Autumn 2023
Week 6

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

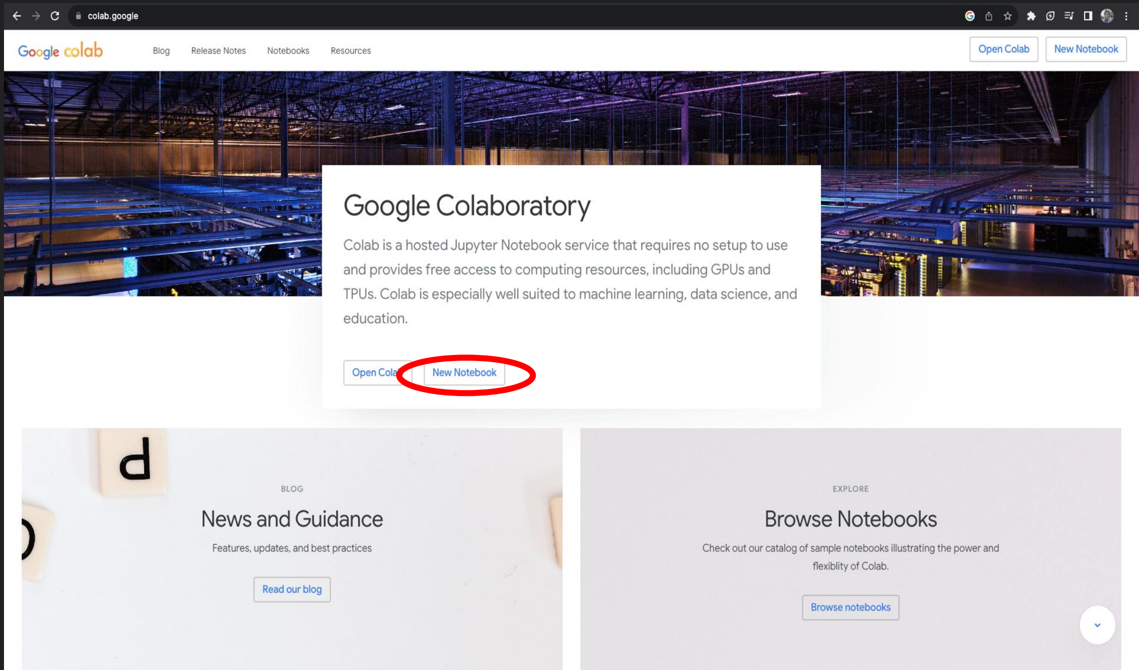
Sponsored by
AI CLUB

Plan for today

- Evaluating our model
- Saving and loading our trained model
- Using our model in code

Before we start…

- Open up Google Colab
- Open your previous notebook
- 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.

