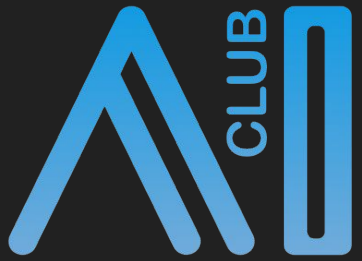


Autumn 2023
Week 5

go.osu.edu/aiclubsignup

IBM Z Career Connection
The Ohio State University

Learn about IBM Z: the extremely powerful computers making the financial world go round with a focus on security, AI, and open source tech!

Pomerene Hall Rm 160
October 16
6:00-8:00pm ET

ibm.biz/OSU-ZCC

Sponsored by IBM

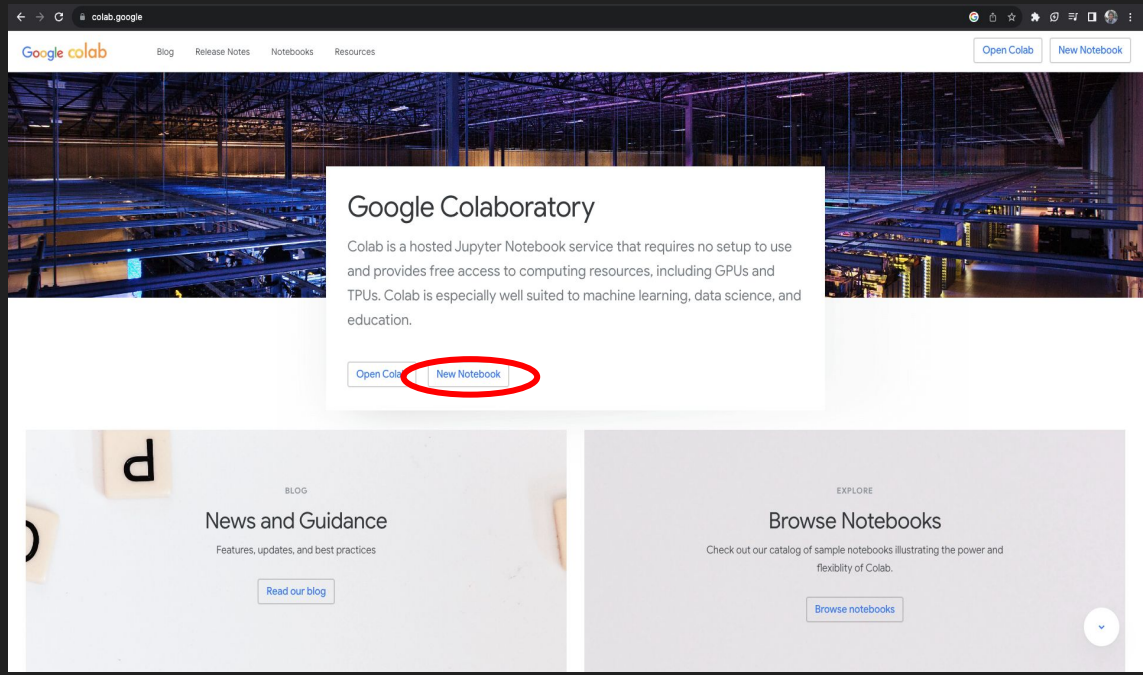
Sponsored by AI CLUB

Plan for today

- What goes behind training a model
- How to train your model

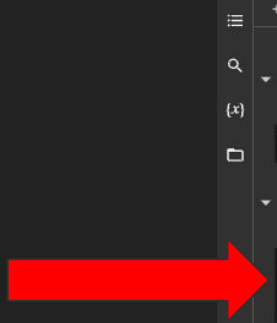
Before we start…

- Open up Google Colab
- Open your previous notebook
- https://colab.research.google.com/



Enter this into your import code block

```
from torch import nn, save, load
from torch.optim import Adam
```



ProjectSeriesLiveLessonipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Getting the dataset into your google colab

