German Traffic Sign Recognition Benchmark

Balahari Vignesh Balu
Sahil Saini
Under guidance by Prof. Dr. Simon Ziegler

Agenda

- Introduction
- Dataset
- Neural Network Architecture
- Hyperparameter Optimization and Model fitting
- Evaluation

Introduction

- Project: German Traffic Sign Recognition
- Dataset: Kaggle Dataset, 50,000 Images, 43 Classes
- Technique Used: Multi-class Convolution Neural Networks

Dataset

There were a total of ~52000 images. The datasets were created as following:

- Train data: 39209 images organized into 43 classes
- Test data: 12630 unorganized images
- Train + Validation Split Ratio: 0.8

```
Found 31368 images belonging to 43 classes.
Found 7841 images belonging to 43 classes.
Found 12630 images belonging to 1 classes.
```

Data augmentation

- Augmentation using ImageDataGenerator from Keras

- Normalize Image
- Shear
- Zoom

- Other Augmentations offered: Rotation, Width Shift, Height Shift

```
training_datagen = ImageDataGenerator(
    rescale = 1./255,
    validation_split= .2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=False,
    fill_mode='nearest'
)
```

Data augmentation

- Generating Augmented Images

- Reshapes image to fixed size
- Does what was earlier done on opencv
- Seed- Ensure same randomness

```
train_generator = training_datagen.flow_from_directory(
    TRAINING_DIR,
    target_size=(30,30),
    color_mode='rgb',
    class_mode='categorical',
    subset='training',
    #batch_size=32,
    #shuffle=TRUE,
    seed=24
)
```

CNN Architecture

2 Convolutions, 1 Maxpooling

2 Convolutions, 1 Maxpooling

2 Hidden Layers: 256 + 128

Output Layer: 43 neurons
Activation = Softmax

Neutral Network Architecture

```
model = tf.keras.models.Sequential([
    # The input shape is the desired size of the image 30x30 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(32, (5,5), activation='relu', input_shape=(30, 30, 3)),
    # The second convolution
    tf.keras.layers.Conv2D(32, (5,5), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Dropout(0.2),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    # The Fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Dropout(0.2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    #  neuron hidden layer
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(43, activation='softmax')
])
```

Optimization and Genralization

We tweaked a few parameters in order to optimize the model.

- Dropouts
- Hidden Layers
- Data augmentation
- Epochs and Batch sizes

With Dropouts Always Better!

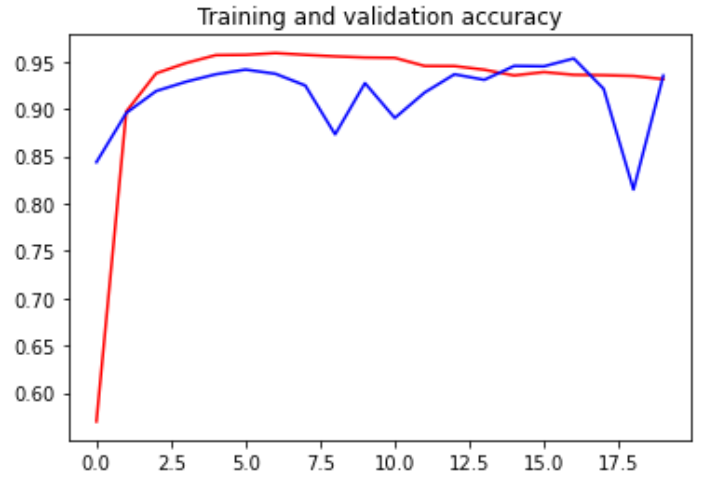
DATA AUGMENTATION	DROPOUTS	HIDDEN LAYER	TRAINING SET	VALIDATION SET	TEST SET
No	Yes	1	0.974751353263855	0.949496209621429	0.951148060174188
Yes	Yes	1	0.959831655025482	0.952174484729767	0.956215360253365
No	No	1	0.997067093849182	0.960336685180664	0.953602533650039
Yes	No	1	0.993560314178467	0.949496209621429	0.954631828978622

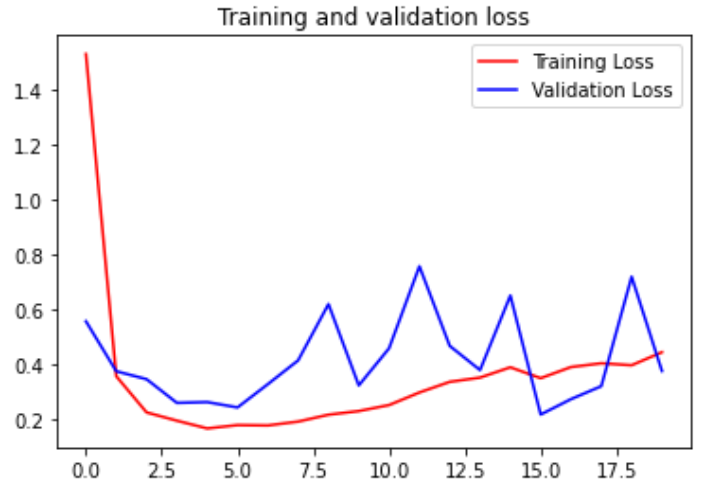
Evaluation- Optimized and Generalised

- Without dropouts, model performance on training and validation is better but performs bad on Test data- Over fitting and failure to generalize.
- With dropouts model performance on training and validation is always bad, but performs better on test data.- Generalizes better
- With Data augmentation always better.
- Network with 2 hidden layers is better.

Evaluation metrics and Parameters

- Training and Validation Loss
- Training and Validation Accuracy
- Confusion Matrix
- Accuracy score
- Mean F1 score





Accuracy Score

- Ratio of number of correct predictions to total number of predictions

```
from sklearn.metrics import accuracy_score
Test_Accuracy = accuracy_score(y_true.astype(int),
predicted_class_indices.astype(int))
```

F1 Score

- F1 score = 2* (1/(1/precision + 1/recall))

  precision = True positives / total positive predictions

  recall = True positives/ total actual Positives
- Mean F1 score for each class's F1 score was calculated

```
from sklearn.metrics import classification_report
print(classification_report(y_true.astype(int),
predicted_class_indices.astype(int), target_names=labels))
```

Is our model generalized enough?

- A more diverse dataset would lead to more genaralised classifier.
- Some real-world signs ended up being misclassified.
- Some non-German traffic signs were also classified better.
- More than one traffic sign, model classified only the bigger sign.

Thank You!