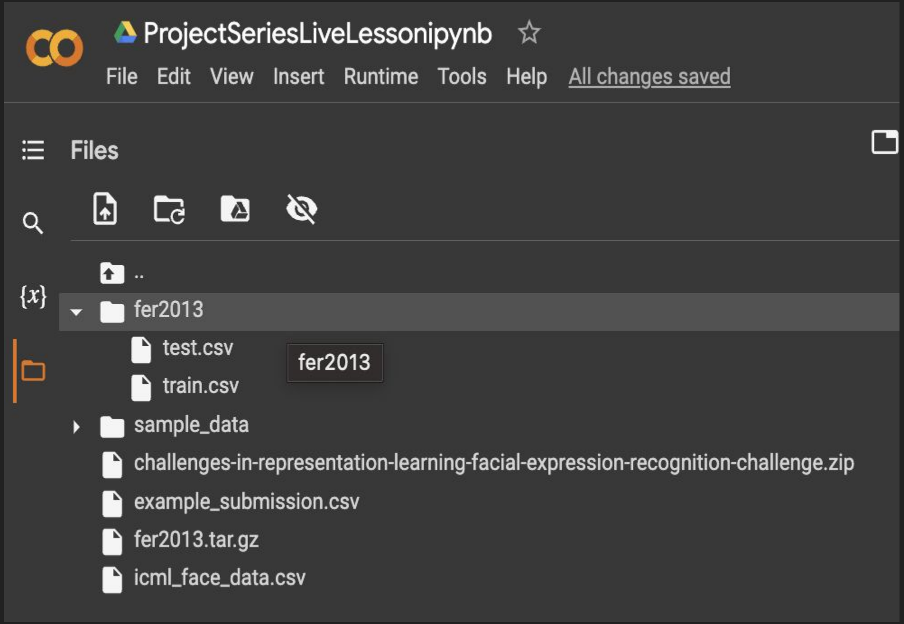
- Once all the files unzipped, right click in the file area and click “New folder” (as shown in the picture to the right)
- Title this folder “fer2013” (MAKE SURE TO COPY PASTE THIS EXACT)



ProjectSeriesLiveLessonipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
..
sample_data
challenges-in-representation-learning-facial-expression-recognition-challenge.zip
example_submission.csv
fer2013.tar.gz
icml_face_data.csv
test.csv
train.csv
Upload
Refresh
New file
New folder

Downloading the FER2013 Dataset into Google Colab

- Move the “train.csv” and “test.csv” files into the “fer2013” folder you just made.
- You should have something that looks like the picture to the right.



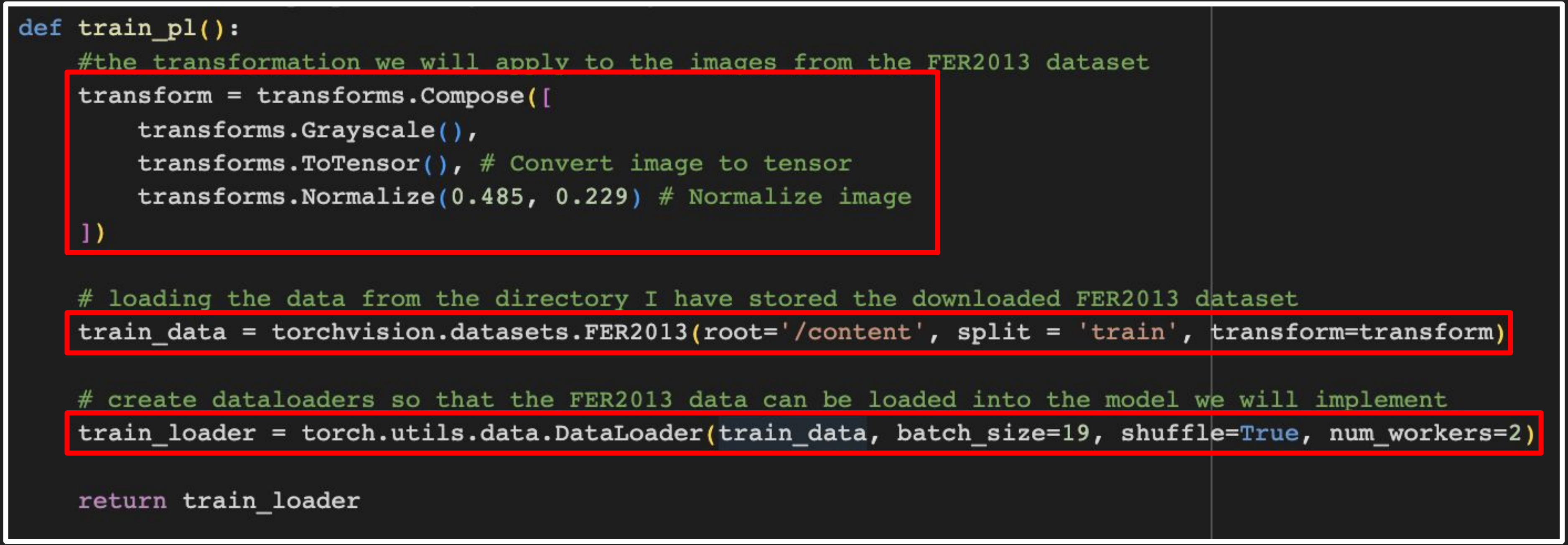
ProjectSeriesLiveLessonipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
..
fer2013
test.csv
train.csv
fer2013
sample_data
challenges-in-representation-learning-facial-expression-recognition-challenge.zip
example_submission.csv
fer2013.tar.gz
icml_face_data.csv

We Have the Dataset in Your Google Colab

You did it! You now have the dataset set up in your Google Colab environment.

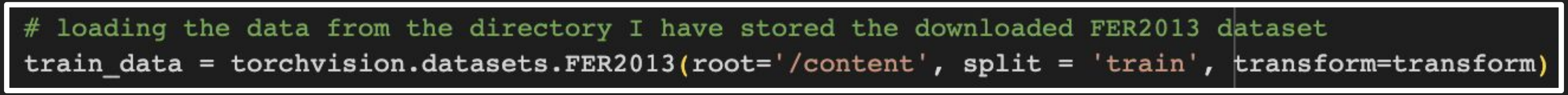
- Unfortunately you will have to do this setup every time your notebook loses connection (which it automatically does if you are inactive).
- SO DON’T DELETE THE ZIP FILE ON YOUR COMPUTER
- It will come in handy when you have to reupload the dataset back into your notebook.

