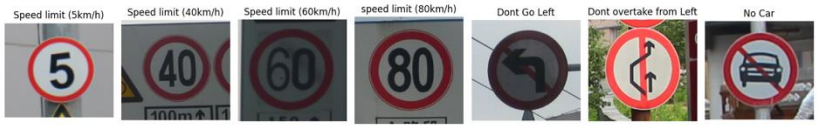
Assignment #2: Traffic Sign Classification

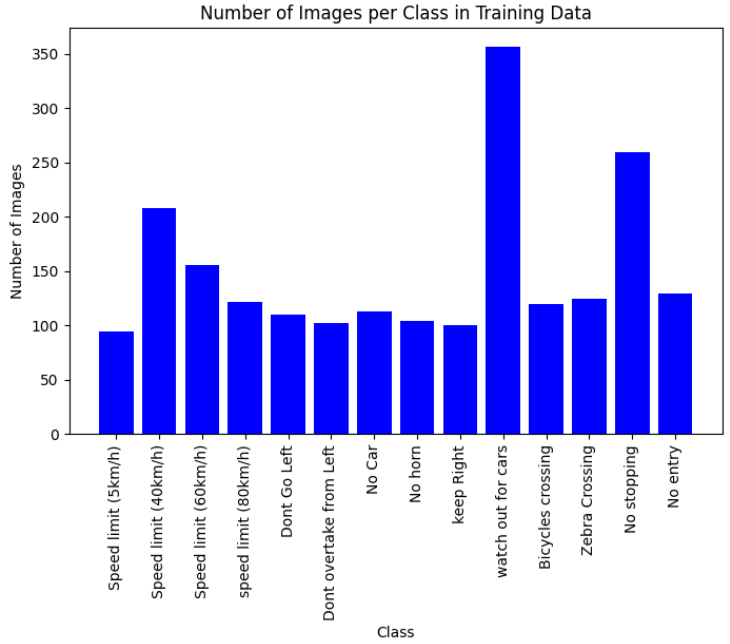
In this project we use a dataset from Kaggle which has 14 classes of traffic sign.



Train: **2095 images**
Test: **531 images**

- **Dataset Infomation**:

	ClassID	Count	Name
0	9	356	watch out for cars
1	0	94	Speed limit (5km/h)
2	11	124	Zebra Crossing
3	7	104	No horn
4	6	113	No Car
5	1	208	Speed limit (40km/h)
6	10	120	Bicycles crossing
7	8	100	keep Right
8	4	110	Dont Go Left
9	3	121	speed limit (80km/h)
10	12	259	No stopping
11	2	155	Speed limit (60km/h)
12	13	129	No entry
13	5	102	Dont overtake from Left



Hyperparameter

Epoch	10
Opimizer	Adam
Loss Function	**Categorical Cross Entropy**
Learning Rate	0.0001
Metrics	Accuracy

Assignment #2: Traffic Sign Classification

In this project we use a **VGG19** pretrain model to fine tune on our dataset on 2 type comparation:

- Freeze all layers except the last 10 layers
- freeze all layers except the last 5 layers

```
#Load model
base_model = VGG19(weights = "imagenet", include_top=False, input_shape = (224,224, 3), pooling='avg')
base_model.summary()
```

Classification Head

```
# use “get_layer” method to save the last layer of the network
last_layer = base_model.get_layer('global_average_pooling2d')
last_output = last_layer.output
x = Dense(num_classes, activation='softmax', name='softmax')(last_output)

# instantiate a new_model using keras’s Model class
model_v1 = Model(inputs=base_model.input, outputs=x)
```

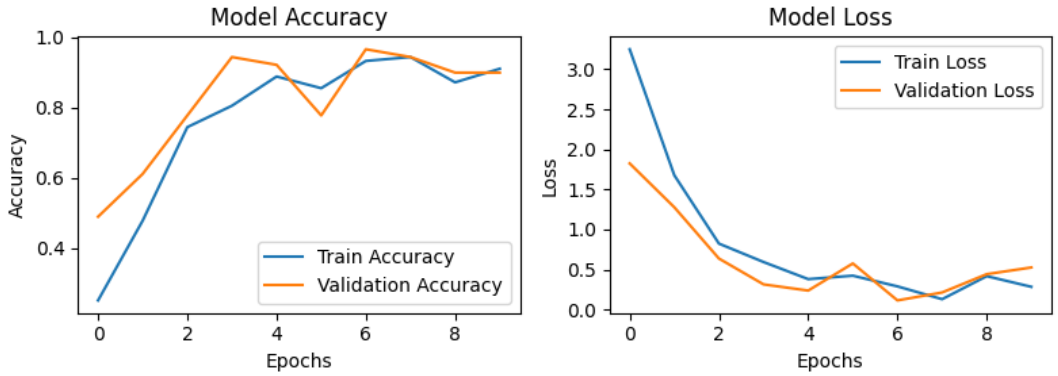
Assignment #2: Traffic Sign Classification

Fine Tune Level 1 (Keep only last 10 layers):

```
# iterate through its layers and lock them to make them not trainable with this code
for layer in base_model.layers[:-10]:
    layer.trainable = False

base_model.summary()
```

