MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

MACHINE LEARNING

SUBMITTED BY : Ankit Kumar

SCHOLAR NO: ~201112471

SECTION: ~ CSE 3

SUBMITTED TO: ~ **RAJESH PATERIYA**

Q1. Use pretrained CNN, RESNET 50 for development of traffic sign classification system. Use GTSRB dataset.

Code:

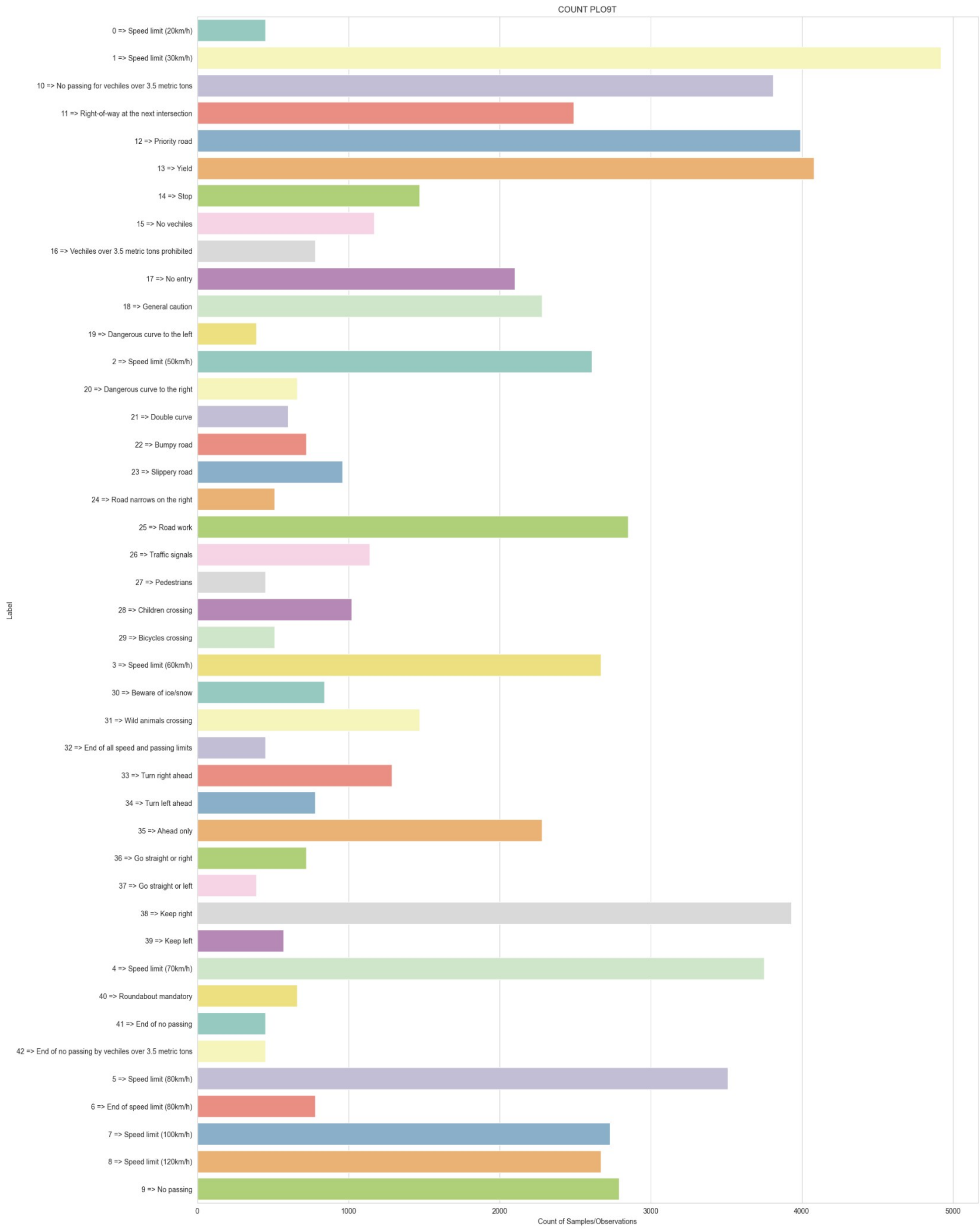
```
import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style('whitegrid')
from tensorflow.keras.utils import plot_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import tensorflow as tf
print('TensoFlow Version: ', tf.__version__)
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten,
GlobalAveragePooling2D, BatchNormalization,
Dropout
from tensorflow.keras.applications.resnet import ResNet50
from tensorflow.keras.callbacks import ModelCheckpoint,
EarlyStopping, ReduceLROnPlateau,
CSVLogger
path = '/kaggle/input/traffic-signs-classification'
lab =
pd.read_csv('/kaggle/input/traffic-signs-classification/labels.csv')
d = dict()
class_labels = dict()for dirs in os.listdir(path + '/myData'):
count = len(os.listdir(path+'/myData/'+dirs))
d[dirs+' => '+lab[lab.ClassId == int(dirs)].values[0][1]] = count
class_labels[int(dirs)] = lab[lab.ClassId == int(dirs)].values[0][1]
plt.figure(figsize = (20, 50))
sns.barplot(y = list(d.keys()), x = list(d.values()), palette =
'Set3')
plt.ylabel('Label')
plt.xlabel('Count of Samples/Observations')
img_rows, img_cols = 32, 32
img_channels = 3
```

```
nb_classes = len(class_labels.keys())
datagen = ImageDataGenerator()
data = datagen.flow_from_directory('/kaggle/input/traffic-signs-
classification/myData',
target_size=(32, 32),
batch_size=73139,
class_mode='categorical',
shuffle=True )
X , y = data.next()
print(f"Data Shape :{X.shape}\nLabels shape :{y.shape}")
fig, axes = plt.subplots(10,10, figsize=(18,18))
for i,ax in enumerate(axes.flat):
r = np.random.randint(X.shape[0])
ax.imshow(X[r].astype('uint8'))
ax.grid(False)
ax.axis('off')
ax.set_title('Label: '+str(np.argmax(y[r])))
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=11)
print("Train Shape: {}\nTest Shape : {}".format(X_train.shape,
X_test.shape))resnet
=
ResNet50(weights=
(img_rows,img_cols,img_channels))
None,
include_top=False,
input_shape=
x = resnet.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
predictions = Dense(nb_classes, activation= 'softmax')(x)
model = Model(inputs = resnet.input, outputs = predictions)
model.summary()
plot_model(model, show_layer_names=True, show_shapes =True,
to_file='model.png', dpi=350)
