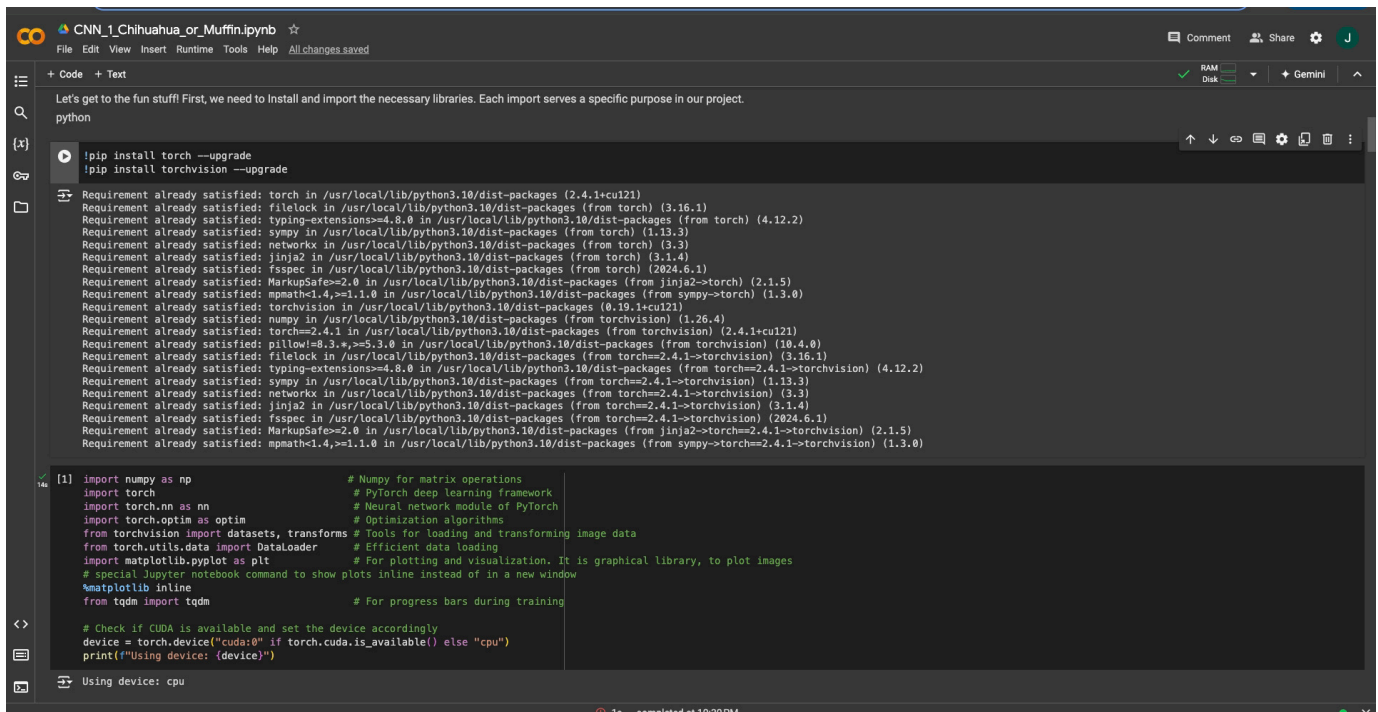


Jeffery Dirden
ITAI - 1378
October 7, 2024

L07 Reflective Journal

This was a very fun lab to partake in learning, and adjusting more on how to incorporate different datasets, debugging and correcting syntax, doing extensive research to understand how to implement everything. CNN which is a convolutional neural networks is a neural network used for processing grid data, such as images. Unlike traditional fully connected networks CNNs use spatial structure from input data making them very effective for tasks like image classification. I personally used the CIFAR-10 dataset to train my CNN model for this case. Started off by importing the required libraries needed I personally feel comfortable using numpy for matrix operations, torch as a deep learning framework, matplotlib for the graphics, if I'm not mistaken this is my first time using "tqdm" for the progress bars during training, it was very



```
Let's get to the fun stuff! First, we need to install and import the necessary libraries. Each import serves a specific purpose in our project.

python

!pip install torch --upgrade
!pip install torchvision --upgrade

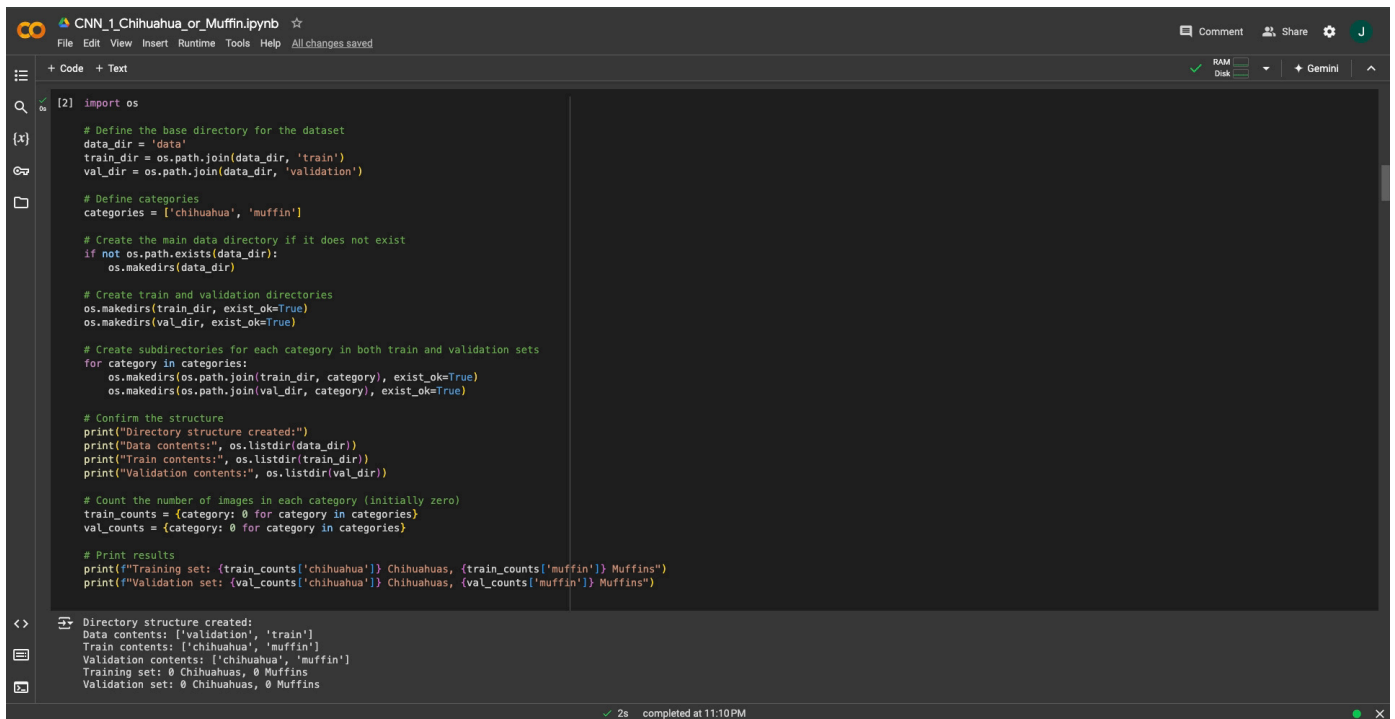
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.4.1+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2024.6.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath<1.4, >=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (0.19.1+cu121)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision) (1.26.4)
Requirement already satisfied: torch==2.4.1 in /usr/local/lib/python3.10/dist-packages (from torchvision) (2.4.1+cu121)
Requirement already satisfied: pillow>=8.3.0 in /usr/local/lib/python3.10/dist-packages (from torchvision) (10.4.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch==2.4.1->torchvision) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch==2.4.1->torchvision) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch==2.4.1->torchvision) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch==2.4.1->torchvision) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch==2.4.1->torchvision) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch==2.4.1->torchvision) (2024.6.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch==2.4.1->torchvision) (2.1.5)
Requirement already satisfied: mpmath<1.4, >=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch==2.4.1->torchvision) (1.3.0)

[1] import numpy as np                # Numpy for matrix operations
import torch                        # PyTorch deep learning framework
import torch.nn as nn              # Neural network module of PyTorch
import torch.optim as optim        # Optimization algorithms
from torchvision import datasets, transforms # Tools for loading and transforming image data
from torch.utils.data import DataLoader # Efficient data loading
import matplotlib.pyplot as plt     # For plotting and visualization. It is graphical library, to plot images