Now to the Good Stuff



```
def train_pl():
    #the transformation we will apply to the images from the FER2013 dataset
    transform = transforms.Compose([
        transforms.Grayscale(),
        transforms.ToTensor(), # Convert image to tensor
        transforms.Normalize(0.485, 0.229) # Normalize image
    ])

    # loading the data from the directory I have stored the downloaded FER2013 dataset
    train_data = torchvision.datasets.FER2013(root='/content', split = 'train', transform=transform)

    # create dataloaders so that the FER2013 data can be loaded into the model we will implement
    train_loader = torch.utils.data.DataLoader(train_data, batch_size=19, shuffle=True, num_workers=2)

    return train_loader
```

PyTorch Datasets



```
# loading the data from the directory I have stored the downloaded FER2013 dataset
train_data = torchvision.datasets.FER2013(root='/content', split = 'train', transform=transform)
```

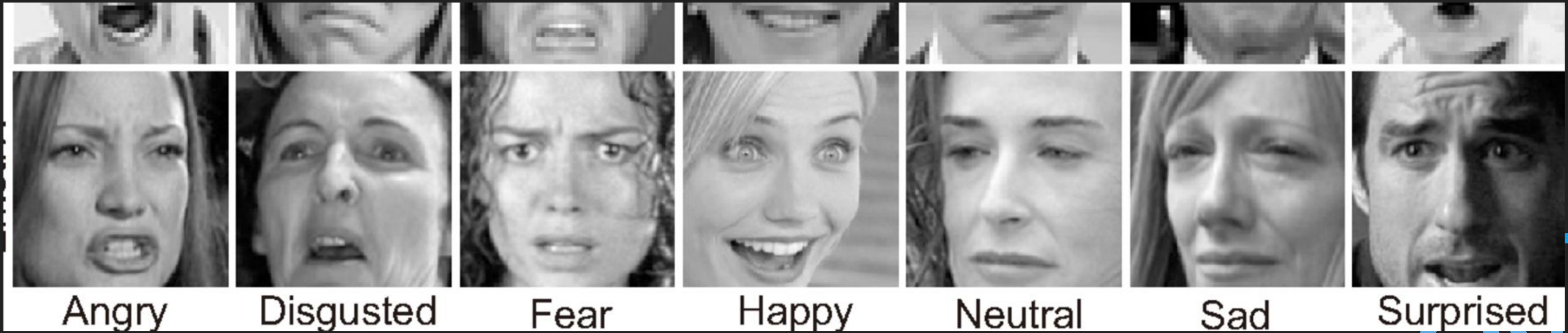
There's multiple parts to this statement:

- torchvision.datasets.FER2013
- 1st parameter: root = ‘/content’
- 2nd parameter: split = ‘train’
- 3rd parameter: transform = transform

Let’s look at each part

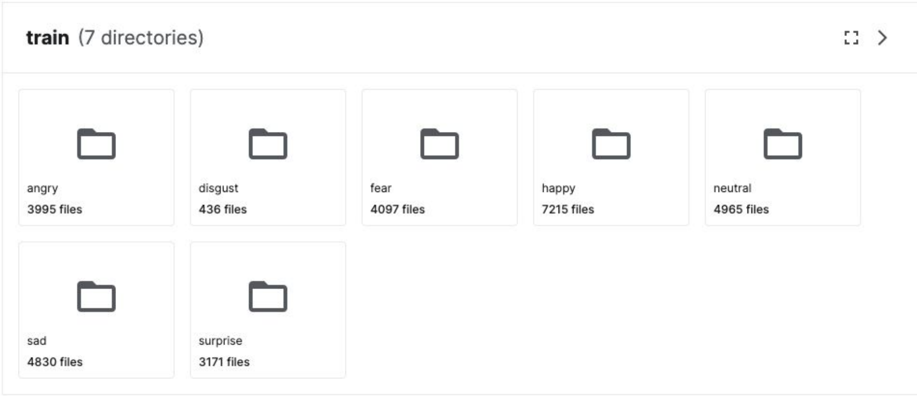
FER2013

- FER2013 is a highly popular dataset for facial emotion recognition
- It was released back in 2013 on Kaggle as a research attempt to open up the improvement of facial detection and recognition algorithms to the public.
- Still widely used today to train facial emotion recognition systems.



FER2013 (cont.)

- Datasets have labels and the corresponding data
- 7 classifications:
  - Happy, neutral, sad, fear, angry, surprise, and disgust
- 48 x 48 pixel images
- Grayscale images
  - A pixel can have a value between 0 (black) and 255 (white)
- Unequal distribution (look to the image on the right)



train (7 directories)

angry
3995 files

disgust
436 files

fear
4097 files

happy
7215 files

neutral
4965 files

sad
4830 files

surprise
3171 files

PyTorch Datasets

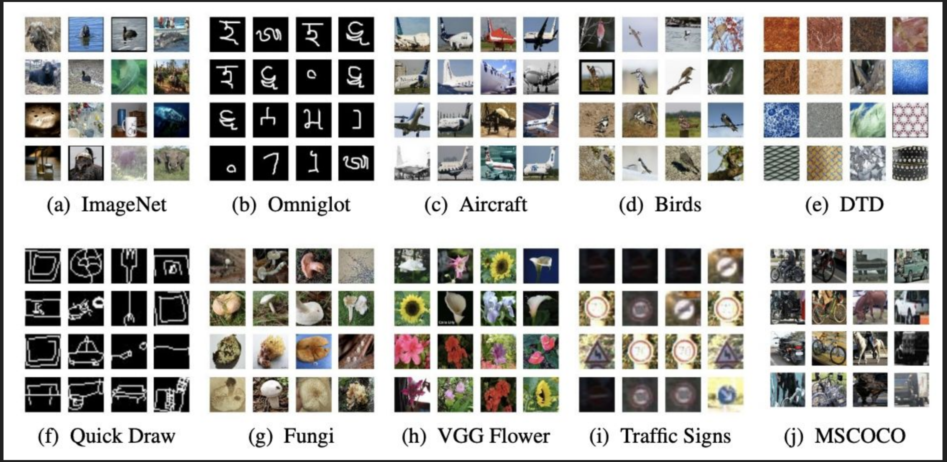
Dataset: a class in PyTorch that defines a way to represent your dataset so that it is organized, structured, and easy to use with PyTorch.

