Traffic Sign Recognition Using Deep Learning

A Deep Learning Approach to Traffic Sign Identification

Done by:

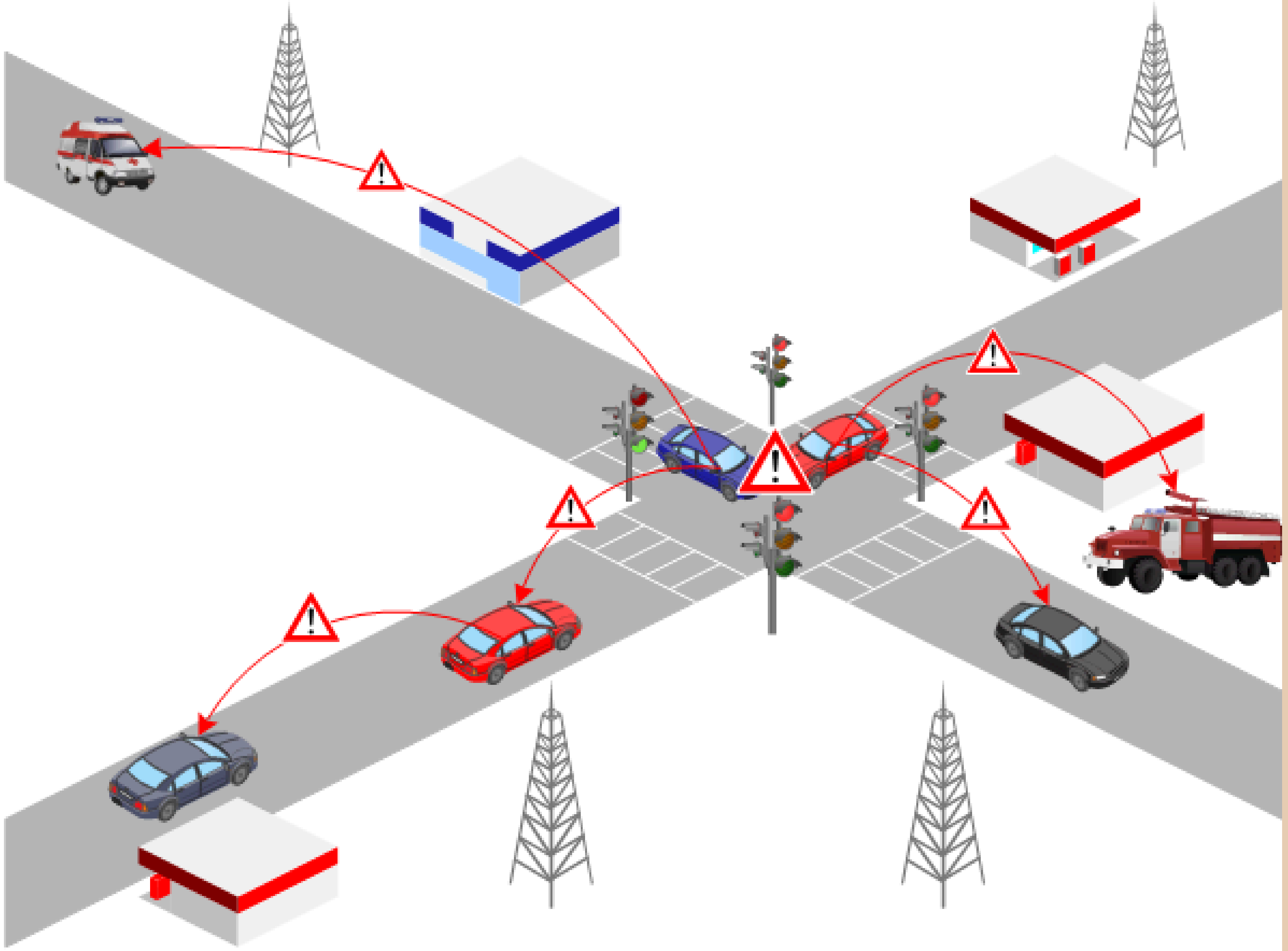
-Maram Alshehri, Shahad Faiz,   Fai Aladyani

Project Objective

- Implement a deep learning model (CNN) to recognize traffic signs.
- Recognizing traffic sign to solve traffic-related problems.

Why Traffic Sign Recognition?

- Improving road safety.
- Assisting autonomous driving systems.