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model_check
=
ModelCheckpoint('best_model.h5',
save_best_only=True, mode='max')
monitor='val_accuracy',
verbose=0,
```

```
early = EarlyStopping(monitor='val_accuracy', min_delta=0,
patience=5, verbose=0, mode='max',
restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2,
patience=5, min_lr=0.001)
csv_logger = CSVLogger('train_log.csv', separator=',')
n_epochs = 30
history = model.fit(X_train, y_train, batch_size = 32, epochs =
n_epochs, verbose = 1,
validation_data = (X_test, y_test), callbacks = [model_check, early,
reduce_lr, csv_logger])
model.save('TSC_model.h5')
loss, acc = model.evaluate(X_test, y_test)
print('Accuracy: ', acc, '\nLoss
: ', loss)
q = len(list(history.history['loss']))
plt.figure(figsize=(12, 6))
sns.lineplot(x = range(1, 1+q), y = history.history['accuracy'],
label = 'Accuracy')
sns.lineplot(x = range(1, 1+q), y = history.history['loss'], label =
'Loss')
plt.xlabel('#epochs')
plt.ylabel('Training')
plt.legend();plt.figure(figsize=(12, 6))
sns.lineplot(x = range(1, 1+q), y = history.history['accuracy'],
label = 'Train')
sns.lineplot(x = range(1, 1+q), y = history.history['val_accuracy'],
label = 'Validation')
plt.xlabel('#epochs')
plt.ylabel('Accuracy') plt.legend()
plt.figure(figsize=(12, 6))
sns.lineplot(x = range(1, 1+q), y = history.history['loss'], label =
'Train')
sns.lineplot(x = range(1, 1+q), y = history.history['val_loss'],
label = 'Validation')
plt.xlabel('#epochs')
plt.ylabel('Loss') plt.legend()
pred = np.argmax(model.predict(X_test), axis = 1)
labels = [class_labels[i] for i in range(43)]
print(classification_report(np.argmax(y_test, axis = 1), pred,
target_names = labels))
cmat = confusion_matrix(np.argmax(y_test, axis=1), pred)
plt.figure(figsize=(16,16))
```

```
sns.heatmap(cmat, annot = True, cbar = False, cmap='Paired',
fmt="d", xticklabels=labels,
yticklabels=labels);
classwise_acc = cmat.diagonal()/cmat.sum(axis=1) * 100
cls_acc = pd.DataFrame({'Class_Label':[class_labels[i] for
classwise_acc.tolist()}, columns = ['Class_Label', 'Accuracy'])
cls_acc.style.format({"Accuracy":
color='tomato')
i
in
range(43)],
'Accuracy':
"{:,.2f}",}).hide_index().bar(subset=["Accuracy"],
fig, axes = plt.subplots(5,5, figsize=(18,18))
for i,ax in enumerate(axes.flat):
r = np.random.randint(X_test.shape[0])
ax.imshow(X_test[r].astype('uint8'))
ax.grid(False)
ax.axis('off')
ax.set_title('Original:
{}
Predicted:
np.argmax(model.predict(X_test[r].reshape(1, 32, 32, 3)))))
{}'.format(np.argmax(y_test[r]),
```