It allows you to:

- Get the length of your dataset
- Get individual data sample in your dataset

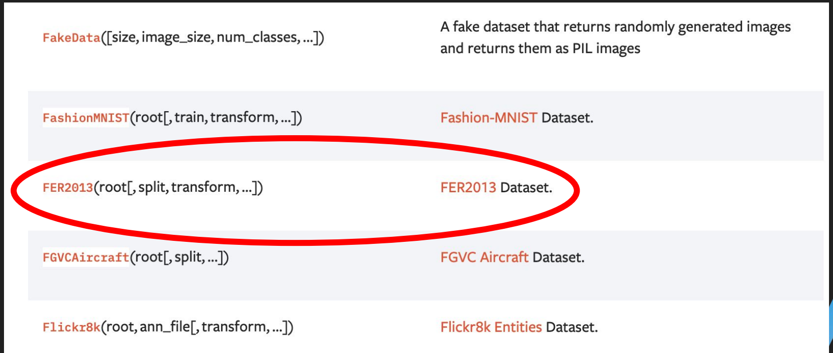
You could either:

- Use the built-in datasets provided by PyTorch (what we will be doing)
- Create your own custom PyTorch dataset by extending the Dataset class (tutorial can be found here at the part that says “Creating a Custom Dataset for your files”)



PyTorch Datasets

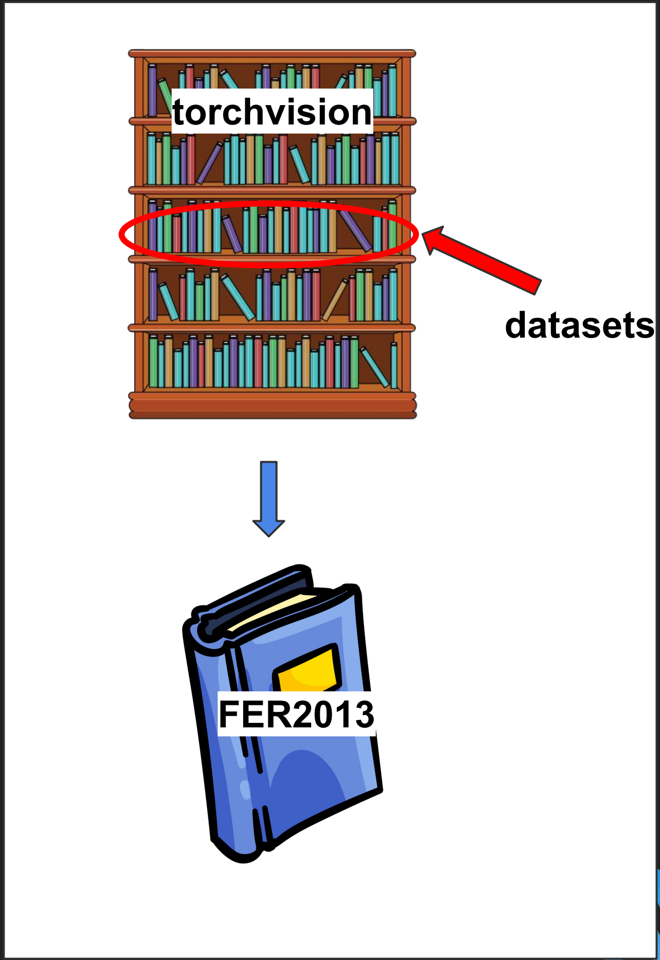
- Remember when we said that the PyTorch Datasets class already has built-in support for FER2013?
- If you visit the documentation for the Pytorch Datasets class, you'll see a list of very popular datasets for different deep learning use cases that PyTorch has built in support. One of those datasets is FER2013 …
- You should find it under the "Image Classification" section



FakeData([size, image_size, num_classes, ...])	A fake dataset that returns randomly generated images and returns them as PIL images
FashionMNIST(root[, train, transform, ...])	Fashion-MNIST Dataset.
FER2013(root[, split, transform, ...])	FER2013 Dataset.
FGVCAircraft(root[, split, ...])	FGVC Aircraft Dataset.
Flickr8k(root, ann_file[, transform, ...])	Flickr8k Entities Dataset.

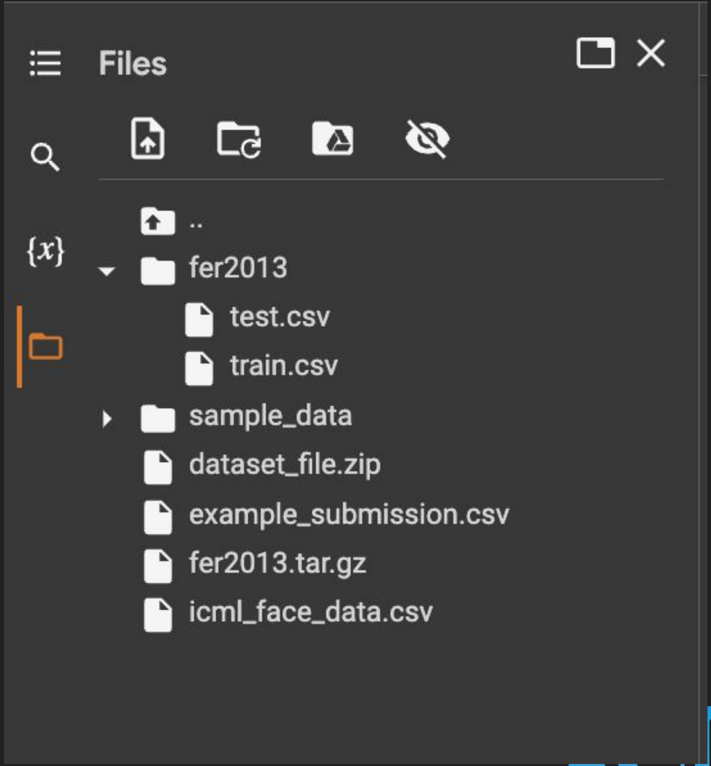
torchvision.datasets.FER2013

- “torchvision.datasets.FER2013” is how you access the built-in PyTorch dataset definition for FER2013 (for beginners, think of how you use System.out.print() in Java).
- “torchvision” is PyTorch’s accompanying library.
- “datasets” is a specific module in the library
- “FER2013” is the code for the class within “datasets” that defines how to use FER2013 with PyTorch
- Info on Python Modules can be found here



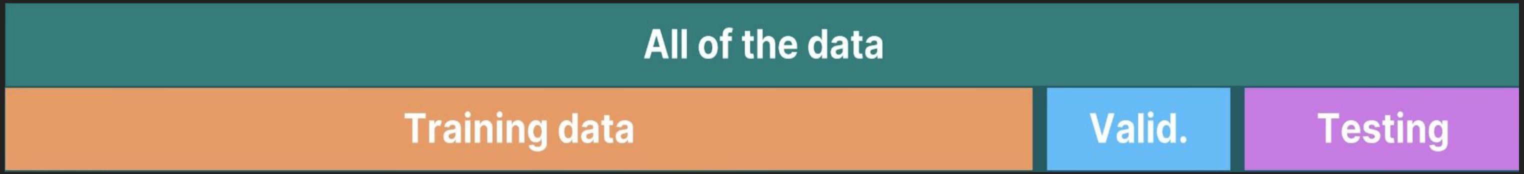
1st parameter: root = ‘/content’

- The root parameter should be set to a file path
- A file path is a set of directions the computer uses to navigate your file system and find what it needs
- In this case, the fer2013 folder we set up can be found in content directory of your notebook.
    - Kind of like Pomerene 160 can be found in Pomerene Hall



2nd parameter: split = ‘train’

- We’re telling torchvision that we want the training data from the dataset you can find at the file path we gave it.
- Datasets are usually split into 3 sections: training, validation, and testing.
- Best practice:
  - Training: 70%
  - Testing: 20%
  - Validation: 10%
- We won’t be going over validation in this course, we will only be going over training and testing.