Data Collection

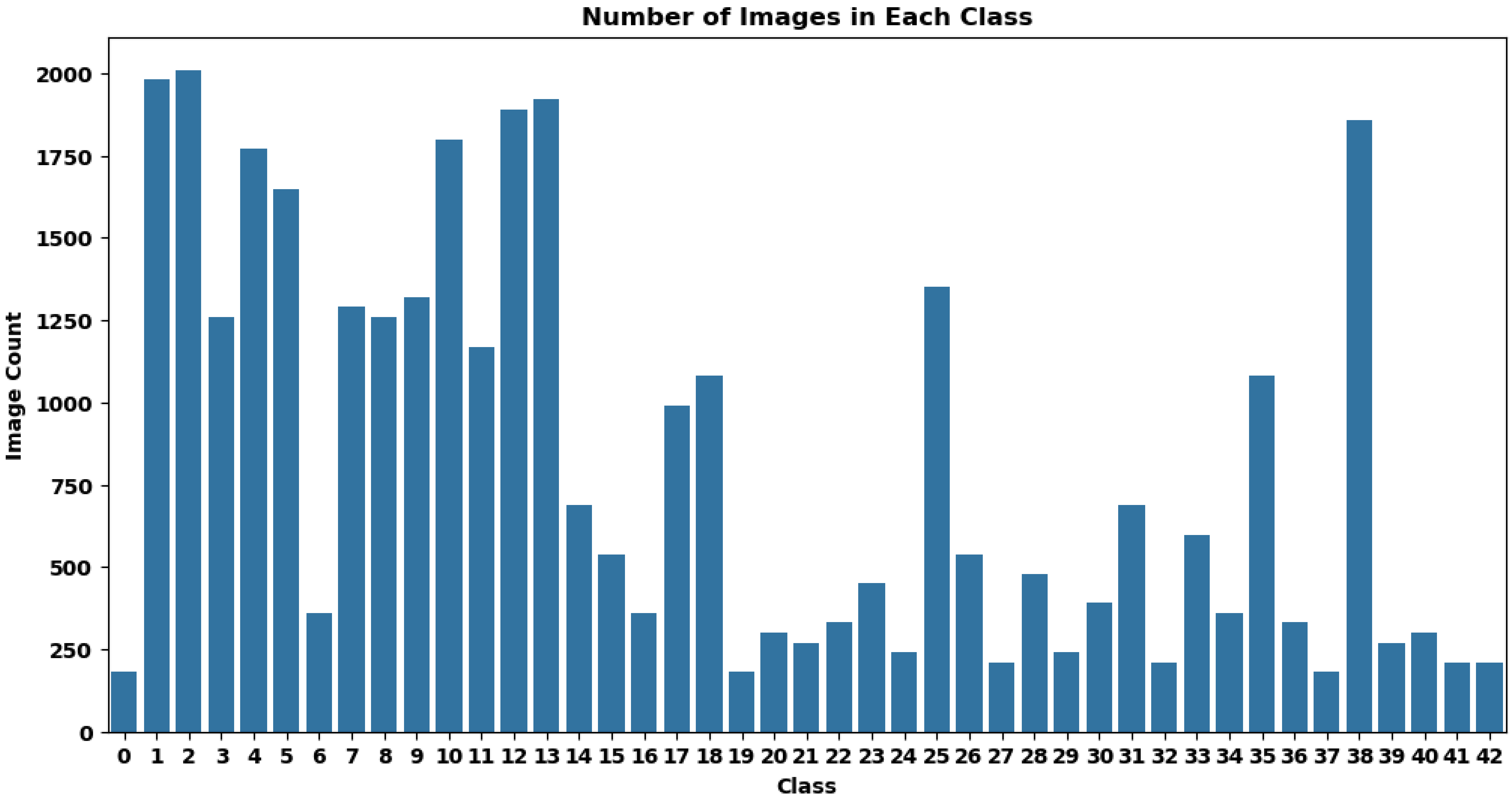
- Name of Dataset: Traffic sign
- Dataset source: Kaggle
- Programming language: Python
- Framework: TensorFlow and Keras.

