Image Classification using Deep Neural Networks
A2-Group18

Introduction:

The field of machine learning plays an increasingly pivotal role in addressing complex real-world challenges, with one such challenge being the classification of images of European road traffic signs. Accurate identification of these signs is crucial for ensuring road safety and facilitating efficient traffic management systems. In this report, we delve into the task of classifying images of European road traffic signs using machine learning techniques.

Data Loading :

- The data loading process gathers grayscale traffic sign images from both original and collected datasets. It organizes these images by shape and sign labels, ensuring proper grouping for further analysis and preprocessing.
- We obtained data from the German Traffic Sign Recognition Benchmark (GTSRB), which we then converted to grayscale, resized to dimensions of 28x28 pixels, and converted to the PNG format.

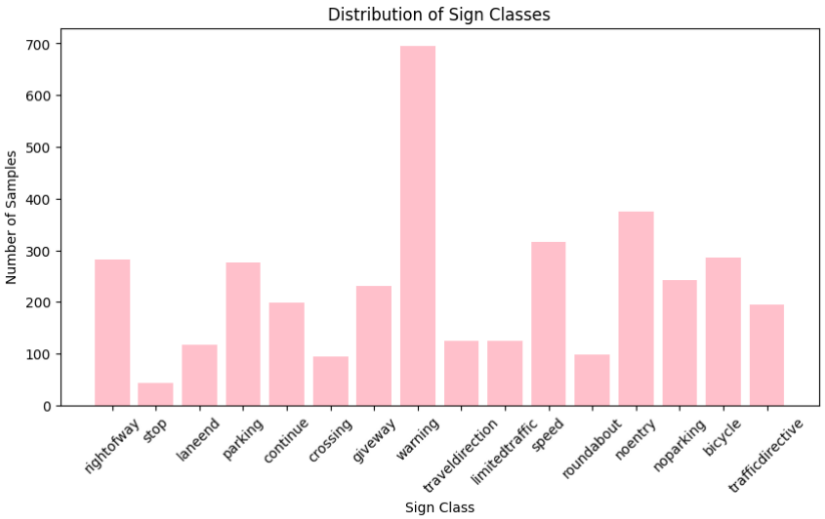
	Original	**Collected**
No. of Images	3699	1747

Exploratory Data Analysis:

- Checking Duplicates
- Visualizing Sample Images
- Analyzing the Distribution of Shape and Sign Classes
- Analyzing Pixel Value Statistics for given Dataset¶
- Plotting Pixel Intensity Histograms to Analyze Image Brightness and Color Distribution
- Edge Detection Analysis for Shape and Sign Images

Analyzing the Distribution of Shape and Sign Classes:

The analysis reveals significant class imbalances in both shape and sign classes within the dataset. Notably, the "round" shape class exhibits a disproportionately large number of samples compared to other shape classes, while the "hex" shape class has a considerably smaller number of samples. Similarly, among sign classes, the "warning" sign class shows a substantially higher number of samples, whereas the "stop" sign class has notably fewer samples. Class imbalances can lead to biased model predictions, where the model tends to favor classes with more samples.

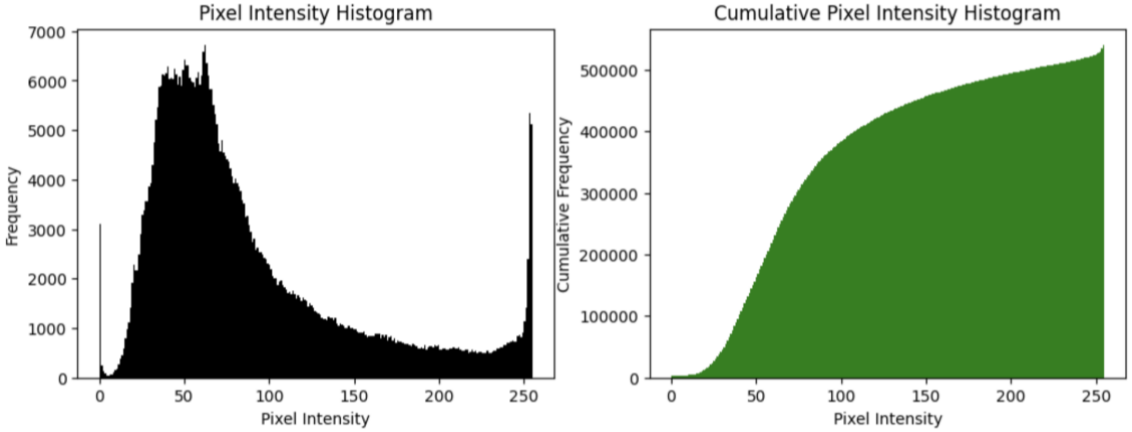


Pixel Value Statistics for given Dataset:

The mean pixel value indicates that, on average, the images in the shape data have a moderate intensity. The median pixel value being lower than the mean suggests that the distribution of pixel intensities may be skewed towards lower values or that there may be some dark areas in the images. The relatively high variance of pixel values indicates that there is a wide range of pixel intensities within the images, suggesting variability in brightness levels or texture across the dataset.

```
Statistics for given Dataset:
Mean Pixel Value: 98.67499184889228
Median Pixel Value: 77.0
Variance of Pixel Values: 4312.757697039149
```

Plotting Pixel Intensity Histograms to Analyze Image Brightness and Color Distribution: The histograms made for our data often show peaks towards the left side, indicating that most pixel values are lower, which corresponds to darker regions in the images. This observation can be crucial for understanding the overall brightness and color distribution within the data sets.