%matplotlib inline                 # special Jupyter notebook command to show plots inline instead of in a new window
from tqdm import tqdm              # For progress bars during training

# Check if CUDA is available and set the device accordingly
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")

Using device: cpu
```

interesting because as I ran the cell with tqdm operating I got to see a visual representation of the progress bar giving me an insight on where everything is at with the initialization. A problem that I experienced through this process was making sure the necessary datasets were imported to get the commands to run.



```
[2] import os

# Define the base directory for the dataset
data_dir = 'data'
train_dir = os.path.join(data_dir, 'train')
val_dir = os.path.join(data_dir, 'validation')

# Define categories
categories = ['chihuahua', 'muffin']

# Create the main data directory if it does not exist
if not os.path.exists(data_dir):
    os.makedirs(data_dir)

# Create train and validation directories
os.makedirs(train_dir, exist_ok=True)
os.makedirs(val_dir, exist_ok=True)

# Create subdirectories for each category in both train and validation sets
for category in categories:
    os.makedirs(os.path.join(train_dir, category), exist_ok=True)
    os.makedirs(os.path.join(val_dir, category), exist_ok=True)

# Confirm the structure
print("Directory structure created:")
print("Data contents:", os.listdir(data_dir))
print("Train contents:", os.listdir(train_dir))
print("Validation contents:", os.listdir(val_dir))

# Count the number of images in each category (initially zero)
train_counts = {category: 0 for category in categories}
val_counts = {category: 0 for category in categories}

# Print results
print(f"Training set: {train_counts['chihuahua']} Chihuahuas, {train_counts['muffin']} Muffins")
print(f"Validation set: {val_counts['chihuahua']} Chihuahuas, {val_counts['muffin']} Muffins")
```

Directory structure created:
Data contents: ['validation', 'train']
Train contents: ['chihuahua', 'muffin']
Validation contents: ['chihuahua', 'muffin']
Training set: 0 Chihuahuas, 0 Muffins
Validation set: 0 Chihuahuas, 0 Muffins

This is where I began data preparation the main problem that I faced was defining the dataset was the main issue that I faced during this stage, I was able to overcome this problem by understanding how to properly intertwine this issue by understanding how paths work. I overviewed the code to understand that the 2 main categories were chihuahua and muffin, the purpose is to understand the difference between both. Based on the final display I understand the importance of CNNs and its abilities to decipher the difference between 2 different images being able to train itself and adapt to current data is very impressive. As far as ethical considerations are concerned I do feel that data is very accessible, but it's also beneficial to the future of how things will go.