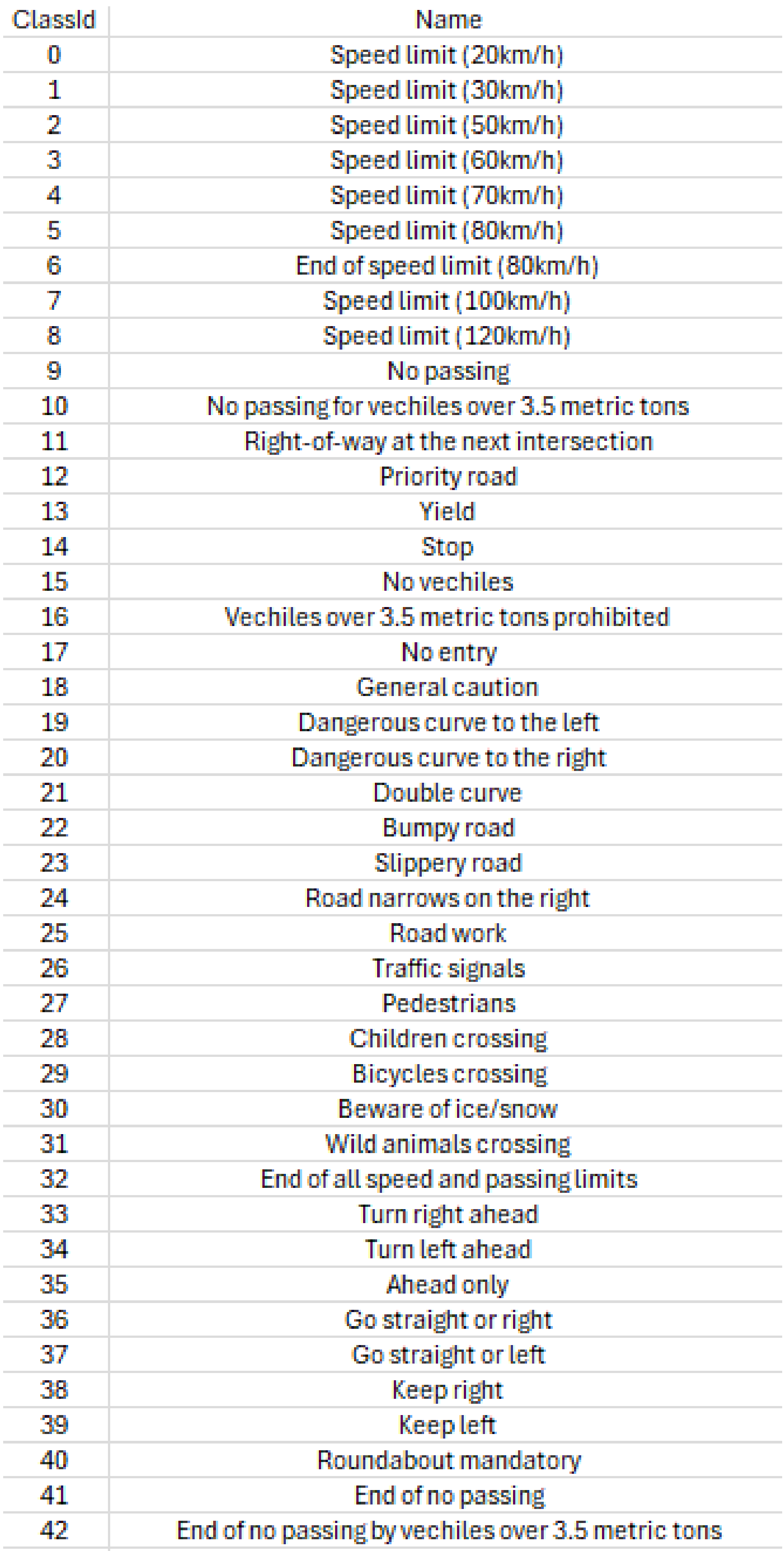
kaggle TensorFlow

python™ Keras

Sign Types

- Number of classes : 43
- Dataset size : 39,000 images

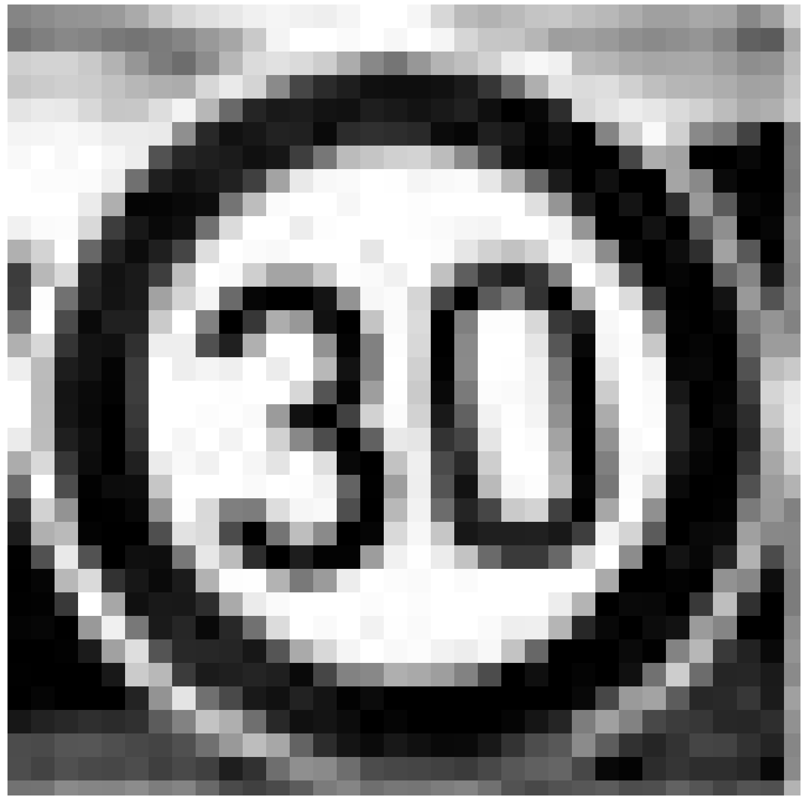




ClassId	Name
0	Speed limit (20km/h)
1	Speed limit (30km/h)
2	Speed limit (50km/h)
3	Speed limit (60km/h)
4	Speed limit (70km/h)
5	Speed limit (80km/h)
6	End of speed limit (80km/h)
7	Speed limit (100km/h)
8	Speed limit (120km/h)
9	No passing
10	No passing for vechiles over 3.5 metric tons
11	Right-of-way 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
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
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
40	Roundabout mandatory
41	End of no passing
42	End of no passing by vechiles over 3.5 metric tons