BLOG

News and Guidance

Features, updates, and best practices

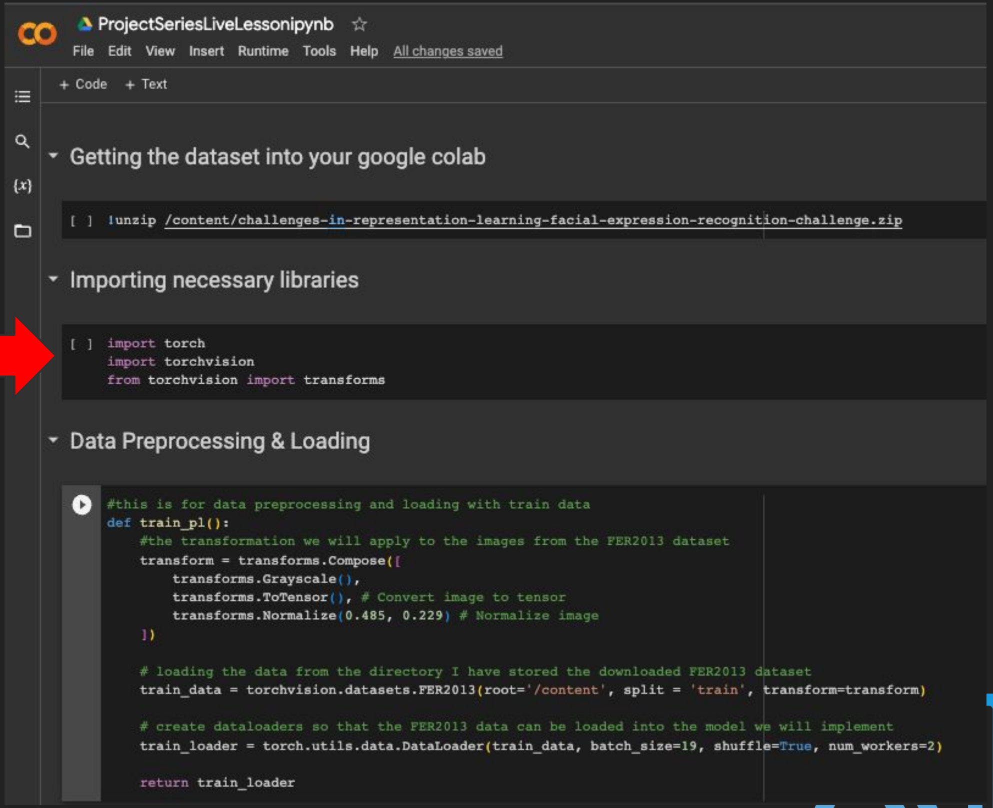
EXPLORE

Browse Notebooks

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

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

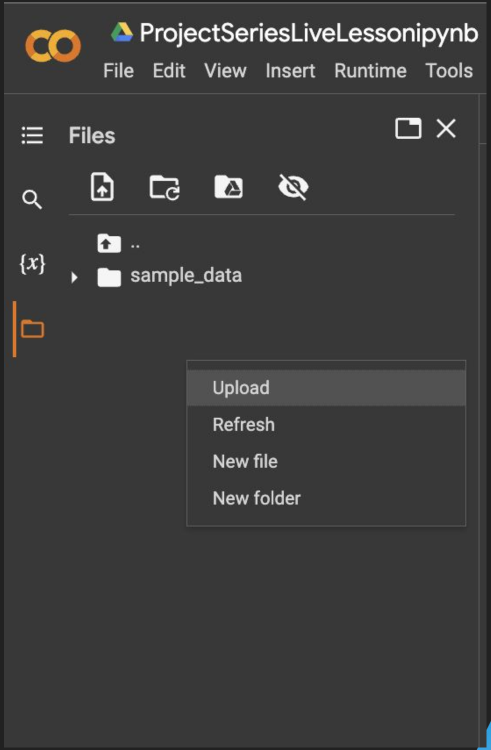
```
import torch
import torchvision
from torchvision import transforms
```

