Final Report CSE164

Ethan Wong 1860406

Model:
While building the model, I tested numerous other pretrained models, such as Resnet18, Resnet50, Vit, and Densenet without pretrained weights. My initial findings indicated that Resnet18 or 50 was the most accurate for the model. In terms of efficiency, Resnet18 was much faster due to having fewer layers. As a result, I decided to continue with ResNet18 to train the labeled images.

Some data augmentations that I applied onto the labeled images include horizontal flips, resizing the images to (224,224), and normalization. I didn't add anything to the model besides the final classification layer since through testing the loss appeared to do better without any additional layers. As for my criterion, I applied cross entropy loss and used the Adam optimizer with a learning rate of 0.0001. The reason for this learning rate is because I found the most success with this learning rate, as well as that was what the baseline model used to train its images.

After training the labeled images for 20-30 epochs, or until I saw divergence between train and validation loss, I then started the semi supervised implementation.

The pipeline for semi supervised training was pretty straightforward. I would generate pseudo labels based on the initial model by accessing the unlabeled images, and only save unlabeled images that the model says are 85-95% confident it is a certain label. I would iterate 1 by 1 and depending on the loss, I would decrease and increase the confidence level. I then saved the list of pseudo labels, and combined the pseudo labels with my training dataset. Then, I retrained my model and repeated this process until I saw the loss converge. In order to avoid duplicates, I saved the pseudo labeled images in a csv file, and appended the images into a combined data loader, which is what I would train on the next round. I would do this cycle until I saw my implementation cap off at around 0.57.

To evaluate the performance of our model, I used accuracy as well as the training and validation loss to see if our model was overfitting or not. Graphs were helpful in visualizing whether or not our model is improving, or if it is overfitting or underfitting. After training my model on the pseudo labeled and labeled images, I used the test file to test our model for its real accuracy.

Environment:
Here are the libraries that we used for our model:

```
import pandas as pd
from PIL import Image
import os
import torch
from torchvision import models
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, ConcatDataset
import torch.optim as optim
import torch.nn.functional as F
import torch.nn as nn
import numpy as np
from torchvision.io import read_image
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
```

Usage:

1. Make sure the dataset directory is structured as follows: The train folder should have 2 folders, 1 labeled and 1 unlabeled.

```
├── train
│   ├── labeled
│   │   ├── 00000.jpg
│   │   ├── ...
│   │   ├── 09853.jpg
│   ├── unlabeled
│   │   ├── 09854.jpg
│   │   ├── ...
│   │   ├── 32848.jpg
├── test
│   ├── 32849.jpg
│   ├── ...
│   ├── 41061.jpg
├── categories.csv
├── train_labeled.csv
└── sample_submission.csv
```

2. Ensure all dependencies are installed and the python version is at least 3.10. Also ensure files are in the same directory as the models
3. To run the models, open using Jupyter Notebook and run all cells.
4. The training files is final.ipynb
5. Output will be saved in a folder called “test_predictions.csv”

Weight:
🗀 164 Final Weight