Output:


COUNT PLO9T
0 => Speed limit (20km/h)
1 => Speed limit (30km/h)
10 => No passing for vechiles over 3.5 metric tons
11 => Right-of-way at the next intersection
12 => Priority road
13 => Yield
14 => Stop
15 => No vechiles
16 => Vechiles over 3.5 metric tons prohibited
17 => No entry
18 => General caution
19 => Dangerous curve to the left
2 => Speed limit (50km/h)
20 => Dangerous curve to the right
21 => Double curve
22 => Bumpy road
23 => Slippery road
24 => Road narrows on the right
25 => Road work
26 => Traffic signals
27 => Pedestrians
28 => Children crossing
29 => Bicycles crossing
3 => Speed limit (60km/h)
30 => Beware of ice/snow
31 => Wild animals crossing
32 => End of all speed and passing limits
33 => Turn right ahead
34 => Turn left ahead
35 => Ahead only
36 => Go straight or right
37 => Go straight or left
38 => Keep right
39 => Keep left
4 => Speed limit (70km/h)
40 => Roundabout mandatory
41 => End of no passing
42 => End of no passing by vechiles over 3.5 metric tons
5 => Speed limit (80km/h)
6 => End of speed limit (80km/h)
7 => Speed limit (100km/h)
8 => Speed limit (120km/h)
9 => No passing
Label
0
1000
2000
3000
4000
5000
Count of Samples/Observations