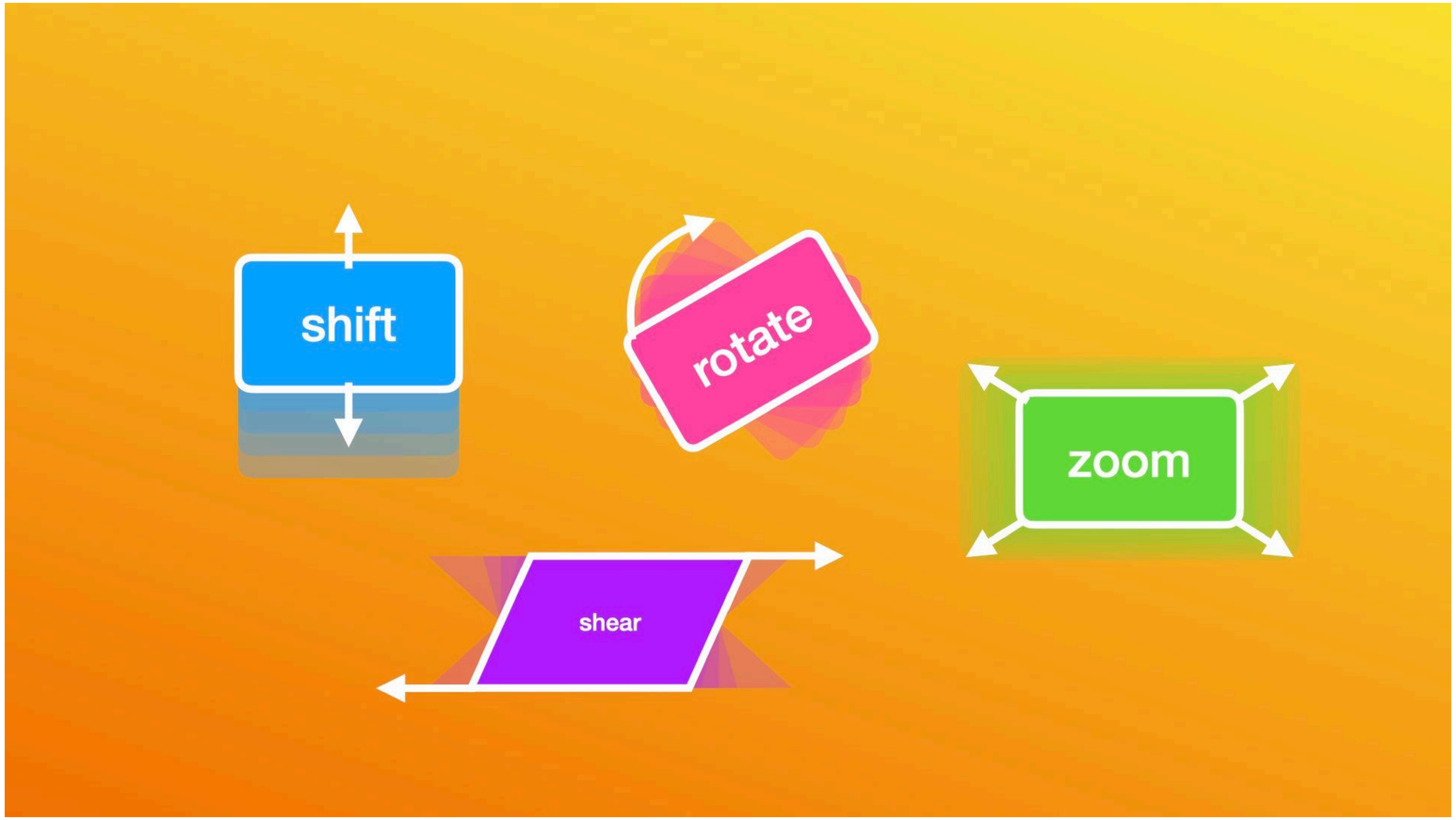
Preprocessing Steps

- Convert image to grayscale.
- Apply histogram equalization to enhance contrast.
- Normalize the pixel values by scaling between 0 and 1.
- Data splitting into training, validation and test sets, ratio 60/20/20 split.



Data Augnemntation

1. Width shift
2. Hight shift
3. Zoom
4. Shear
5. Rotation



Model Selection

Model Chosen:

Convolutional Neural Network (CNN)

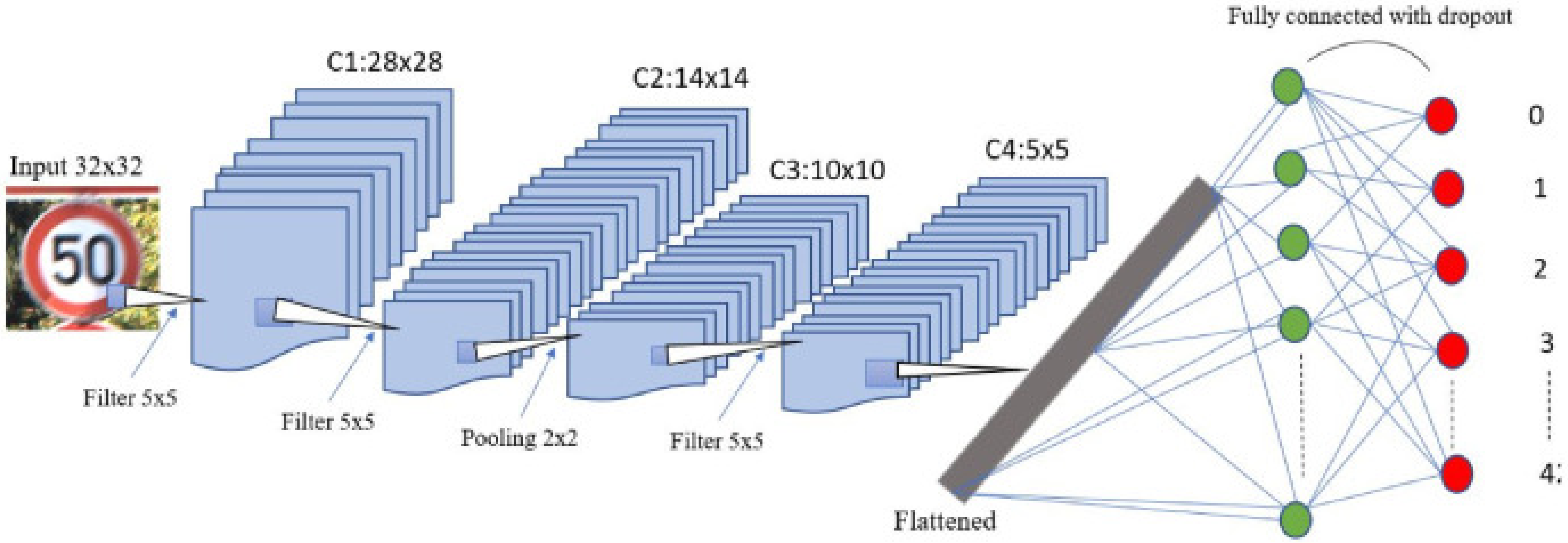
Why CNN?

Ideal for image data, automatically detects and extracts key features (edges, shapes).

Outperforms traditional models in image classification.

Architecture Overview:

Convolution layers extract features, pooling reduces size, and dense layers classify.



Model Implementation

Layers Explained:

- Conv2D:

Extracts features from the image.