All of the data

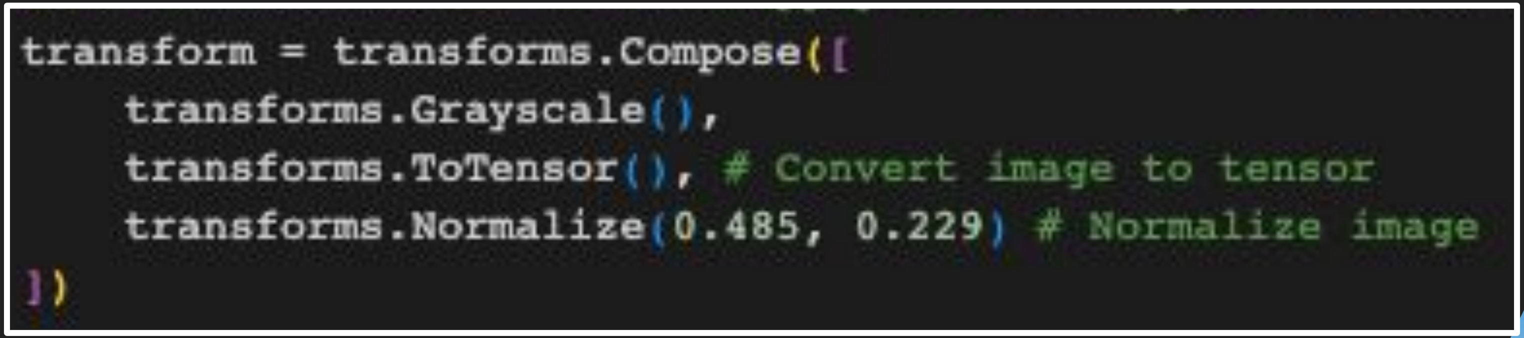
Training data

Valid.

Testing

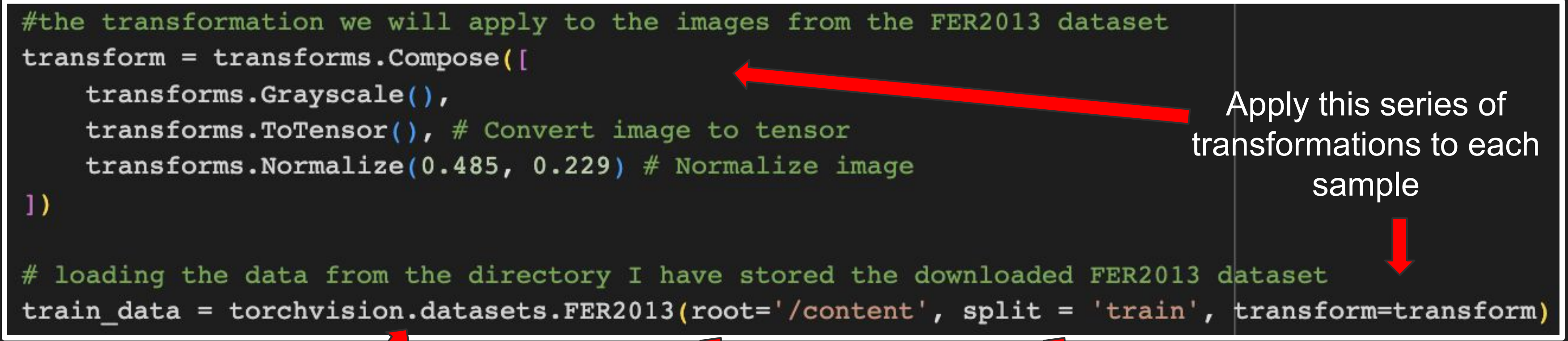
3rd parameter: transform = transform

- Before we start feeding the model our data, we need to adjust it with the specifications of our dataset. Make it more digestible and useful for our model.
- “transforms.Compose” is a function in torchvision that allows us to make a series of transformations.
- We want to convert our data samples to grayscale.
- We want to convert each sample to a tensor (required for feeding data into a model, more on this next week)
- We want to normalize our data so that we can standardize each sample
  - This involves statistics we won’t be going over this course, but the 0.485 is the mean and the 0.229 is the standard deviation. In all honesty, I got those numbers from ChatGPT.



```
transform = transforms.Compose([
    transforms.Grayscale(),
    transforms.ToTensor(), # Convert image to tensor
    transforms.Normalize(0.485, 0.229) # Normalize image
])
```

Tying everything together



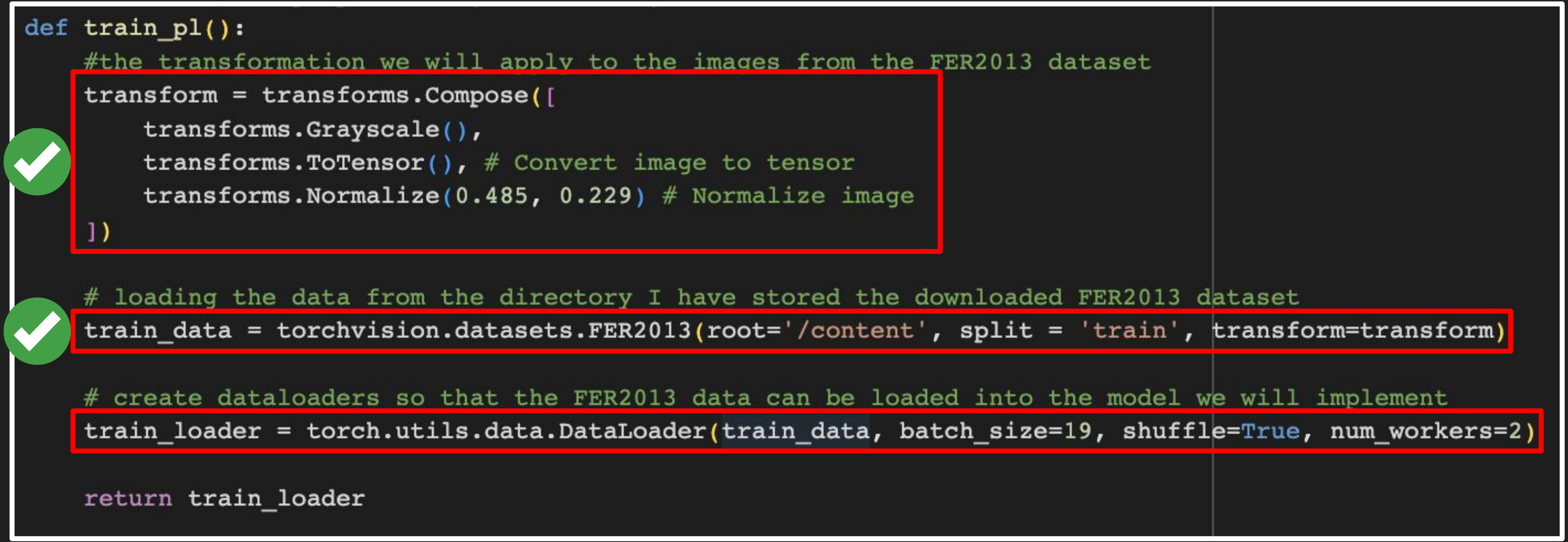
```
#the transformation we will apply to the images from the FER2013 dataset
transform = transforms.Compose([
    transforms.Grayscale(),
    transforms.ToTensor(), # Convert image to tensor
    transforms.Normalize(0.485, 0.229) # Normalize image
])

# loading the data from the directory I have stored the downloaded FER2013 dataset
train_data = torchvision.datasets.FER2013(root='/content', split = 'train', transform=transform)
```