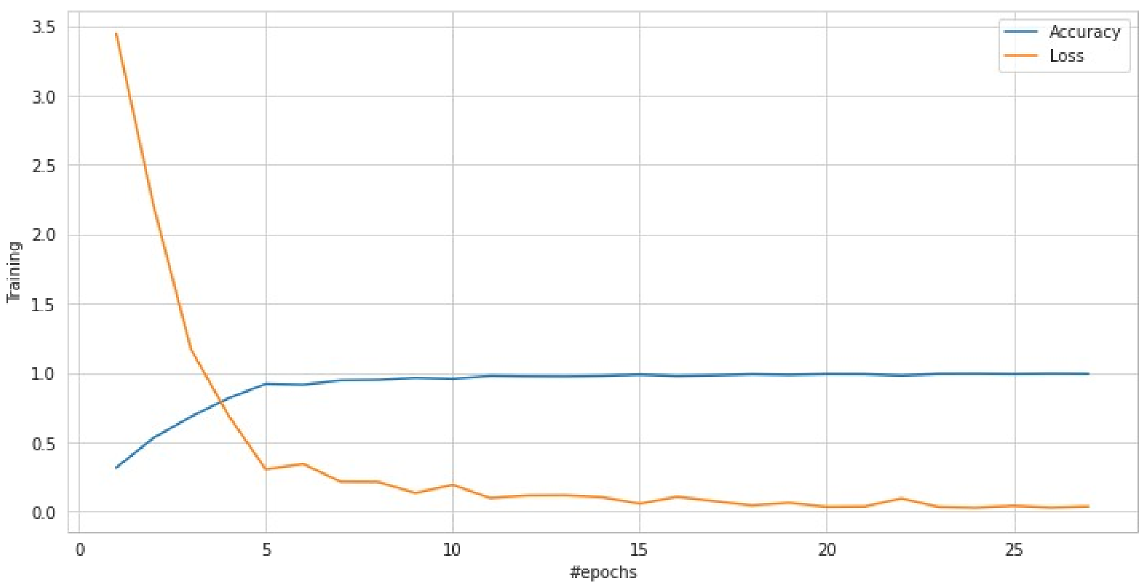

Label: 38
Label: 27
Label: 31
Label: 7
Label: 5
Label: 41
Label: 38
Label: 34
Label: 2
Label: 39
Label: 16
Label: 1
Label: 11
Label: 1
Label: 23
Label: 7
Label: 25
Label: 29
Label: 35
Label: 12
Label: 19
Label: 23
Label: 25
Label: 15
Label: 34
Label: 38
Label: 41
Label: 29
Label: 1
Label: 15
Label: 41
Label: 26
Label: 12
Label: 2
Label: 38
Label: 4
Label: 2
Label: 40
Label: 38
Label: 16
Label: 34
Label: 38
Label: 26
Label: 2
Label: 28
Label: 41
Label: 21
Label: 6
Label: 18
Label: 34
Label: 9
Label: 9
Label: 13
Label: 34
Label: 21
Label: 3
Label: 30
Label: 35
Label: 19
Label: 25
Label: 5
Label: 1
Label: 27
Label: 23
Label: 42
Label: 38
Label: 39
Label: 29
Label: 32
Label: 41
Label: 3
Label: 1
Label: 41
Label: 33
Label: 19
Label: 40
Label: 7
Label: 18
Label: 13
Label: 38
Label: 9
Label: 33
Label: 39
Label: 12
Label: 8
Label: 32
Label: 7
Label: 4
Label: 29
Label: 2
Label: 23
Label: 5
Label: 25
Label: 4
Label: 29
Label: 0
Label: 16
Label: 10
Label: 41
Label: 3

