IT1244: Traffic Sign Classifier Using Neural Networks

By: Evan Ng, Kyi Wai, Mario Hanzel, Sydney Sim

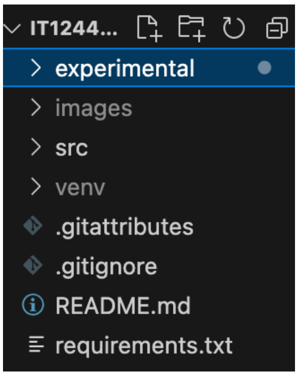
1) Firstly, Install all requirements from requirements.txt file

```
pip install -r requirements.txt
```

2) Download Models from google drive

https://drive.google.com/drive/folders/1zAgRTRNzlkOubOKzDlF99UdJUOqatqKq?usp=sharing

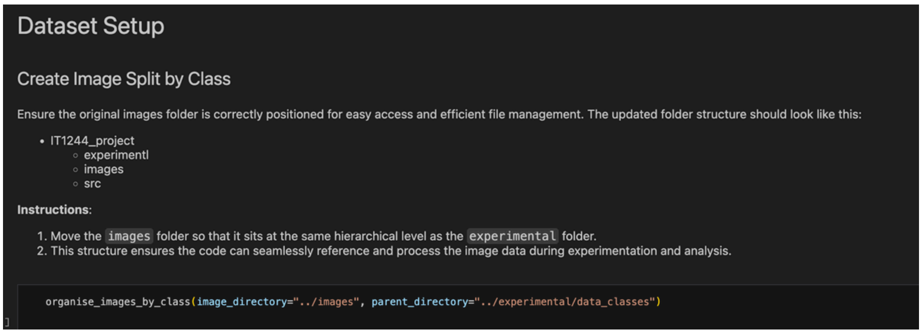
3) Pull Images Dataset into Parent Directory, example folder structure below:



```
IT1244...
  > experimental
  > images
  > src
  > venv
    .gitattributes
    .gitignore
    README.md
    requirements.txt
```

4) Running of cells:

- Run ALL cells in the Dataset Setup and Train-Test Split sections



Dataset Setup

Create Image Split by Class

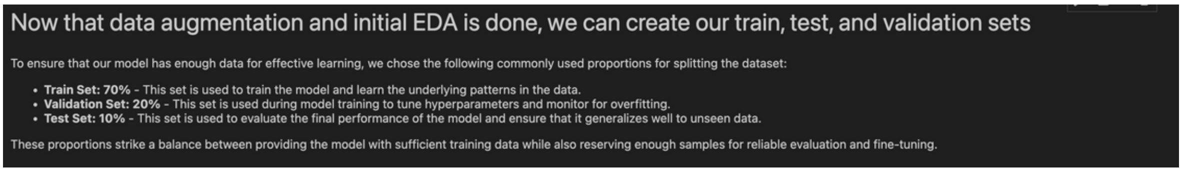
Ensure the original images folder is correctly positioned for easy access and efficient file management. The updated folder structure should look like this:

- IT1244_project
  - experimentl
  - images
  - src

Instructions:

1. Move the `images` folder so that it sits at the same hierarchical level as the `experimental` folder.
2. This structure ensures the code can seamlessly reference and process the image data during experimentation and analysis.

```
organise_images_by_class(image_directory="../images", parent_directory="../experimental/data_classes")
```



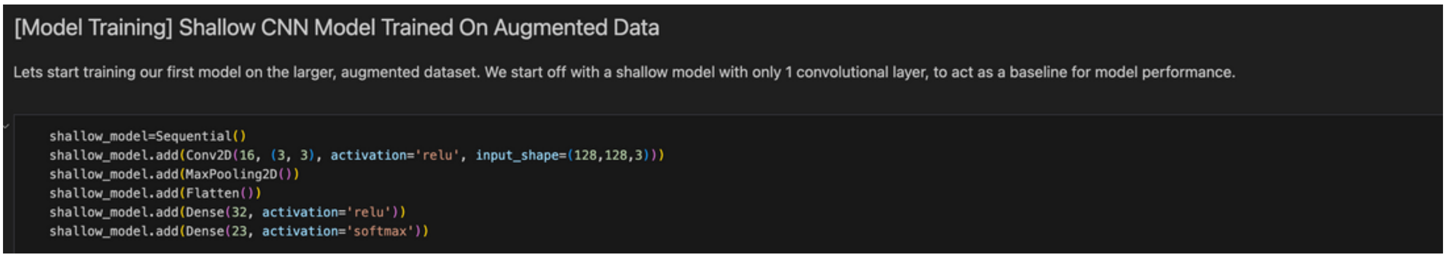
Now that data augmentation and initial EDA is done, we can create our train, test, and validation sets

To ensure that our model has enough data for effective learning, we chose the following commonly used proportions for splitting the dataset:

- **Train Set: 70%** - This set is used to train the model and learn the underlying patterns in the data.
- **Validation Set: 20%** - This set is used during model training to tune hyperparameters and monitor for overfitting.
- **Test Set: 10%** - This set is used to evaluate the final performance of the model and ensure that it generalizes well to unseen data.