Apply this series of transformations to each sample

Create a new Dataset with the FER2013 blueprint

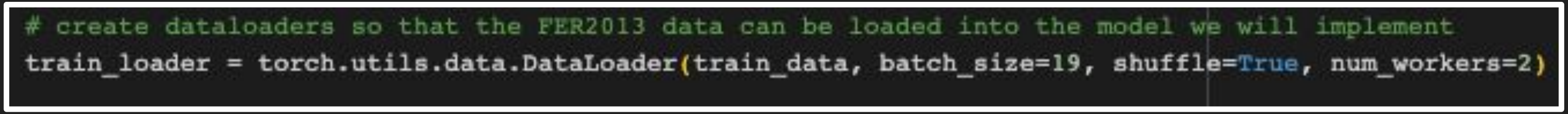
Find the dataset at this location in the file system

Only retrieve the training data



```
def train_pl():
    #the transformation we will apply to the images from the FER2013 dataset
    transform = transforms.Compose([
        transforms.Grayscale(),
        transforms.ToTensor(), # Convert image to tensor
        transforms.Normalize(0.485, 0.229) # Normalize image
    ])

    # loading the data from the directory I have stored the downloaded FER2013 dataset
    train_data = torchvision.datasets.FER2013(root='/content', split = 'train', transform=transform)

    # create dataloaders so that the FER2013 data can be loaded into the model we will implement
    train_loader = torch.utils.data.DataLoader(train_data, batch_size=19, shuffle=True, num_workers=2)

    return train_loader
```

PyTorch DataLoader



```
# create dataloaders so that the FER2013 data can be loaded into the model we will implement
train_loader = torch.utils.data.DataLoader(train_data, batch_size=19, shuffle=True, num_workers=2)
```

There's multiple parts to this statement:

- torch.utils.data.DataLoader
- 1st parameter: train_data
- 2nd parameter: batch_size = 19
- 3rd parameter: shuffle = True
- 4th parameter: num_workers = 2

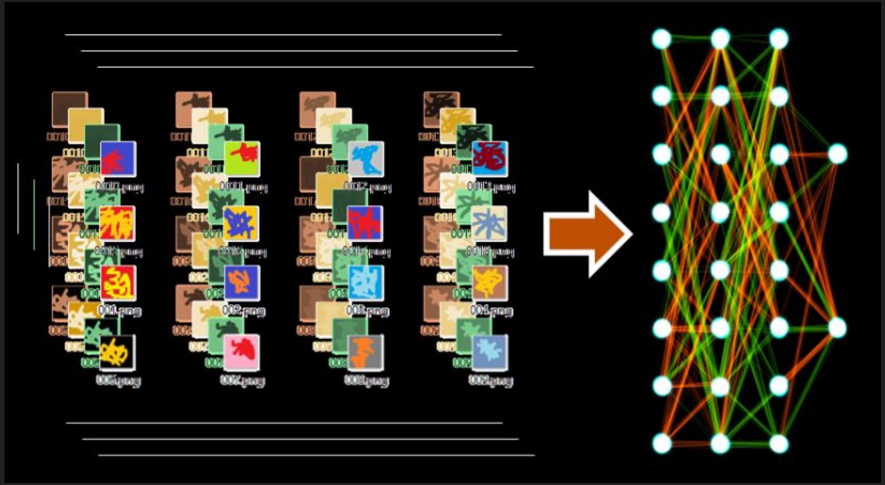
Let’s look at each part

PyTorch DataLoader

DataLoader: a class in PyTorch that sets up your dataset in a way where it could be fed into your model.

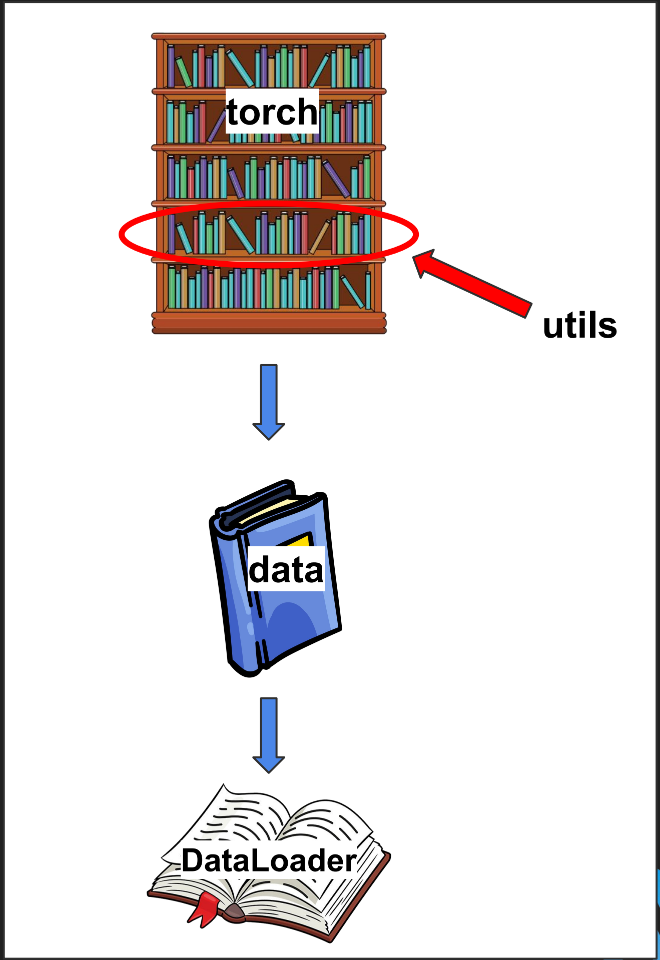
It allows you to:

- Cut your data into batches
- Iterate through your data/batches
- Shuffle your data to prevent overfitting
- Create multiple pipelines
- etc.
- **More on these later…**



torch.utils.data.DataLoader

- “torch.utils.data.DataLoader” is how you access the code for DataLoader (for beginners, think of how you use System.out.print() in Java).
- “torch” is the PyTorch library
- “utils” is a specific module in the library
- “data” is another module within “utils”
- “DataLoader” is the code for the class within “data”
- Info on Python Modules can be found [here]



1st parameter: train_data

- The DataLoader class needs information from you first to create a new DataLoader object
  - Kind of like when you need ingredients in order to follow a cooking recipe
- The first one being the actual data.
- It needs to know the dataset that you want to train your model on.
- Only accepts PyTorch datasets.
- This is why we converted the dataset we have into a PyTorch dataset and stored it in a variable called “train_data”

2nd parameter: batch_size = 19 (**BATCH TRAINING)**

Batch Training: a machine learning technique that is commonly used for training deep learning models.

- We cut down the dataset into multiple batches
  - In this case, multiple batches with 19 data points.
- We feed the deep learning those smaller batches instead of the whole dataset at once
- More efficient because we aren’t overloading the model
- More accurate because we’re having the model train more.

What’s the catch?

- It requires more memory on your computer, but since we are using Google Colab, we don’t need to worry about that (as much)

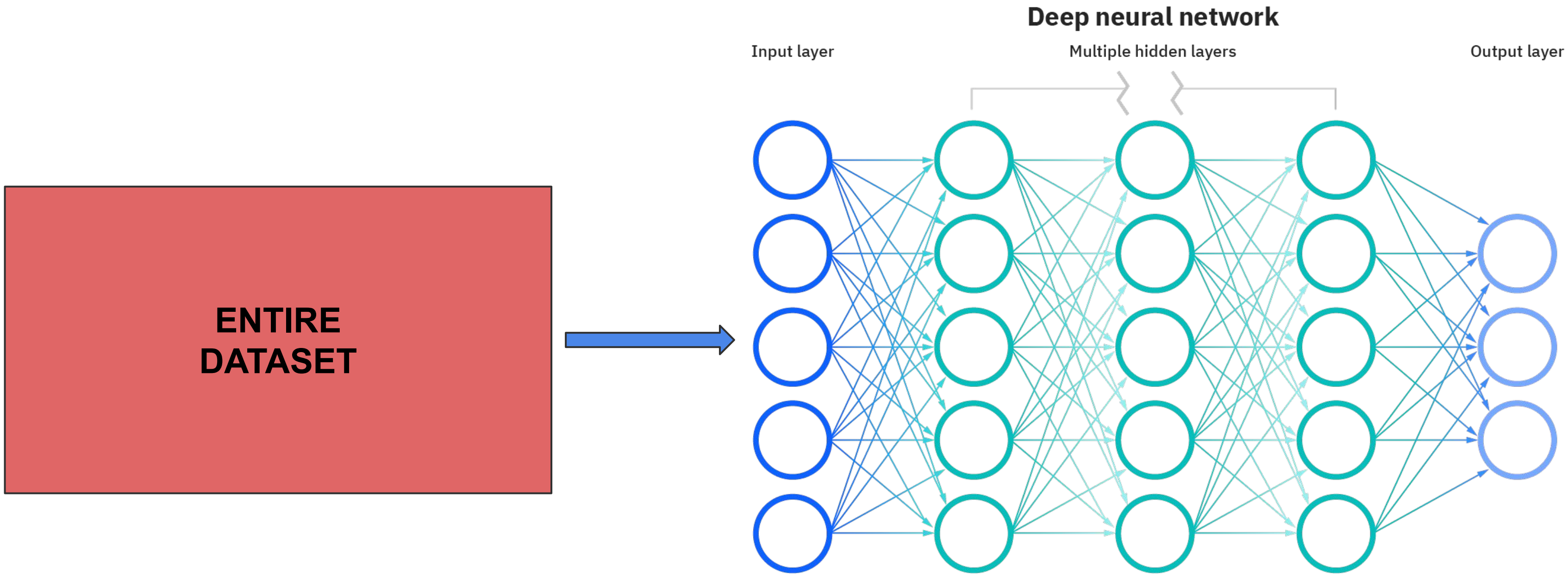
See proceeding slides for example and illustration

Motivating Example for Batch Training

Consider eating a large piece of steak:

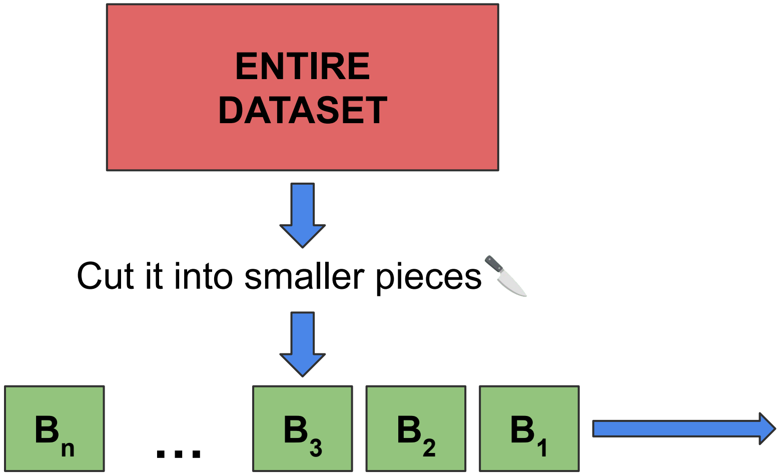
- Would you eat the steak whole?
- Or cut it into smaller pieces?
- Eating the steak whole would take a lot of time, effort, strain, and you don’t get to enjoy it as much.
- Cutting it into pieces is easier, more efficient, and you get to enjoy your food.