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
```

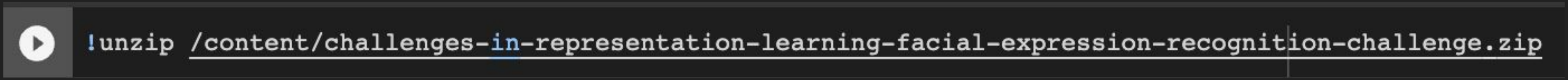
Downloading Dataset into Google Colab (renewed)

- 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)

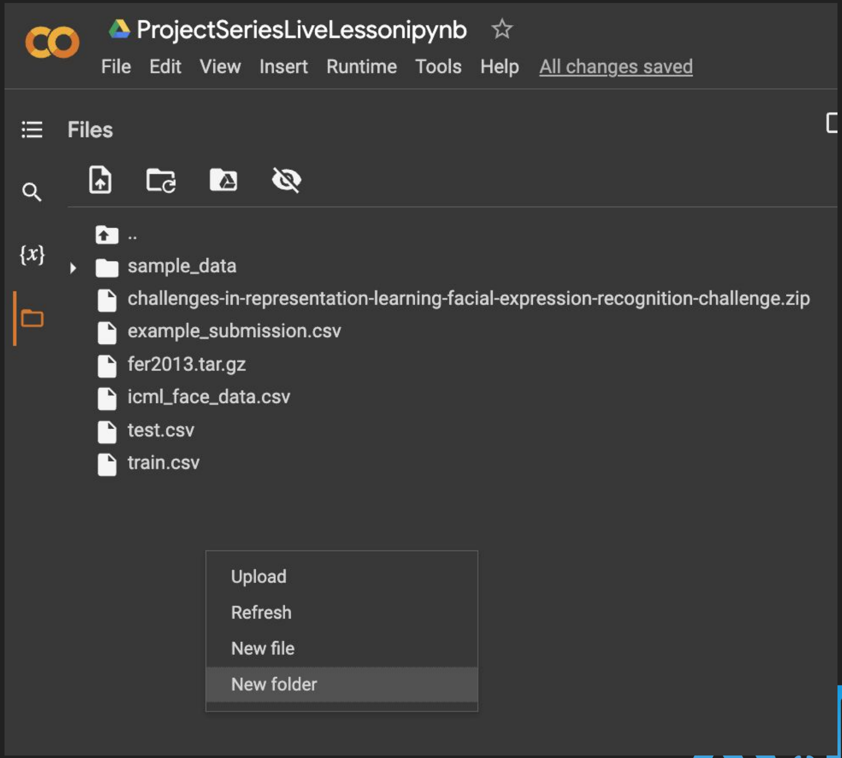
- Create a new code block with the button that says “ + Code ” at the top.
- Type in this line into that code block and click run (if your zip file is called something else, replace the “challenges-in-representation-learning-facial-expression-recognition-challenge” part with the name of your zip file
- 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!!!



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

Downloading Dataset into Google Colab (renewed)

