ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

AUTHOR: M.JOVANOVSKI

Enhancing Decision-Making through Data-Driven Insights

GitHub repository:
https://github.com/majajov/ML-Deep-Learning-for-Traffic-Sign-Recognition/tree/main

ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

ENHANCING DECISION-MAKING THROUGH DATA-DRIVEN INSIGHTS

➢ PROBLEM DESCRIPTION

- **Develop Traffic Sign Recognition system using 6 different models: AlexNet/DenseNet/ResNet50/ResNet101/VGGNet/CNN**
- Traffic Sign Classification model using various neural network architectures, including AlexNet, DenseNet, ResNet50, ResNet101, VGGNet, and CNN to explore each model's capabilities in traffic sign classification
- The objective was to accurately classify traffic signs based on their visual features.
- Trained and evaluated each model using a large dataset of traffic sign images. The training process involved optimizing the model parameters and adjusting hyperparameters to achieve the best performance..

ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

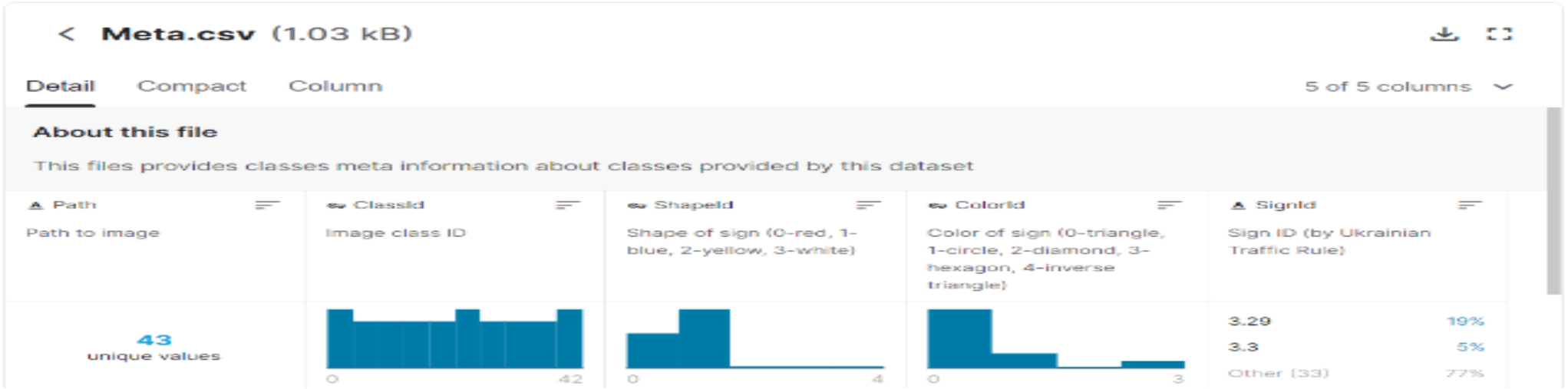
ENHANCING DECISION-MAKING THROUGH DATA-DRIVEN INSIGHTS

➢DATASET DESCRIPTION

❖Traffic_sign dataset is readily available for download from the free data depository: https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign/code (642MB)

❖ Dataset contains of 43 classes grouped into 43 subfolders (ranging from 0 to 42) and over 50,000 images split into 39,209 training images and 12,630 test images.

❖ Acknowledgement: INI Benchmark Website: https://benchmark.ini.rub.de/ (German Traffic Sign Benchmark)