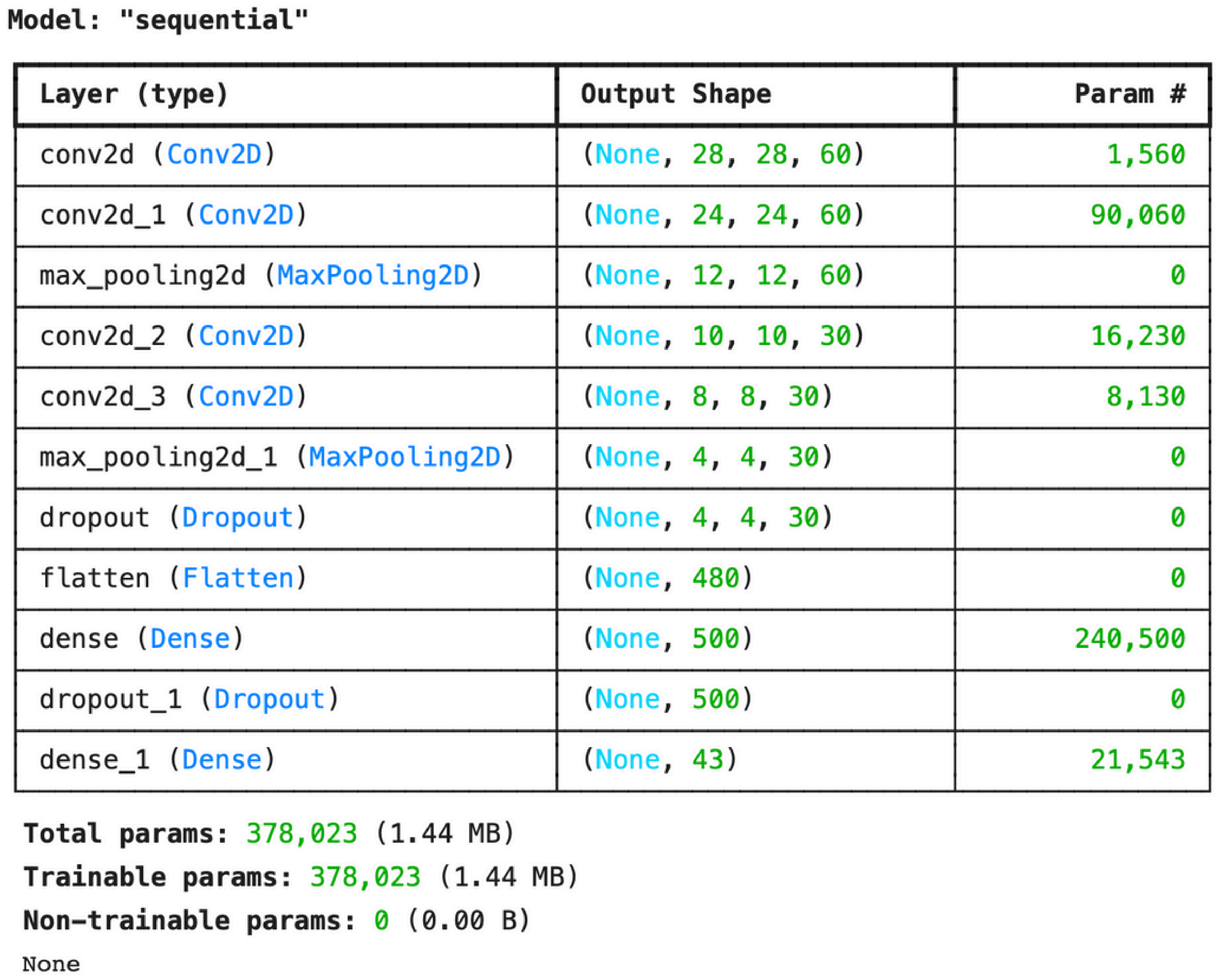
- MaxPooling:

Reduces the spatial size of the feature maps.

- Fully Connected (Dense):

Final classification based on extracted features.

- Loss Function: Categorical Crossentropy
- Optimizer: Adam
- Evaluation Metric: Accuracy



```
Model: "sequential"
┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d (Conv2D)                 │ (None, 28, 28, 60)     │         1,560 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 24, 24, 60)     │        90,060 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 12, 12, 60)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_2 (Conv2D)               │ (None, 10, 10, 30)     │        16,230 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_3 (Conv2D)               │ (None, 8, 8, 30)       │         8,130 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 4, 4, 30)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 4, 4, 30)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 480)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 500)            │       240,500 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)             │ (None, 500)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 43)             │        21,543 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 378,023 (1.44 MB)
 Trainable params: 378,023 (1.44 MB)
 Non-trainable params: 0 (0.00 B)
None
```