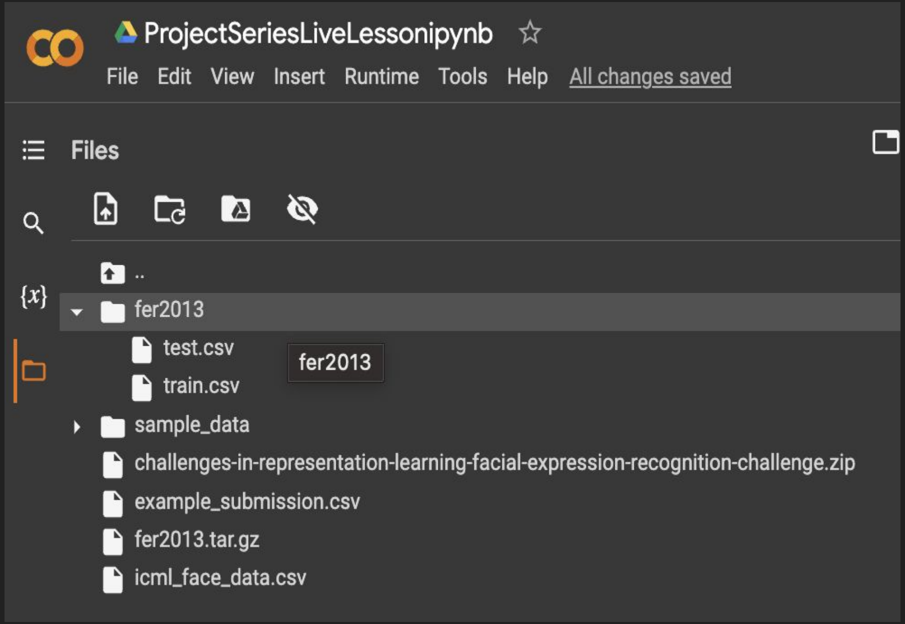
- Once all the files unzipped, right click in the file area and click "New folder" (as shown in the picture to the right)
- Title this folder "fer2013" (MAKE SURE TO COPY PASTE THIS EXACT)



challenges-in-representation-learning-facial-expression-recognition-challenge.zip
example_submission.csv
fer2013.tar.gz
icml_face_data.csv
test.csv
train.csv

Downloading Dataset into Google Colab (renewed)

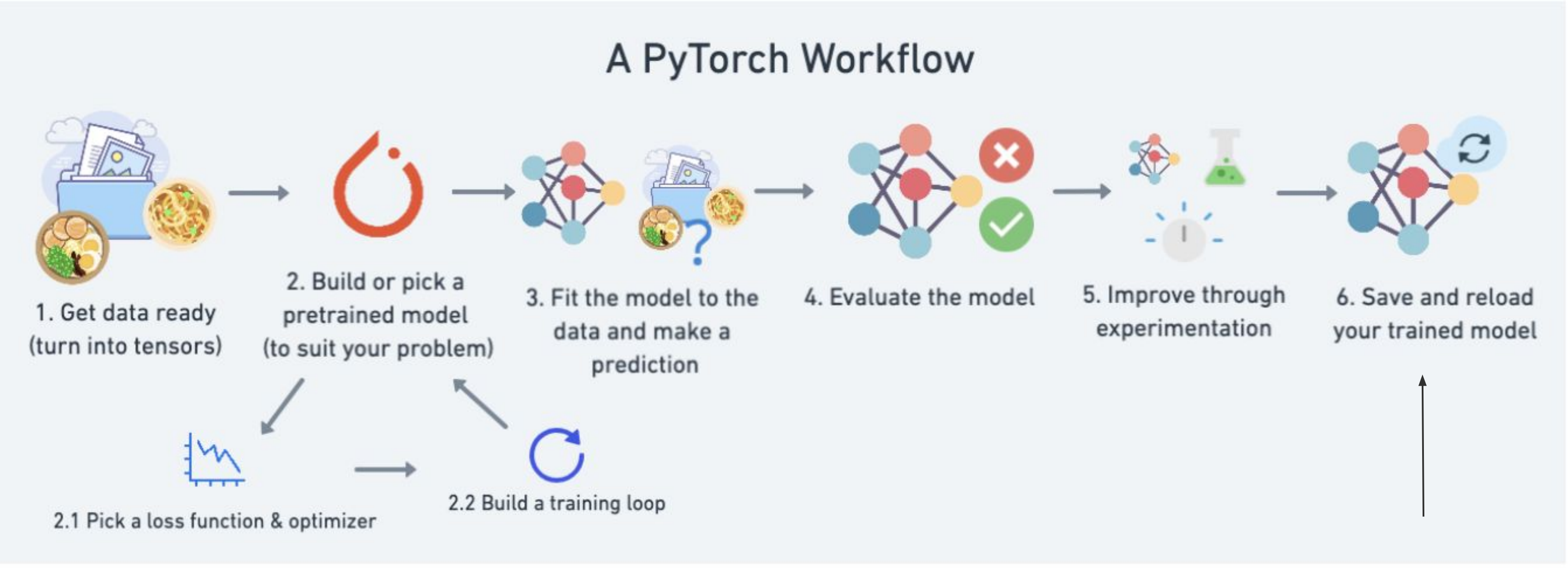
- Move the “train.csv” and “test.csv” files into the “fer2013” folder you just made.
- You should have something that looks like the picture to the right.



CO
ProjectSeriesLiveLessonipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
Files
{x}