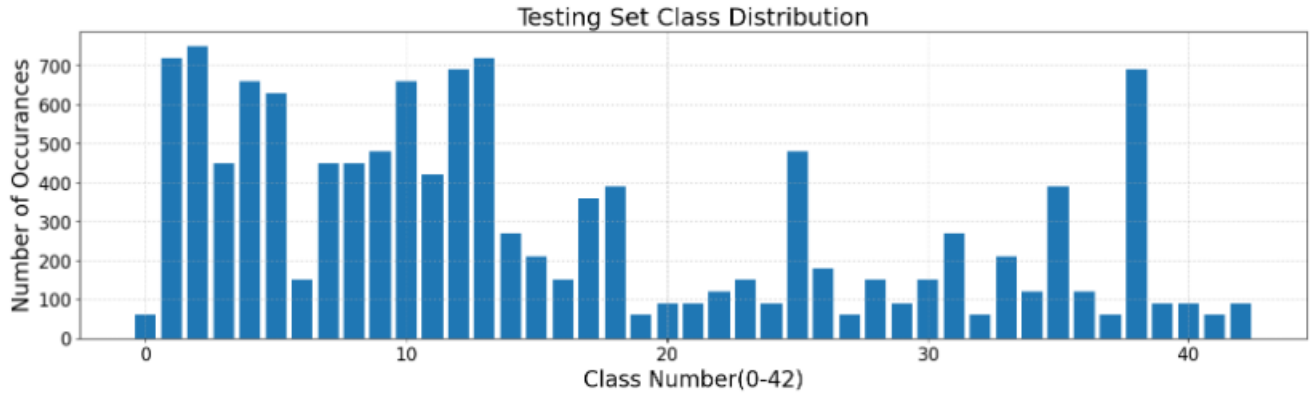


ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

ENHANCING DECISION-MAKING THROUGH DATA-DRIVEN INSIGHTS

➢ EXPLORATORY DATA ANALYSIS (EDA) – KEY INSIGNTHS

Histogram of class distributions across data set splits




```
Number of training examples = 34799
Number of validation examples = 4410
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43
```

ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

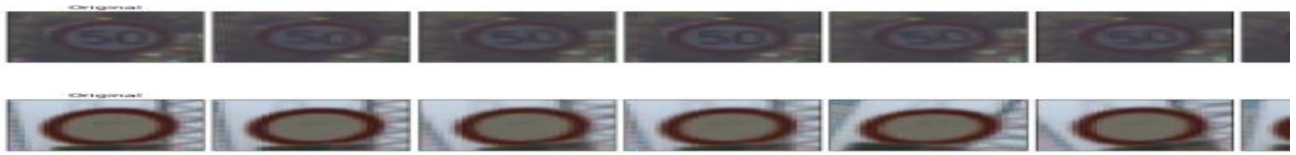
ENHANCING DECISION-MAKING THROUGH DATA-DRIVEN INSIGHTS

➢ DATA AUGMENTATION

❖Data augmentation → creates more data because histogram showed that distribution of classes is extremely unbalanced

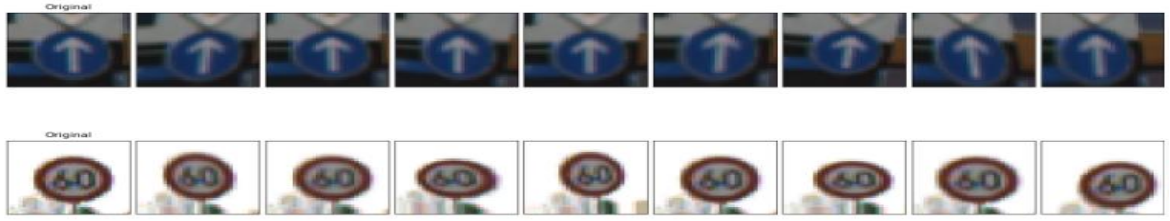
❖ Unbalanced datasets causes a heavy amount of bias in the training process.

➢Data Augmentation by Rotation



➢Data Augmentation by Translation

➢Data Augmentation by Holography Projection



ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

ENHANCING DECISION-MAKING THROUGH DATA-DRIVEN INSIGHTS

➢Deep Classification Architectures

- **Purpose**: Image classification using deep learning techniques.
- **Model**: AlexNet, a popular convolutional neural network architecture.
- **Implementation**: Utilizes Keras library for building and training the model.
- **Components**:
  1. **Convolutional Layers**: Extract image features, include batch normalization.
  2. **Activation Functions**: Enhance non-linearity.
  3. **Max Pooling**: Reduce feature map dimensions.
  4. **Flatten Layer**: Convert to 1D.
  5. **Fully Connected Layers**: High-level feature extraction.
  6. **Dropout**: Prevent overfitting.
  7. **Softmax Layer**: Multiclass classification.

Training:

- Input images categorized into 43 classes.
- Optimizer (e.g., Adam) and loss function (e.g., cross-entropy).
- Validation data for model performance.