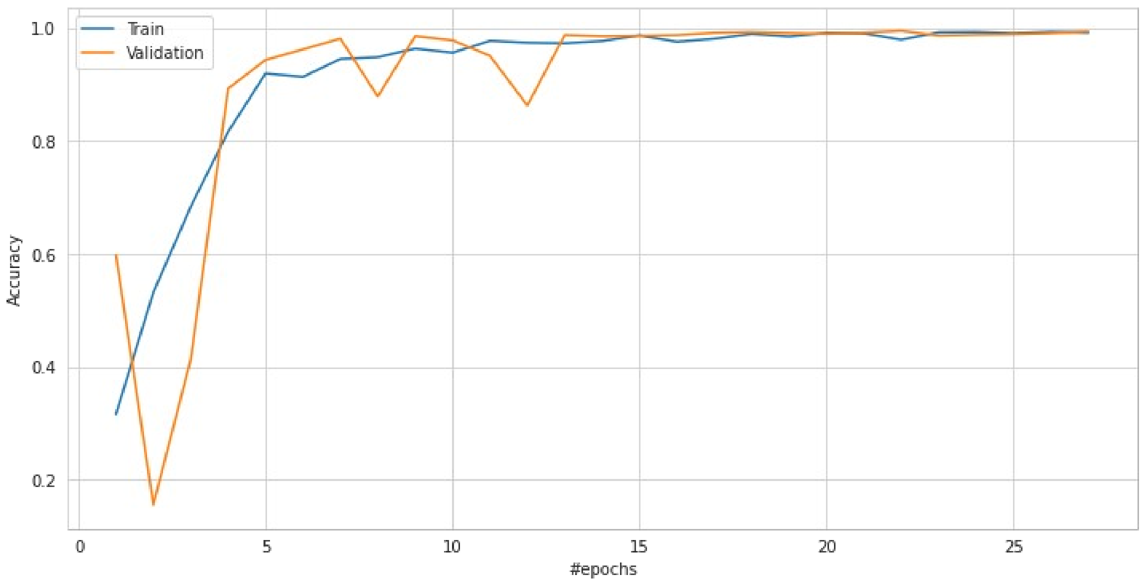
```
Model: "model"
____________________________________________________________________________________________
______
Layer (type)                    Output Shape          Param #     Connected to
============================================================================================
======
input_1 (InputLayer)            [(None, 32, 32, 3)]   0
____________________________________________________________________________________________
______
conv1_pad (ZeroPadding2D)       (None, 38, 38, 3)     0           input_1[0][0]
____________________________________________________________________________________________
______
conv1_conv (Conv2D)             (None, 16, 16, 64)    9472        conv1_pad[0][0]
____________________________________________________________________________________________
______
conv1_bn (BatchNormalization)   (None, 16, 16, 64)    256         conv1_conv[0][0]
____________________________________________________________________________________________
______
conv1_relu (Activation)         (None, 16, 16, 64)    0           conv1_bn[0][0]
____________________________________________________________________________________________
______
pool1_pad (ZeroPadding2D)       (None, 18, 18, 64)    0           conv1_relu[0][0]
____________________________________________________________________________________________
______
pool1_pool (MaxPooling2D)       (None, 8, 8, 64)      0           pool1_pad[0][0]
____________________________________________________________________________________________
______
conv2_block1_1_conv (Conv2D)    (None, 8, 8, 64)      4160        pool1_pool[0][0]
____________________________________________________________________________________________
______
conv2_block1_1_bn (BatchNormali (None, 8, 8, 64)      256         conv2_block1_1_conv[0][0]
____________________________________________________________________________________________
______
```

```
conv5_block3_2_relu (Activation (None, 1, 1, 512)    0           conv5_block3_2_bn[0][0]
____________________________________________________________________________________________
______
conv5_block3_3_conv (Conv2D)    (None, 1, 1, 2048)   1050624     conv5_block3_2_relu[0][0]
____________________________________________________________________________________________
______
conv5_block3_3_bn (BatchNormali (None, 1, 1, 2048)   8192        conv5_block3_3_conv[0][0]
____________________________________________________________________________________________
______
conv5_block3_add (Add)          (None, 1, 1, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
____________________________________________________________________________________________
______
conv5_block3_out (Activation)   (None, 1, 1, 2048)   0           conv5_block3_add[0][0]
____________________________________________________________________________________________
______
global_average_pooling2d (Globa (None, 2048)         0           conv5_block3_out[0][0]
____________________________________________________________________________________________
______
dropout (Dropout)               (None, 2048)         0           global_average_pooling2d[0]
[0]
____________________________________________________________________________________________
______
dense (Dense)                   (None, 43)           88107       dropout[0][0]
============================================================================================
======
Total params: 23,675,819
Trainable params: 23,622,699
Non-trainable params: 53,120
____________________________________________________________________________________________
```