```
..
fer2013
    test.csv
    train.csv
sample_data
challenges-in-representation-learning-facial-expression-recognition-challenge.zip
example_submission.csv
fer2013.tar.gz
icml_face_data.csv
```

Our Goal:

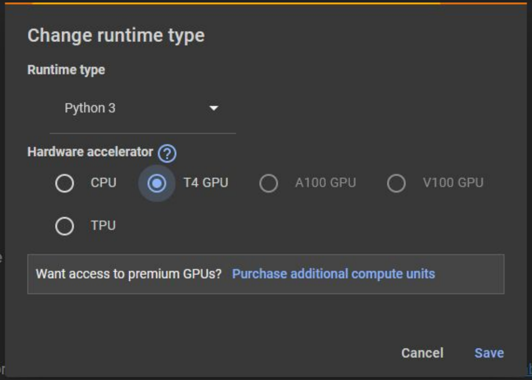


A PyTorch Workflow
1. Get data ready (turn into tensors)
2. Build or pick a pretrained model (to suit your problem)
3. Fit the model to the data and make a prediction
4. Evaluate the model
5. Improve through experimentation
6. Save and reload your trained model
2.1 Pick a loss function & optimizer
2.2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



Change runtime type

Runtime type

Python 3

Hardware accelerator

○ CPU ● T4 GPU ○ A100 GPU ○ V100 GPU

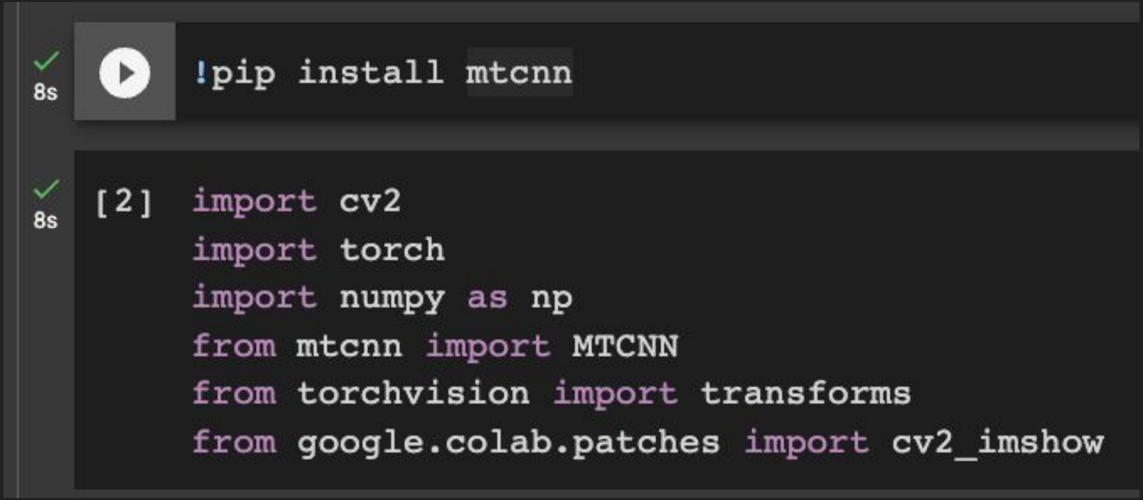
○ TPU

Want access to premium GPUs? Purchase additional compute units

Pitfall of FER2013

- In the dataset we downloaded at the beginning of this course, the test.csv file is corrupted.
- There is some issue with the actual data values being stored.
- This means we can’t test our model in the conventional way (with a test dataset)
- One way we worked around this is by having our model run real time on images/videos.
- We’ll go over the implementation on how to run it on images.

Running the model on random images (imports)

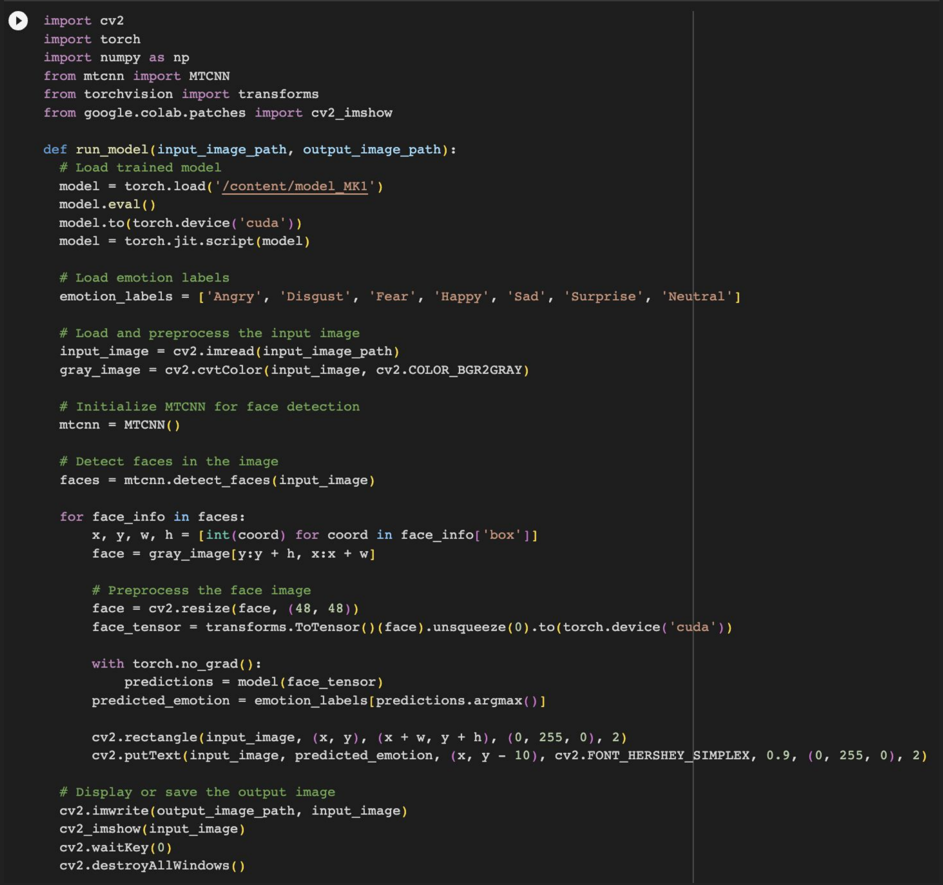