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

Importing necessary libraries

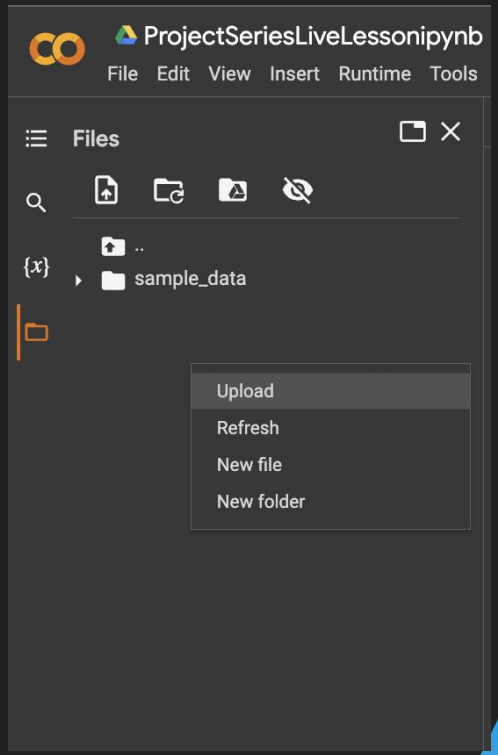
```
[ ] import torch
    import torchvision
    from torchvision import transforms
```

Data Preprocessing & Loading

```
#this is for data preprocessing and loading with train 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
```

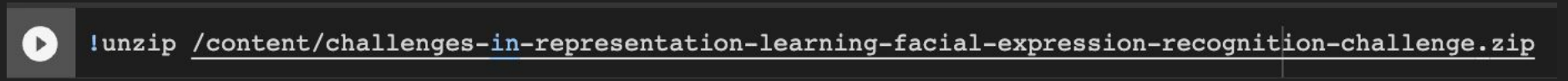
Downloading Dataset into Google Colab (renewed)

- 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



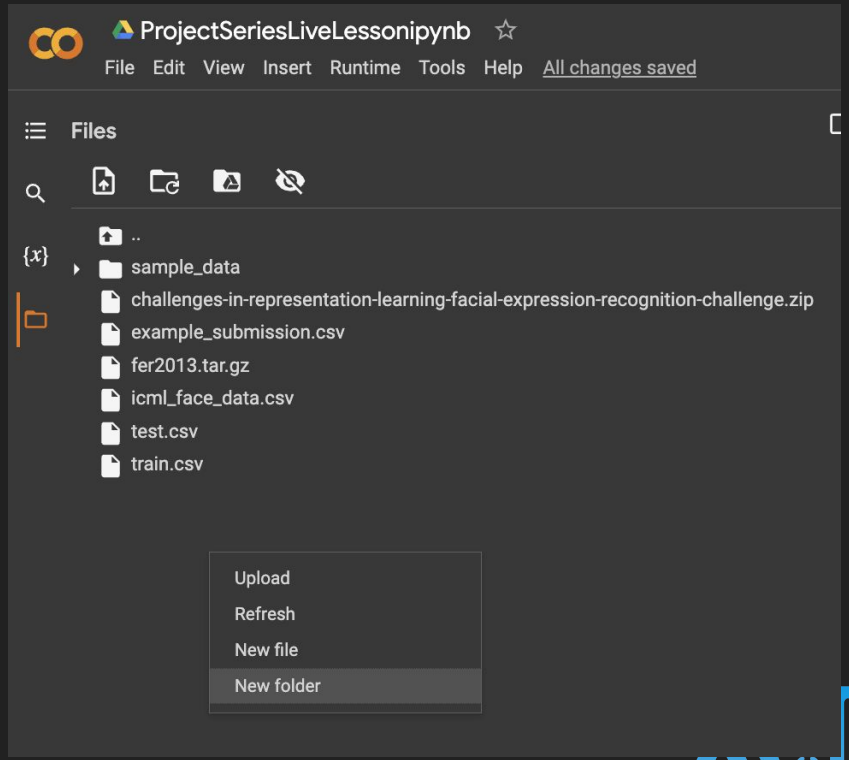
Downloading Dataset into Google Colab (renewed)

- 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.
- DELETE this code block from your notebook once you’ve done this!!!