Training and Fine-Tuning

Epochs: 10

Batch Size: 32

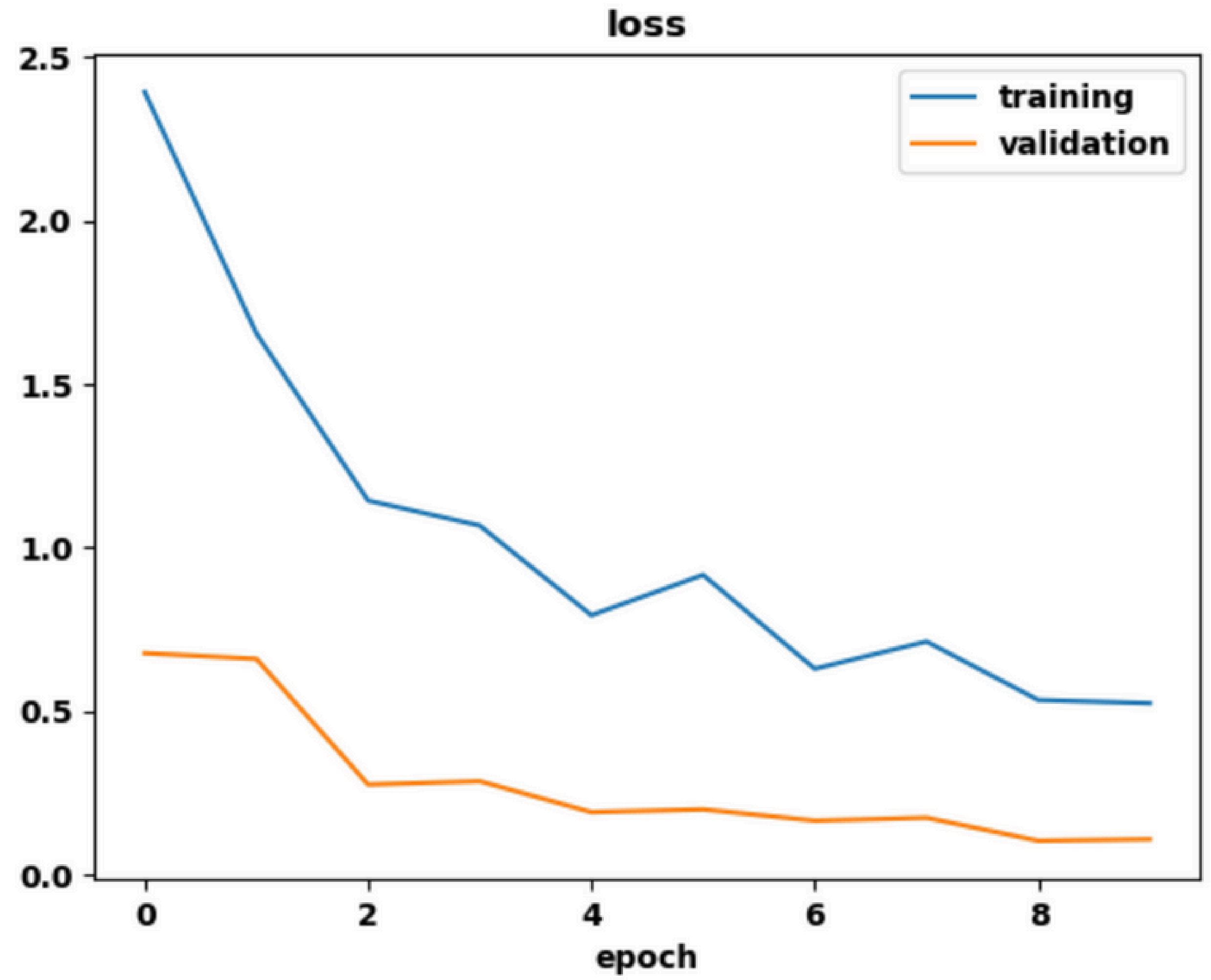
Optimizer: Adam (Learning rate: 0.001)