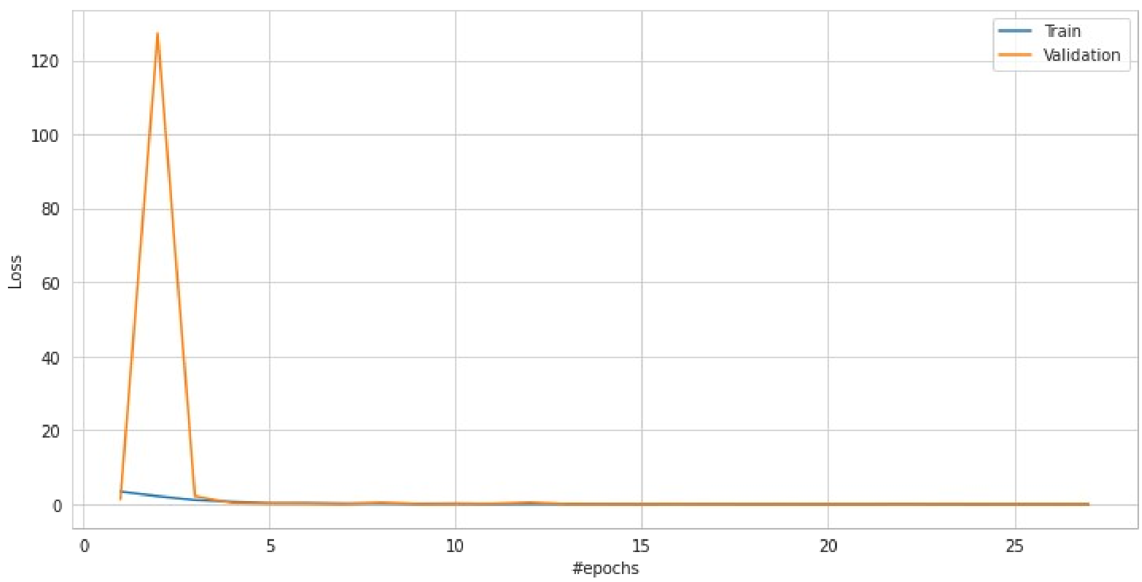
```
Train on 58511 samples, validate on 14628 samples
Epoch 1/50
58511/58511 [==============================] - 88s 2ms/sample - loss: 4.4293 - accuracy: 0.0
942 - val_loss: 4.9793 - val_accuracy: 0.0792
Epoch 2/50
58511/58511 [==============================] - 76s 1ms/sample - loss: 4.0415 - accuracy: 0.0
882 - val_loss: 7.5414 - val_accuracy: 0.0616
Epoch 3/50
58511/58511 [==============================] - 78s 1ms/sample - loss: 3.7661 - accuracy: 0.0
968 - val_loss: 4.3542 - val_accuracy: 0.1265
Epoch 4/50
58511/58511 [==============================] - 77s 1ms/sample - loss: 3.1627 - accuracy: 0.1
917 - val_loss: 15.9657 - val_accuracy: 0.3561
Epoch 5/50
58511/58511 [==============================] - 77s 1ms/sample - loss: 1.4248 - accuracy: 0.6
216 - val_loss: 0.5162 - val_accuracy: 0.8434
Epoch 6/50
58511/58511 [==============================] - 78s 1ms/sample - loss: 0.6379 - accuracy: 0.8
356 - val_loss: 0.7554 - val_accuracy: 0.9290
Epoch 7/50
58511/58511 [==============================] - 77s 1ms/sample - loss: 0.4123 - accuracy: 0.8
824 - val_loss: 0.5858 - val_accuracy: 0.9354
Epoch 8/50
58511/58511 [==============================] - 76s 1ms/sample - loss: 0.3387 - accuracy: 0.9
005 - val_loss: 3.7539 - val_accuracy: 0.9286
Epoch 9/50
58511/58511 [==============================] - 77s 1ms/sample - loss: 0.1734 - accuracy: 0.9
496 - val_loss: 0.3902 - val_accuracy: 0.9491
```

```
Epoch 10/50
58511/58511 [==============================] - 78s 1ms/sample - loss: 0.1091 - accuracy: 0.9
687 - val_loss: 1.4289 - val_accuracy: 0.9701
Epoch 11/50
58511/58511 [==============================] - 77s 1ms/sample - loss: 0.0960 - accuracy: 0.9
728 - val_loss: 0.1849 - val_accuracy: 0.9788
Epoch 12/50
58511/58511 [==============================] - 76s 1ms/sample - loss: 0.2142 - accuracy: 0.9
405 - val_loss: 0.0445 - val_accuracy: 0.9867
Epoch 13/50
58511/58511 [==============================] - 77s 1ms/sample - loss: 0.0793 - accuracy: 0.9
780 - val_loss: 0.0440 - val_accuracy: 0.9869
Epoch 14/50
58511/58511 [==============================] - 76s 1ms/sample - loss: 0.0458 - accuracy: 0.9
870 - val_loss: 0.0367 - val_accuracy: 0.9890
Epoch 15/50
58511/58511 [==============================] - 75s 1ms/sample - loss: 0.0725 - accuracy: 0.9
801 - val_loss: 0.0623 - val_accuracy: 0.9817
Epoch 16/50
58511/58511 [==============================] - 76s 1ms/sample - loss: 0.0464 - accuracy: 0.9
870 - val_loss: 0.0252 - val_accuracy: 0.9926
Epoch 17/50
58511/58511 [==============================] - 76s 1ms/sample - loss: 0.0464 - accuracy: 0.9
878 - val_loss: 0.0324 - val_accuracy: 0.9891
Epoch 18/50
58511/58511 [==============================] - 76s 1ms/sample - loss: 0.0429 - accuracy: 0.9
889 - val_loss: 0.0152 - val_accuracy: 0.9950
Epoch 19/50
```