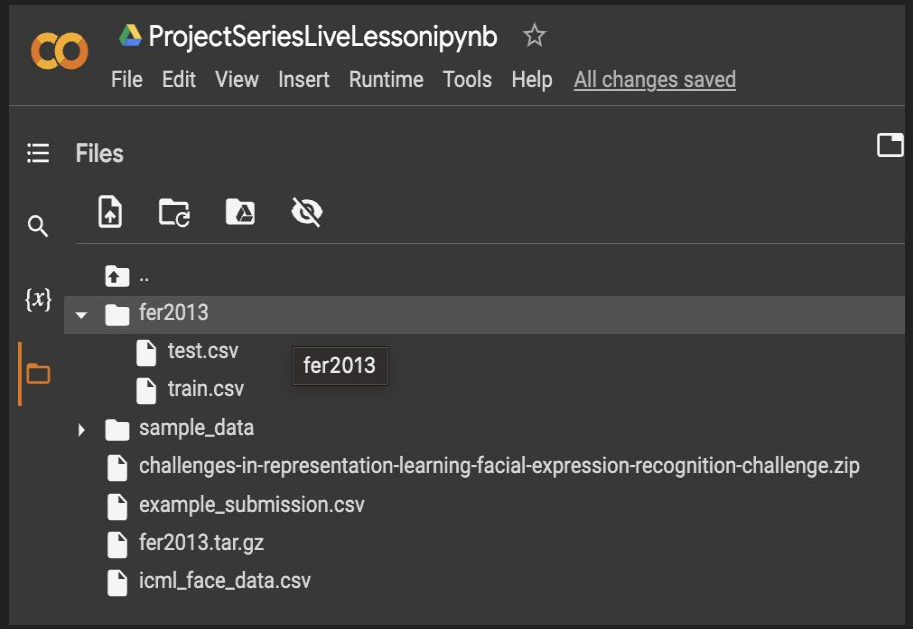
Downloading Dataset into Google Colab (renewed)

- 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)

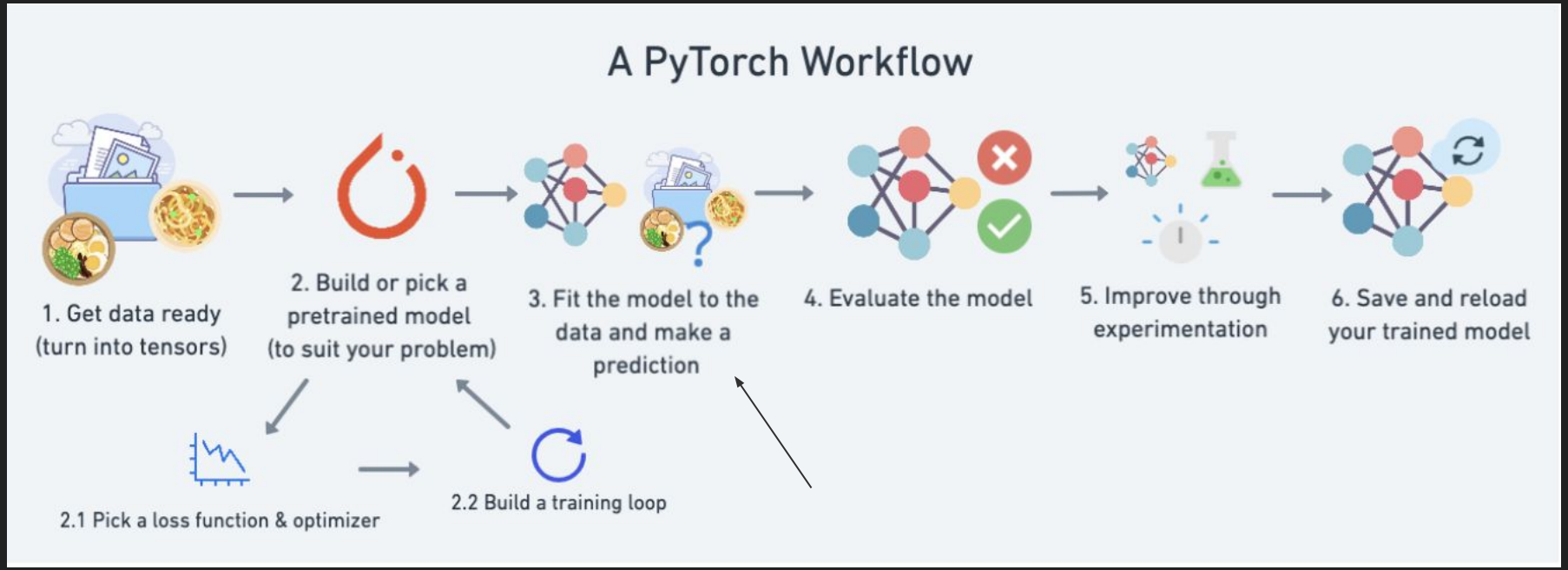


Downloading Dataset into Google Colab (renewed)

- 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.