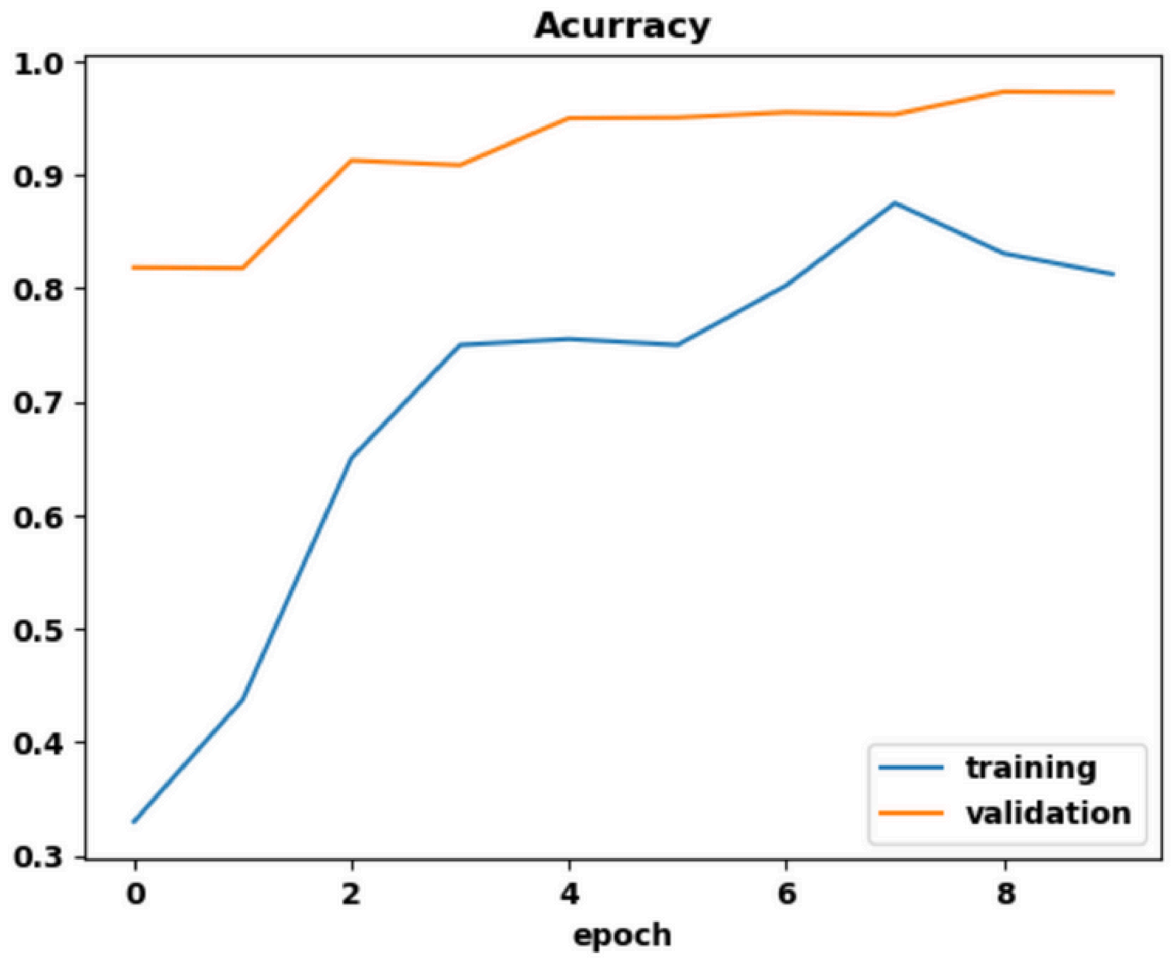
Loss Function: Categorical Crossentropy

Metrics: Accuracy

Hyperparameter Tuning:

Learning rate & dropout tuning for optimization.

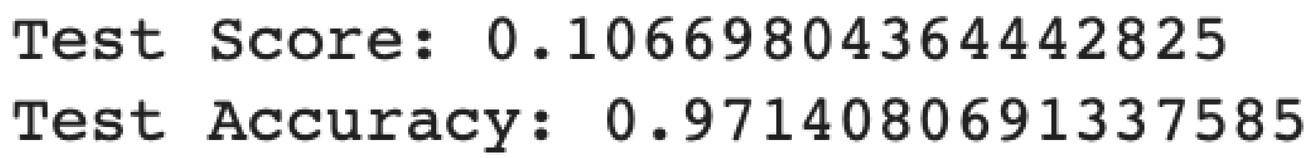




Model Evaluation

Accuracy: 97.14%

Test Score (Loss): 0.107



```
Test Score: 0.10669804364442825
Test Accuracy: 0.9714080691337585
```

Model Performance:

The model achieved high accuracy and low test loss, indicating effective traffic sign recognition on unseen data.

Conclusion and Future Work

Model performs well in identifying traffic signs.

Future improvements:

- Experiment with over-sampling and under-sampling techniques
- Add more signs
- Improve model accuracy.
- Real-time traffic sign detection

Thank you
for listening!