No Batch Training



ENTIRE DATASET

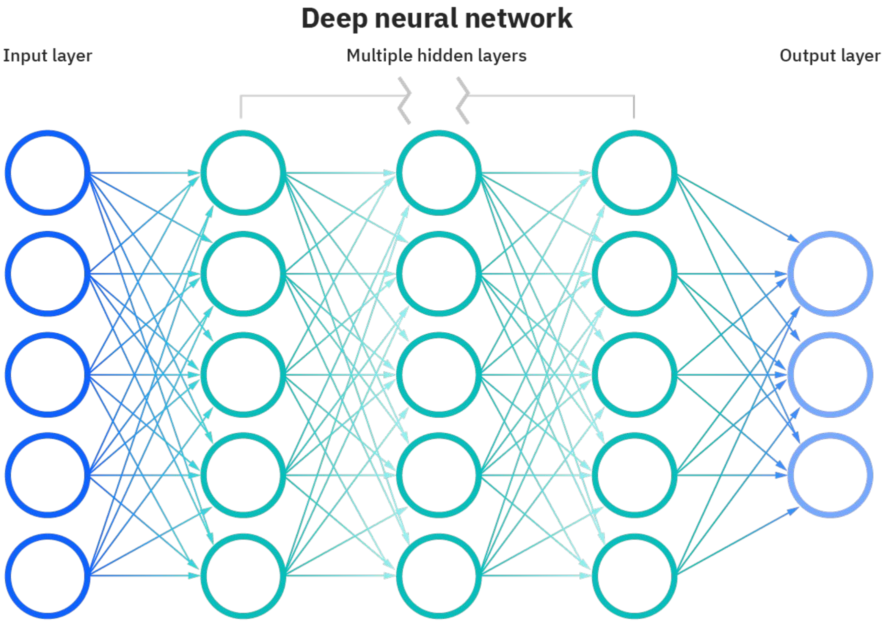
vs. Batch Training



ENTIRE DATASET

Cut it into smaller pieces🔪

B_n … B_3 B_2 B_1



3rd parameter: shuffle = True

- We’re setting the shuffle parameter to true.
- We’re taking the batches we made and mixing them up.
- We want to do this to prevent something called **overfitting**
- **Overfitting**: when you deep learning models memorizes the data it trained on
  - It’ll do really well everytime you train, but it’ll do really bad when you actually test it.
  - Imagine if you were trying to teach a kid how to read. Would you have them memorize a paragraph, or try to understand the words in the paragraph?

4th parameter: num_workers = 2

- “num_workers” is a parameter that allows you to create worker processes.
- These worker processes branch off from your main process.
- Your main process is how the data is being loaded in by default.
- Allows for parallel data loading.
  - Higher speeds when training your model
  - Requires more memory on your computer
- If you only use the main process, all of your data will be fed to the model in a single feed.
- If you use the main process + worker processes, your data will be split up fed to the model in multiple feeds.
- See next slide for example