Evaluation:

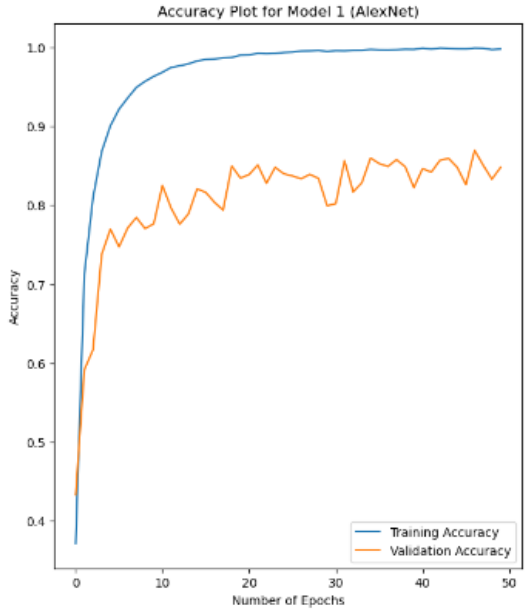
- Accuracy measures correctness.
- Testing dataset for generalization.

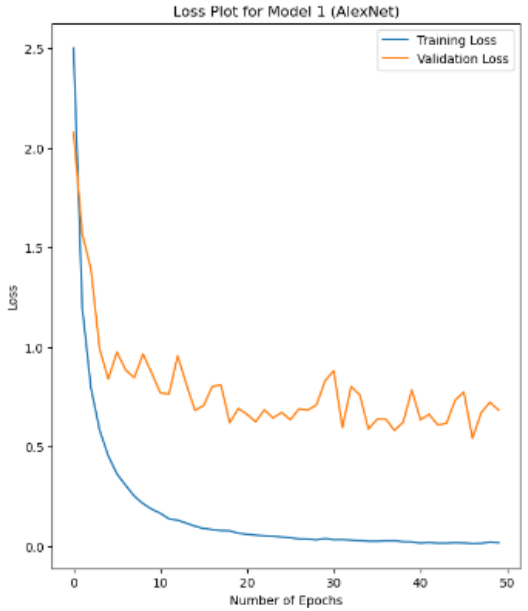
ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

ENHANCING DECISION-MAKING THROUGH DATA-DRIVEN INSIGHTS

➢ Model 1 - AlexNet

- The code implements AlexNet, a powerful CNN architecture for image classification using Keras. It employs convolutional layers, batch normalization, and max pooling for feature extraction. Fully connected layers with dropout ensure robustness, and the model summary provides an in-depth overview of its structure and parameters.



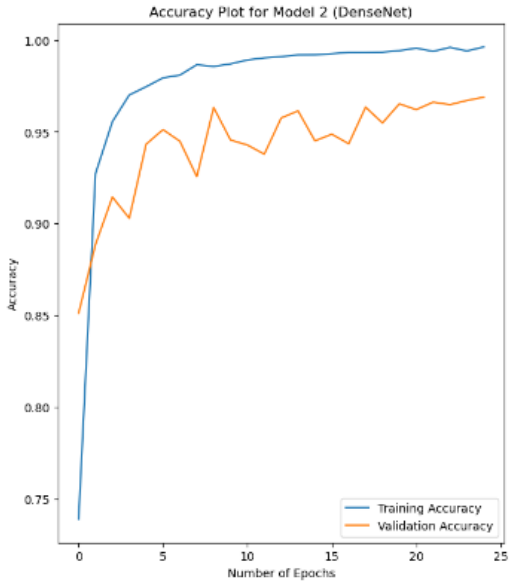


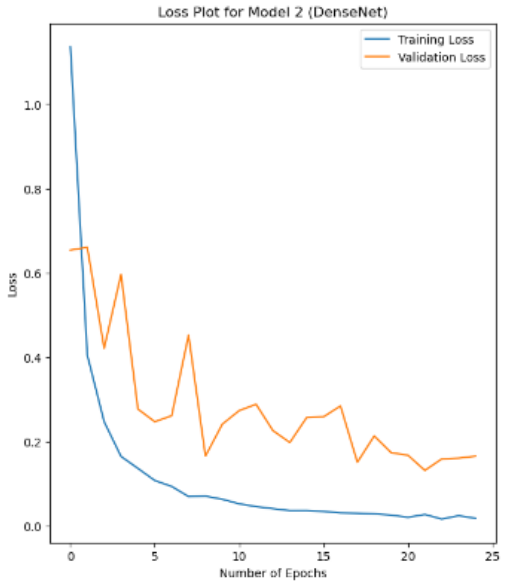
Model 1 Evaluation & Report

```
]: New_model1.evaluate(X_test_gray_norm, y_test)

   395/395 [==============================] - 2s 5ms/step - loss: 0.6142 - accuracy: 0.8655

]: [0.6142269968986511, 0.8654789924621582]
```

➢ Model 2 - DenseNet

- The enhanced DenseNET integrates hyperparameter tweaks, early stopping, and class weighting. Offers efficiency, better results, and addresses class skew.