These proportions strike a balance between providing the model with sufficient training data while also reserving enough samples for reliable evaluation and fine-tuning.

- For the Model Training and evaluation sections, run all except the cells marked [Model Training] (example shown below)

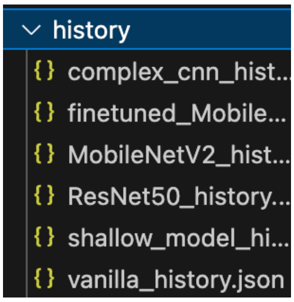


[Model Training] Shallow CNN Model Trained On Augmented Data

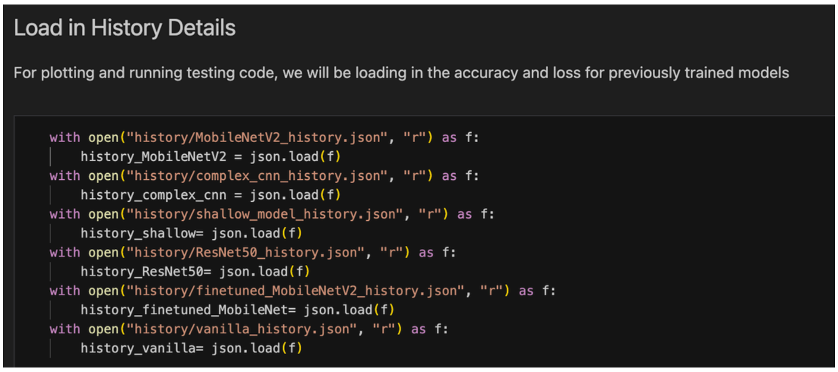
Lets start training our first model on the larger, augmented dataset. We start off with a shallow model with only 1 convolutional layer, to act as a baseline for model performance.

```
shallow_model=Sequential()
shallow_model.add(Conv2D(16, (3, 3), activation='relu', input_shape=(128,128,3)))
shallow_model.add(MaxPooling2D())
shallow_model.add(Flatten())
shallow_model.add(Dense(32, activation='relu'))
shallow_model.add(Dense(23, activation='softmax'))
```

- All plots (training history data) are saved under history folder, and should be initialised in when running the notebook



```
history
  {} complex_cnn_hist...
  {} finetuned_Mobile...
  {} MobileNetV2_hist...
  {} ResNet50_history...
  {} shallow_model_hi...
  {} vanilla_history.json
```

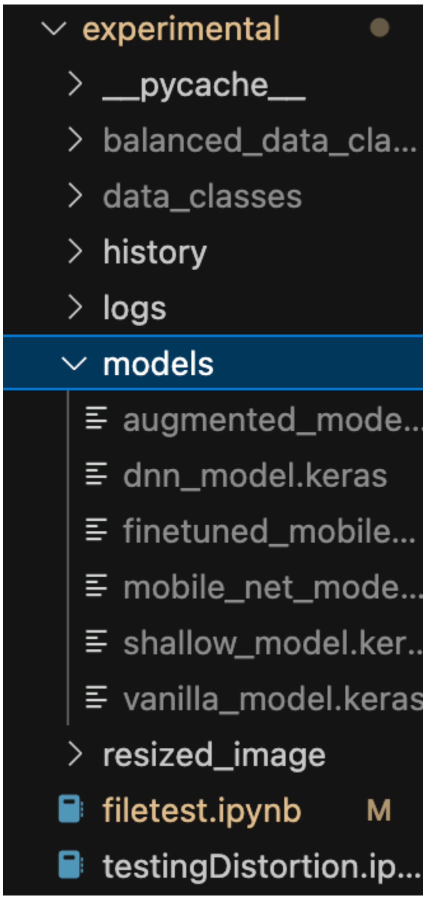


Load in History Details

For plotting and running testing code, we will be loading in the accuracy and loss for previously trained models

```
with open("history/MobileNetV2_history.json", "r") as f:
    history_MobileNetV2 = json.load(f)
with open("history/complex_cnn_history.json", "r") as f:
    history_complex_cnn = json.load(f)
with open("history/shallow_model_history.json", "r") as f:
    history_shallow= json.load(f)
with open("history/ResNet50_history.json", "r") as f:
    history_ResNet50= json.load(f)
with open("history/finetuned_MobileNetV2_history.json", "r") as f:
    history_finetuned_MobileNet= json.load(f)
with open("history/vanilla_history.json", "r") as f:
    history_vanilla= json.load(f)
```

- All Models should be kept in a folder “models” and placed inside the “experimental” folder



```
experimental
  __pycache__
  balanced_data_cla...
  data_classes
  history
  logs
  models
    augmented_mode...
    dnn_model.keras
    finetuned_mobile...
    mobile_net_mode...
    shallow_model.ker..
    vanilla_model.keras
  resized_image
  filetest.ipynb      M
  testingDistortion.ip...
```

THANK YOU FOR YOUR TIME!