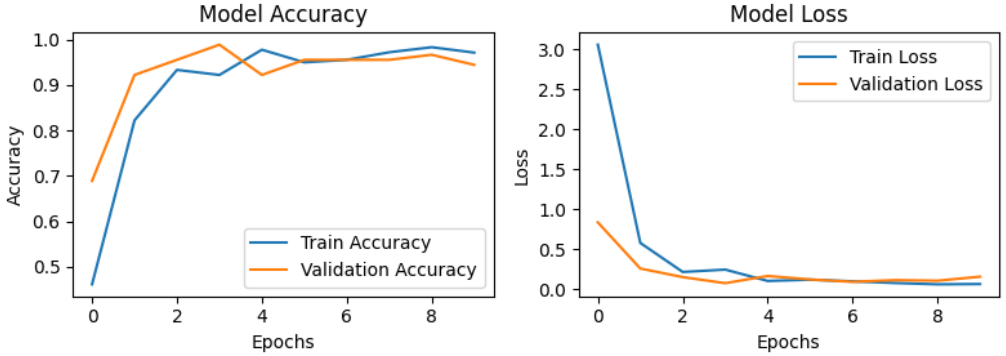
```
=================================================================
Total params: 20031566 (76.41 MB)
Trainable params: 16525838 (63.04 MB)
Non-trainable params: 3505728 (13.37 MB)
_________________________________________________________________
```



Fine Tune Level 2 (Keep only last 5 layers):

```
# Freeze lower 5 layers
for layer in base_model.layers[:-5]:
    layer.trainable = False
```

```
=================================================================
Total params: 20031566 (76.41 MB)
Trainable params: 7086606 (27.03 MB)
Non-trainable params: 12944960 (49.38 MB)
_________________________________________________________________
```

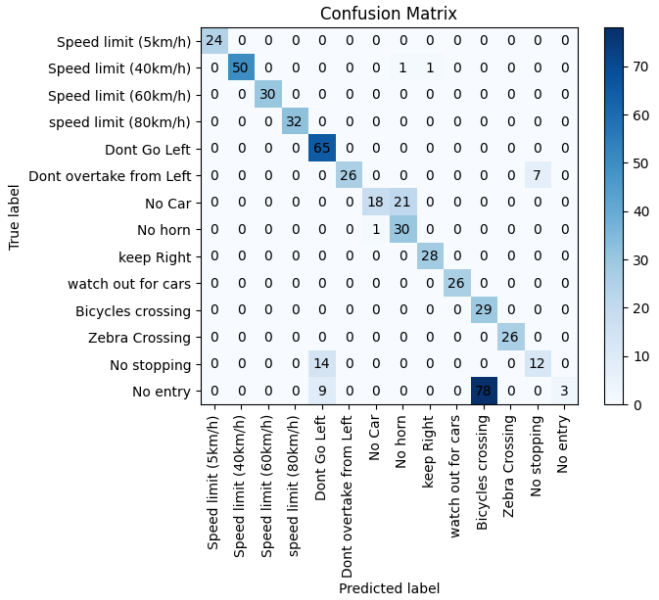


Assignment #2: Traffic Sign Classification

Conclusion

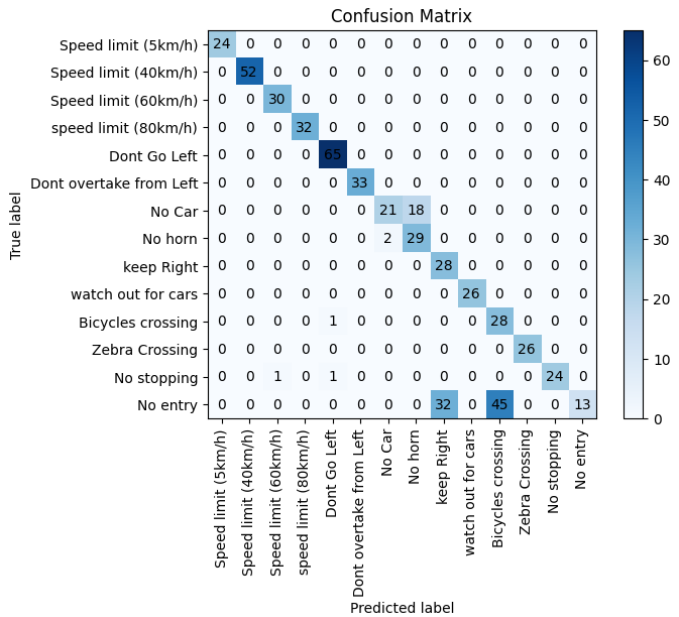
- When freezing all layers **except the last 10**, the model achieved an accuracy of **75%** at tuning level 1. However, by only freezing all layers **except the last 5**, the accuracy improved significantly to **81%** at tuning level 2.

Fine-Tune Level 1 (fix lower 10)



Testing loss: **0.6343**
Testing accuracy: **0.7514**

Fine-Tune Level 2 (fix lower 5)



Testing loss: **0.3916**
Testing accuracy: **0.8117**

Conclusion

- In the first project, which is centered around sign language classification, the VGG16 model was employed. Within this context, tuning Level 1 (freeze all layers except the last 10 layers) yielded better results than tuning Level 2 (freeze all layers except the last 5 layers).

- Conversely, in the second project focusing on traffic sign classification, the VGG19 model was used, and here, tuning Level 2 outperformed tuning Level 1.