Motivating Example for num_workers

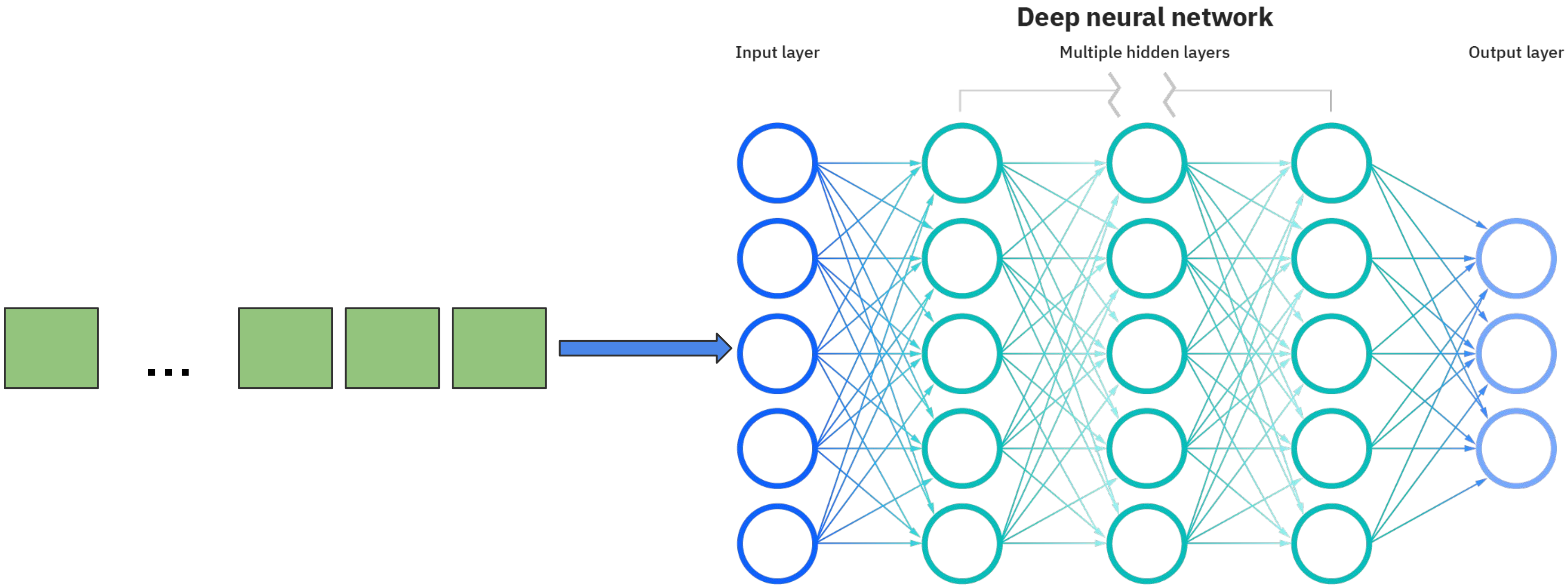
U.S. Department of Homeland Security

Transportation Security Administration

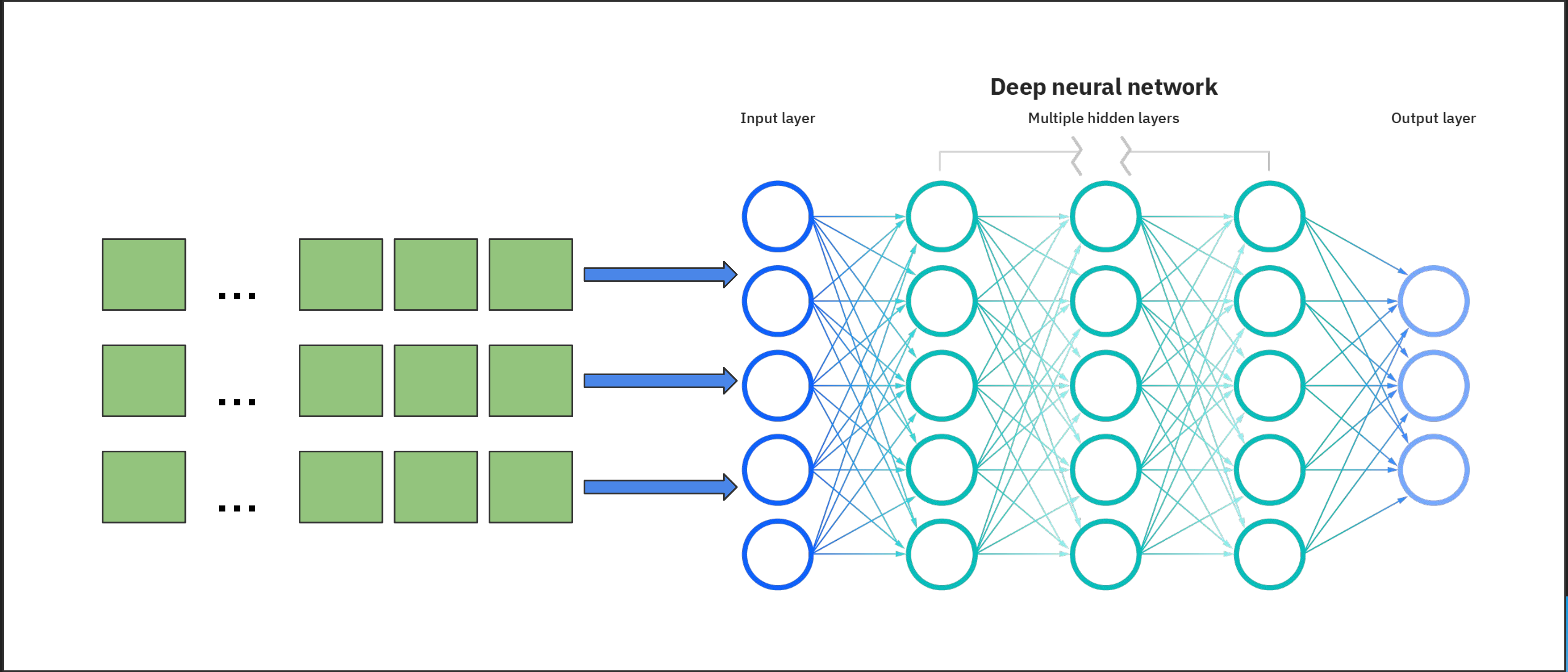
Consider TSA at airports

- A lot of times there's way too many people and security only has 1-3 checkpoints open for people to pass through
- Inefficient, inaccurate, and a real pain in the ass
- Wouldn’t it be better to have more checkpoints open?

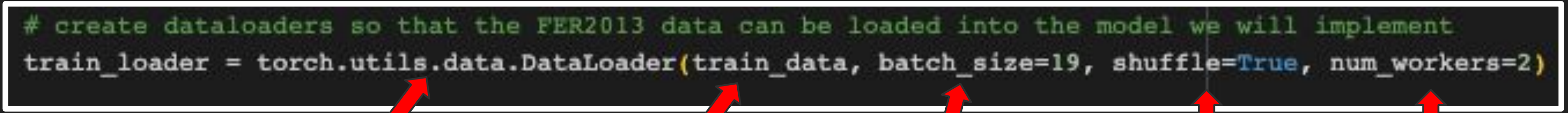
Main process only



vs. main process + worker processes



Tying everything together



```
# create dataloaders so that the FER2013 data can be loaded into the model we will implement
train_loader = torch.utils.data.DataLoader(train_data, batch_size=19, shuffle=True, num_workers=2)
```

Create a new DataLoader object

Using the FER2013 train dataset

Split the train data up into batches of size 19

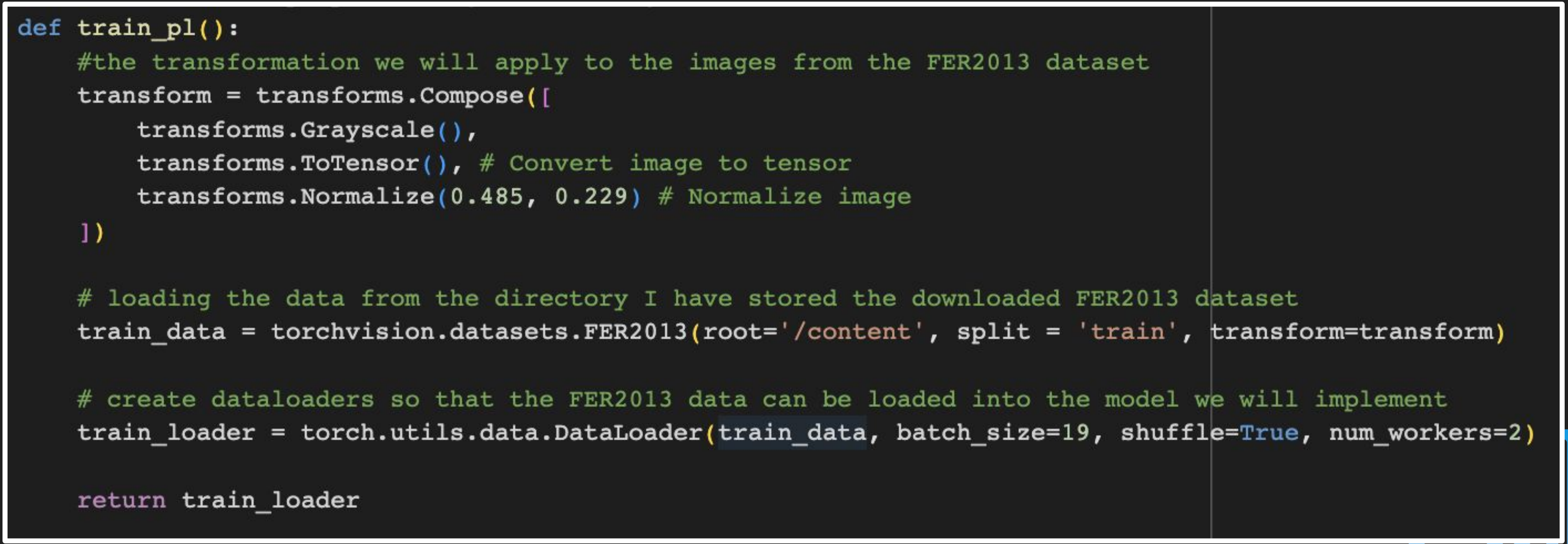
Shuffle the batches

Create 2 extra worker processes (main + worker + worker = 3 streams)

Congrats!

You now have your dataset fully set up and ready to go

- With this, you will be able to feed your data into the model you will make.



```
def train_pl():
    #the transformation we will apply to the images from the FER2013 dataset
    transform = transforms.Compose([
        transforms.Grayscale(),
        transforms.ToTensor(), # Convert image to tensor
        transforms.Normalize(0.485, 0.229) # Normalize image
    ])

    # loading the data from the directory I have stored the downloaded FER2013 dataset
    train_data = torchvision.datasets.FER2013(root='/content', split = 'train', transform=transform)

    # create dataloaders so that the FER2013 data can be loaded into the model we will implement
    train_loader = torch.utils.data.DataLoader(train_data, batch_size=19, shuffle=True, num_workers=2)

    return train_loader
```

Next week: creating a model

What “making a model” sounds like

What it’s actually like (with PyTorch)

From now on meetings are in Hitchcock 031!!

Will be posted in Announcements, Discord, and changed on Website

CLUB

Enjoy your week!

go.osu.edu/aiclubsignup

CLUB