Model 2 Evaluation & Report

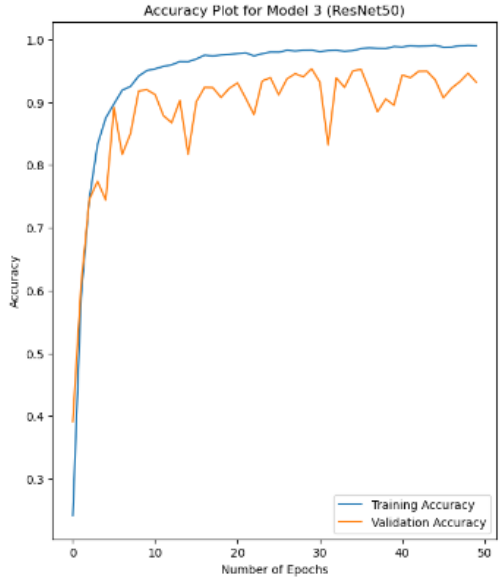
```
[38]: Best_DenseNet.evaluate(X_test_gray_norm, y_test)

   395/395 [==============================] - 8s 16ms/step - loss: 0.3071 - accuracy: 0.9430

[38]: [0.3071339428424835, 0.9429928660392761]
```

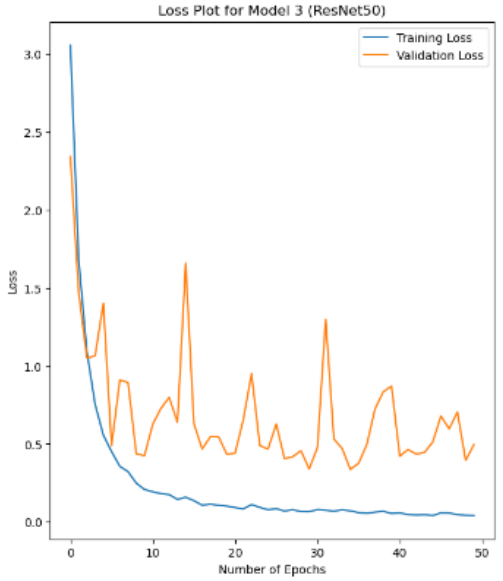
ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

ENHANCING DECISION-MAKING THROUGH DATA-DRIVEN INSIGHTS

➢ Model 3 –ResNet50

- This code uses ResNet50 as a base model, fine-tuning it with added layers for multi-class classification. Batch normalization, dropout, and softmax activation enhance performance while the summary offers a concise model overview.





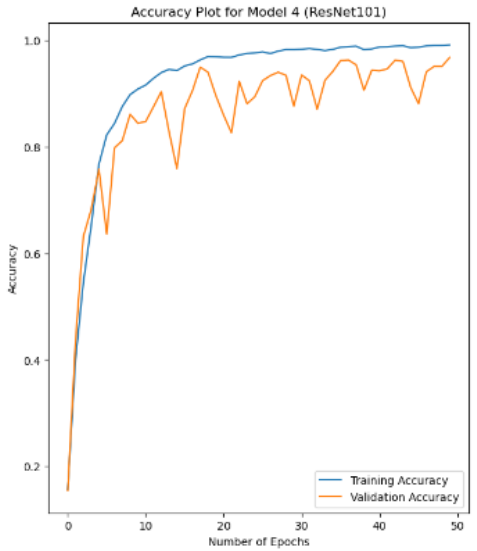
Model 3 Evaluation & Report

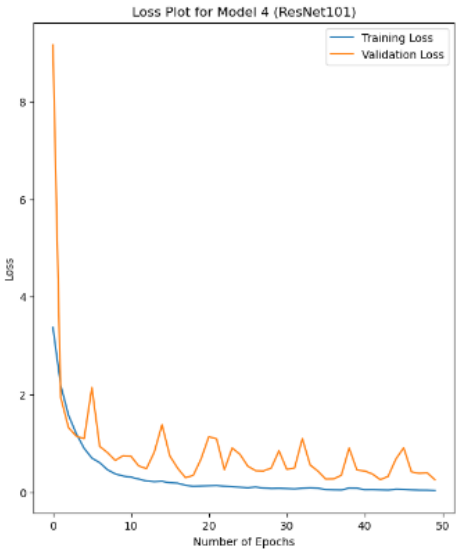
44]:
```
# Load the best performing ResNet model
best_ResNet50 = load_model('best_res50.h5')
```