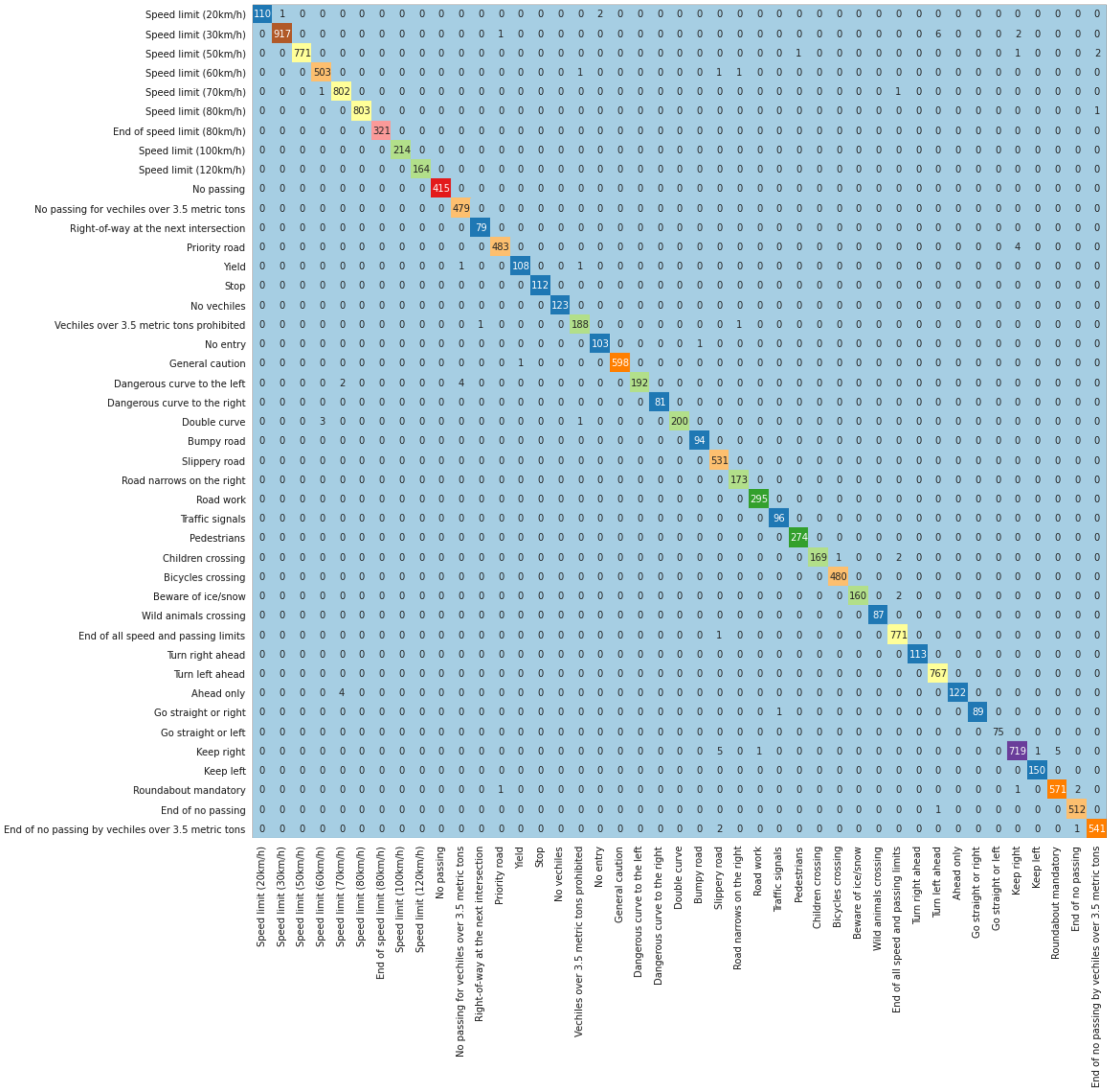




Train
Validation
Accuracy
#epochs



Train
Validation
Loss
#epochs



Speed limit (20km/h)
Speed limit (30km/h)
Speed limit (50km/h)
Speed limit (60km/h)
Speed limit (70km/h)
Speed limit (80km/h)
End of speed limit (80km/h)
Speed limit (100km/h)
Speed limit (120km/h)
No passing
No passing for vechiles over 3.5 metric tons
Right-of-way at the next intersection
Priority road
Yield
Stop
No vechiles
Vechiles over 3.5 metric tons prohibited
No entry
General caution
Dangerous curve to the left
Dangerous curve to the right
Double curve
Bumpy road
Slippery road
Road narrows on the right
Road work
Traffic signals
Pedestrians
Children crossing
Bicycles crossing
Beware of ice/snow
Wild animals crossing
End of all speed and passing limits
Turn right ahead
Turn left ahead
Ahead only
Go straight or right
Go straight or left
Keep right
Keep left
Roundabout mandatory
End of no passing
End of no passing by vechiles over 3.5 metric tons

Class_Label	Accuracy
Speed limit (20km/h)	97.35
Speed limit (30km/h)	99.03
Speed limit (50km/h)	99.48
Speed limit (60km/h)	99.41
Speed limit (70km/h)	99.75
Speed limit (80km/h)	99.88
End of speed limit (80km/h)	100.00
Speed limit (100km/h)	100.00
Speed limit (120km/h)	100.00
No passing	100.00
No passing for vechiles over 3.5 metric tons	100.00
Right-of-way at the next intersection	100.00
Priority road	99.18
Yield	98.18