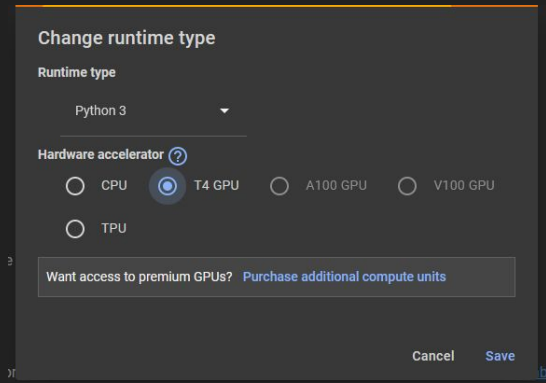
Our Goal:


A PyTorch Workflow
1. Get data ready (turn into tensors)
2. Build or pick a pretrained model (to suit your problem)
3. Fit the model to the data and make a prediction
4. Evaluate the model
5. Improve through experimentation
6. Save and reload your trained model
2.1 Pick a loss function & optimizer
2.2 Build a training loop

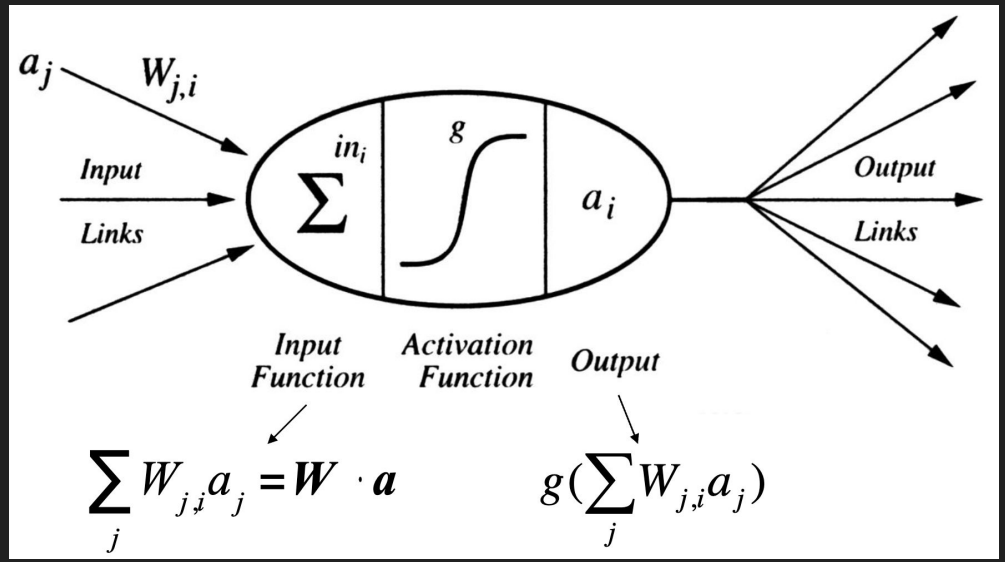
Setting Colab up with a GPU:

WITH the runtime prepared (all data set up):

1. Click the “Runtime” button under the notebook title
2. Click “Change runtime type” in the dropdown
3. Select T4 GPU and click “Save”



Context: The Anatomy of a Neuron



Some Definitions:

- **Loss**: A numeric value representing how bad your model is.
- **Loss Function**: The equation used to calculate the loss.
- **Optimizer**: The algorithm used to adjust the model and decrease the *Loss* after each batch of data.
- **Epoch**: One run of the training loop, AKA one training session using all of the training data.

Loss and the Loss Function

Cool depictions of Loss Functions: https://losslandscape.com/gallery/

In our case (image classification), the ideal loss function is “Cross Entropy Loss”.

The theory behind cross entropy loss is worthy of its own lecture…

v7Labs Overview of Cross Entropy

To calculate model loss, the outputs are compared to the expected outputs.

The closer the loss is to 0, the better the model is.