```
!pip install mtcnn
```

```
import cv2
import torch
import numpy as np
from mtcnn import MTCNN
from torchvision import transforms
from google.colab.patches import cv2_imshow
```

- In a singular code box, type in that pip install line.
- Then in a separate one after the pip install, type in all of those import statements.

Running the model on random images



```
import cv2
import torch
import numpy as np
from mtcnn import MTCNN
from torchvision import transforms
from google.colab.patches import cv2_imshow

def run_model(input_image_path, output_image_path):
  # Load trained model
  model = torch.load('/content/model_MK1')
  model.eval()
  model.to(torch.device('cuda'))
  model = torch.jit.script(model)

  # Load emotion labels
  emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']

  # Load and preprocess the input image
  input_image = cv2.imread(input_image_path)
  gray_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)

  # Initialize MTCNN for face detection
  mtcnn = MTCNN()

  # Detect faces in the image
  faces = mtcnn.detect_faces(input_image)

  for face_info in faces:
    x, y, w, h = [int(coord) for coord in face_info['box']]
    face = gray_image[y:y + h, x:x + w]

    # Preprocess the face image
    face = cv2.resize(face, (48, 48))
    face_tensor = transforms.ToTensor()(face).unsqueeze(0).to(torch.device('cuda'))

    with torch.no_grad():
      predictions = model(face_tensor)
    predicted_emotion = emotion_labels[predictions.argmax()]

    cv2.rectangle(input_image, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.putText(input_image, predicted_emotion, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

  # Display or save the output image
  cv2.imwrite(output_image_path, input_image)
  cv2_imshow(input_image)
  cv2.waitKey(0)
  cv2.destroyAllWindows()
```

- This is a lot to take in, and we won’t really be going over what is going on here.
- Copy this code into your project notebook.
- With this code, we should be able to input images into the model and see what emotion the model thinks the image has.
- Based on this, we can gauge how well our model is performing.
- Not as satisfying or helpful as using the test dataset, but at least it’s something

torch.save()

Function: torch.save(model, “filename”)

- Saves CURRENT model parameters as a file
  - Weights, biases, etc.
  - ONLY saves the parameters, doesn’t contain the definition of the model itself
- Call after training your model
- File can be loaded…

torch.load()

Function: model = torch.load(“filename”)

- Model architecture needs to be defined in the same file (doesn’t matter whether it’s imported or typed out)
- Model is “pre-trained” - the parameters that make the model accurate are loaded from the file

Ways to improve

- Better data
  - Specifically for our issue, what emotions do we need more data on?
- Improved loss functions and optimizer
  - Research more applicable options (there aren’t many)
- Tweaking hidden layers
  - Generally requires knowledge of the math behind the layers
  - Trial and error

What next?

Research ways of getting your model ready to use in an application.

First, you need to transfer everything from a notebook into code (.py) files. For a tutorial on this:

PyTorch Going Modular (There are also some other really cool tutorials in this series, on the left hand side)

Now, some easy ways to deploy:

- In a REST API with Flask
- Pytorch Lightning (I’ve never used this, but look through the docs)
- A tutorial from the same series as the Going Modular one

First Time Sign up

AI Club Website

Enjoy your week!

go.osu.edu/aiclubsignup