Stop	100.00
No vechiles	100.00
Vechiles over 3.5 metric tons prohibited	98.95
No entry	99.04
General caution	99.83
Dangerous curve to the left	96.97
Dangerous curve to the right	100.00
Double curve	98.04
Bumpy road	100.00

Slippery road	100.00
Road narrows on the right	100.00
Road work	100.00
Traffic signals	100.00
Pedestrians	100.00
Children crossing	98.26
Bicycles crossing	100.00
Beware of ice/snow	98.77
Wild animals crossing	100.00
End of all speed and passing limits	99.87
Turn right ahead	100.00
Turn left ahead	100.00
Ahead only	96.83
Go straight or right	98.89
Go straight or left	100.00
Keep right	98.36
Keep left	100.00
Roundabout mandatory	99.30
End of no passing	99.81
End of no passing by vechiles over 3.5 metric tons	99.45


Original: 36 Predicted: 36
Original: 21 Predicted: 21
Original: 9 Predicted: 9
Original: 30 Predicted: 30
Original: 38 Predicted: 38
Original: 32 Predicted: 32
Original: 40 Predicted: 40
Original: 35 Predicted: 35
Original: 34 Predicted: 34
Original: 4 Predicted: 4
Original: 1 Predicted: 1
Original: 40 Predicted: 40
Original: 1 Predicted: 1
Original: 36 Predicted: 36
Original: 1 Predicted: 1
Original: 18 Predicted: 18
Original: 10 Predicted: 10
Original: 25 Predicted: 25
Original: 4 Predicted: 4
Original: 9 Predicted: 9
Original: 29 Predicted: 29
Original: 32 Predicted: 32
Original: 32 Predicted: 32
Original: 34 Predicted: 34
Original: 22 Predicted: 22