45]:
```
# Evaluate the model on the test data
evaluation_results = best_ResNet50.evaluate(X_test, y_test)
```

```
395/395 [==============================] - 6s 12ms/step - loss: 0.5225 - accuracy: 0.9317
```

➢ Model 4 – ResNet101





Model 4 Evaluation & Report

[51]:
```
from keras.models import load_model

# Load the best performing ResNet model
best_ResNet101 = load_model('best_res101.h5')
```

[52]:
```
# Evaluate the model on the test data
evaluation_results = best_ResNet101.evaluate(X_test, y_test)
```

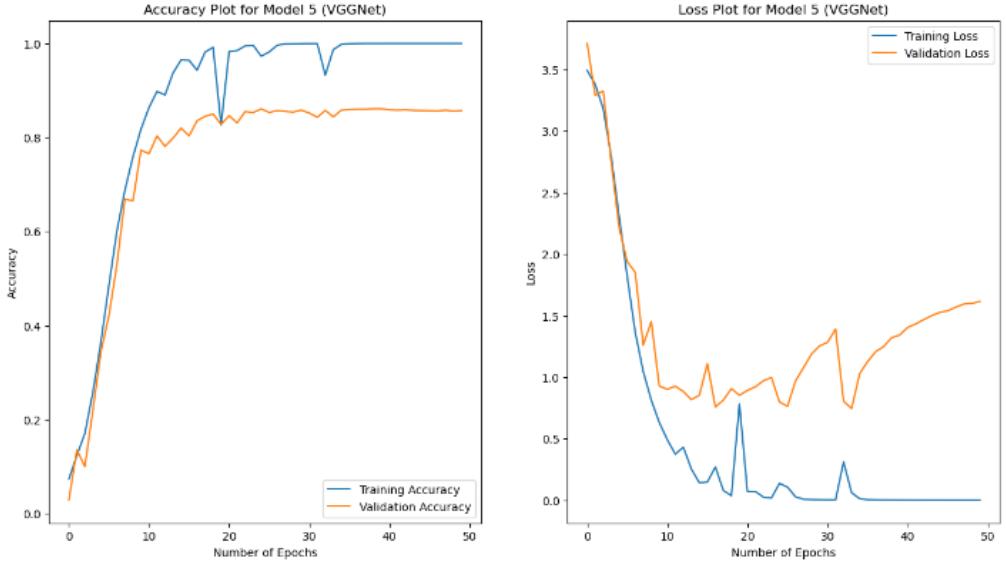
```
395/395 [==============================] - 10s 20ms/step - loss: 0.3429 - accuracy: 0.9556
```

ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

ENHANCING DECISION-MAKING THROUGH DATA-DRIVEN INSIGHTS

➢ Model 5 –VGGNet

- This code creates a CNN model based on VGG architecture, featuring convolutional and max pooling layers with ReLU activation. The summary offers insights into layer setups and parameter quantities, optimizing the architecture for multi-class classification using softmax activation.



Model 5 Evaluation & Report

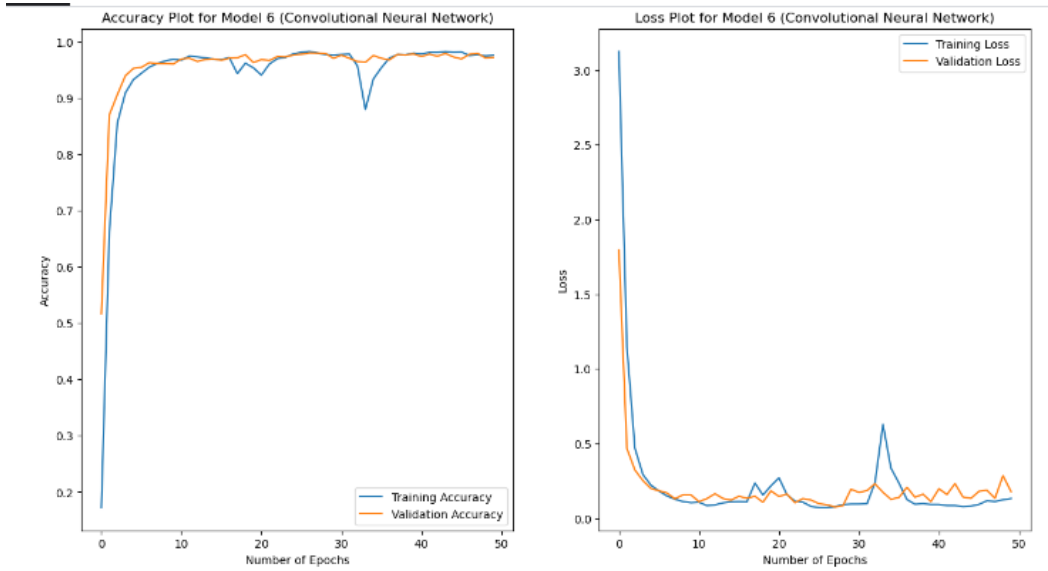
```
[58]: # Load the best performing ResNet model
best_VGGNet = load_model('best_vggnet.h5')
```

```
[59]: # Evaluate the model on the test data
evaluation_results = best_VGGNet.evaluate(X_test, y_test)
```

```
395/395 [==============================] - 3s 7ms/step - loss: 0.8629 - accuracy: 0.8384
```