Gradient Descent

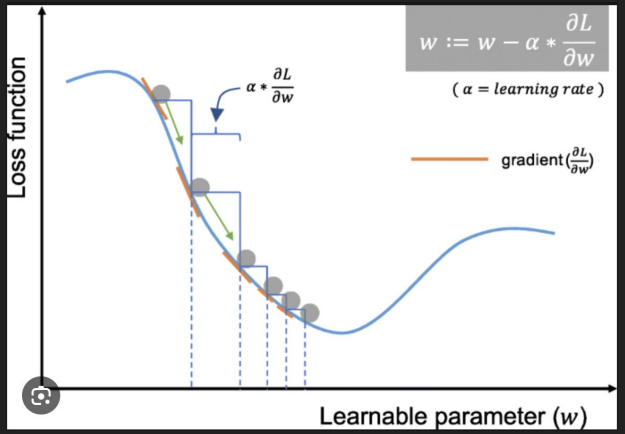
Definition: The process of gradually minimizing the loss value

Think about it like descending down a hill.

Calculus connection: Finding the local minima of a 2D function.

In our model, this process will be done with the Adam optimizer, an improvement on stochastic gradient descent. Here's an overview:

[Stochastic Gradient Descent]



What does training actually do?

During one epoch, the model will…

1. Do a “forward pass”, of each batch of training data
    a. Loss is calculated after each batch
2. Calculate the average loss of the model
3. Update the parameters of each neuron
    a. Parameters = weights - think back to the anatomy of a neuron
    b. This process is called backpropagation (again, theory worthy of a whole lecture)

Our code for training the model

```
[ ]  def train():
       model = EmotionModel().to('cuda')
       optimizer = Adam(model.parameters(), lr = 1e-3)
       loss_fn = nn.CrossEntropyLoss()

       train_set = train_pl()

       for epoch in range(50): #train for 50 epochs
           for batch in train_set:
               X, y = batch
               X, y = X.to('cuda'), y.to('cuda')
               prediction = model(X)
               loss = loss_fn(prediction, y)

               optimizer.zero_grad()
               loss.backward()
               optimizer.step()

           print(f"Epoch {epoch+1}\n-------------------------------")
           print(f"\tloss:{loss}")
           print("-----------------------------------------------")

       # saving our model to our environment

       return model
```

1. Setting up the needed components and variables: the model, an optimizer, a loss function, and the training data set (more on optimizer and loss fn later)
2. Training the model over and over again

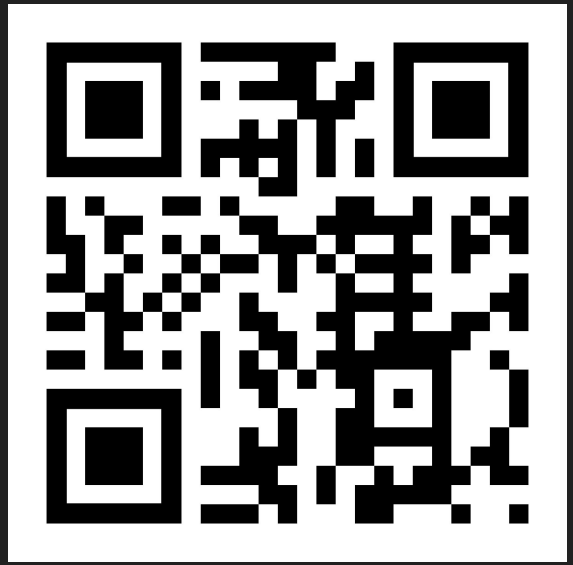
Component and Variable Initializations

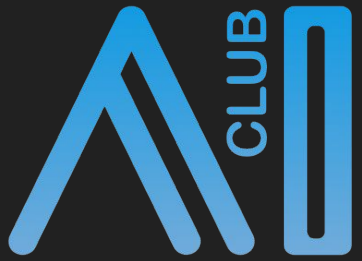
```
model = EmotionModel().to('cuda')
optimizer = Adam(model.parameters(), lr = 1e-3)
loss_fn = nn.CrossEntropyLoss()

train_set = train_pl()
```

- First line creates a new instance of our model and sets it up on a GPU (faster to train on GPU than CPU).
- Sets up an optimizer called Adam.
- Sets up a cross entropy loss function.
- Sets up the training dataset.
- More on the optimizer and loss function later.

First Time Sign up

AI Club Website





Enjoy your week!

go.osu.edu/aiclubsignup