➢ Model 6 – CNN

- This CNN image classifier employs convolutional layers, max pooling, and dropout for regularization. It processes 32x32 grayscale images, outputs into 43 classes, and is summarized by its structure and parameters



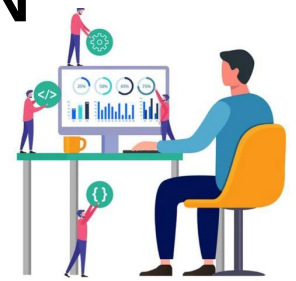
Model 6 Evaluation & Report

```
5]: from keras.models import Sequential, load_model

# Load the best trained model from the saved file
best_CNN_model = load_model('best_CNN_model.h5')

# Evaluate the model performance on the test data
evaluation_results = best_CNN_model.evaluate(X_test_gray_norm, y_test)
```

```
395/395 [==============================] - 2s 3ms/step - loss: 0.8738 - accuracy: 0.9394
```

ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

ENHANCING DECISION-MAKING THROUGH DATA-DRIVEN INSIGHTS



➢ DISCUSSION AND CONCLUSION

- Based on above results, the CNN Model achieved the highest accuracy of 97.31%, making it the most effective architecture for traffic sign classification in this study. The CNN Model likely benefited from its ability to learn complex spatial patterns and hierarchical representations through convolutional and pooling layers.

- The DenseNet architecture also performed well with an accuracy of 95.12%. DenseNet's unique connectivity pattern, where each layer receives direct inputs from all preceding layers, might have contributed to its strong performance by promoting feature reuse and alleviating the vanishing gradient problem.

- The ResNet50 and ResNet101 architectures achieved similar accuracies of 94.83% and 94.95% respectively. ResNet models employ residual connections to enable training of very deep networks, which likely helped in capturing intricate details of traffic signs and improving overall accuracy.

- On the other hand, VGGNet achieved an accuracy of 85.03%, which is relatively lower compared to other models. VGGNet's uniform architecture with smaller filter sizes might have limited its ability to capture fine-grained details, resulting in reduced accuracy.

Prediction on Test Data with Convolutional Neural Network model

ML DEEP LEARNING PROJECT: TRAFFIC SIGN RECOGNITION

ENHANCING DECISION-MAKING THROUGH DATA-DRIVEN INSIGHTS

➢ DISCUSSION AND CONCLUSION (CONT)

- **Potential Improvements:**
  - More Data: Expanding the dataset with more diverse examples could further improve the models. Collectin data from different sources or time periods might help capture a wider range of sign attributes.
  - External Validation: Validating the models on completely new and independent datasets (not used during training and testing) can provide further evidence of their effectiveness and real-world applicability
  - Keeping up with best practices and staying informed about the latest advancements in machine learning is crucial to continually improve and optimize ML effectiveness
  - In conclusion, the CNN Model demonstrated the highest accuracy among the evaluated architectures, followed closely by DenseNet, ResNet50, and ResNet101. These models showcase the effectiveness of convolutional neural networks in traffic sign classification tasks, while VGGNet achieved a comparatively lower accuracy. It is important to note that the choice of the best model architecture depends on the specific dataset and task at hand, and further experimentation and tuning may be necessary for optimal results.

GitHub repository:
https://github.com/majajov/ML-Deep-Learning-for-Traffic-Sign-Recognition/tree/main