Data Preprocessing: Data preprocessing is essential in machine learning to enhance data quality and model performance. It involves transforming raw data into a format that is suitable for analysis.

- **Data Scaling:** We rescaled the pixel values to a range between 0 and 1 to normalize the data using the rescale parameter in the ImageDataGenerator, which facilitates effective training of machine learning models by simplifying the data representation and aiding in model convergence. The need for scaling was indicated by the wide range of pixel value statistics in our dataset, such as a mean of 98.67, median of 77.0, and variance of 4312.76.

- **Data Formatting:** We specified the data format as 'channels_last', indicating that the color channels are represented as the last dimension of the input data. This format ensures compatibility with our deep learning models, simplifying the integration of image data into the training pipeline.

- **Data Splitting for Model Training and Evaluation:** Train (70%), Validation (15%), Test (15%)

```
Training, Validation, and Test Sets for Shapes:
```

Set	Train Images(70%)	Validation Images(15%)	Test Images(15%)	Collected Test Images
diamond	196	43	43	63
hex	29	7	7	36
square	480	104	104	157
triangle	648	139	139	574
round	1227	263	264	917

- **Converting Labels into numerical labels using One-Hot Encoding:** The shape labels are converted into numerical labels and then subjected to one-hot encoding for both the shape and sign datasets. This process facilitates categorical representation for model training and evaluation.

```
Shape Label: diamond
One-hot Encoded Label: [1. 0. 0. 0. 0.]
```

Evaluation Framework:

Evaluation metrics are used to assess the performance of machine learning models. They provide quantitative measures to evaluate how well a model's predictions match the actual outcomes. Below are some which we used:

- **Accuracy**: The ratio of correctly predicted instances to the total instances.
- **Precision**: The ratio of correctly predicted positive observations to the total predicted positive observations.
- **Recall**: The ratio of correctly predicted positive observations to the all observations in the actual class.
- **F1 Score**: The weighted average of Precision and Recall.
- **Cohen's Kappa**: A statistic that measures inter-rater agreement for categorical items. Cohen's Kappa is used to assess the agreement between the predicted labels and the true labels, taking into account the possibility of agreement occurring by chance. It provides a more robust evaluation of model performance, especially in scenarios where classes are imbalanced or when the accuracy metric alone may not be sufficient.

Validation Set Up:

- Validation was conducted using a rigorous and standardized approach to ensure the reliability and accuracy of the model's performance assessment.
- The dataset was divided into training and testing sets to facilitate model training and evaluation.
- The training set was used to train the model and validation set was used for validating results, while the testing set was kept separate to evaluate the model's performance on unseen data.

Model Evaluation and Comparison Table:

	Model Description	**Details**	**Accurcy**	**Precision**	**Recall**	**F1 Score**	**Cohen's Kappa**	
1	Base line model: Multilayer Perceptron	- Scaling applied - Flattening of images - activation: Sigmoid - optimizer: SGD - learning rate: 0.01	0.74	0.72	0.74	0.67	0.56	Shape
			0.18	0.03	0.18	0.05	0	sign
2	Basic CNN	- 3 convolutional layer (3x3) - 3 Pooling Layer (2,2) - Flatten Layer - Dense Layer - activation: Sigmoid - optimizer: SGD	0.47	0.22	0.47	0.30	0	shape
			0.18	0.03	0.18	0.05	0	sign
3	CNN -Batch Normalisation	- 3 convolutional layer (3x3) - batch normalization - 3 Pooling Layer (2,2) - Flatten Layer - Dense Layer - activation: Sigmoid - optimizer: SGD	0.99	0.99	0.99	0.99	0.98	shape
			0.97	0.98	0.97	0.97	0.97	sign
4	Optimised CNN	- 3 convolutional layer (3x3) - l2 regularization applied - batch normalization - padding: same - 3 Pooling Layer (2,2) - dropout applied: 0.6 - Flatten Layer - Dense Layer - activation: relu	0.87	0.87	0.87	0.87	0.81	shape
			0.80	0.83	0.80	0.79	0.78	sign

		- optimizer: adam - learning rate: 0.01 - bias: true						
5	CNN – Data Augmentation (2 augmented images generated for each image)	- 3 convolutional layer (3x3) - l2 regularization applied - batch normalization - padding: same - 3 Pooling Layer (2,2) - dropout applied: 0.6 - Flatten Layer - Dense Layer - activation: Sigmoid - optimizer: SGD - learning rate: 0.01 - bias: true	0.92	0.91	0.92	0.92	0.89	shape
			0.86	0.86	0.86	0.85	0.84	sign
	CNN – Edge Detection	- hyperparameter tuned parameter - activation: relu - layer_size:128 - No. of Layers: 3 - optimizer: adam (shape) - optimizer: adam (sign) - learning rate: 0.01 - bias: true - dropout applied: 0.6	0.95	0.94	0.95	0.94	0.94	shape
			0.67	0.64	0.67	0.62	0.83	sign
	Inspired ResNet Architecture	- convolutional block: 4 layers - identity block: 3 layers - model block - activation: relu - optimizer: adam - learning rate: 0.01	0.78	0.76	0.78	0.77	0.67	shape
			0.58	0.49	0.58	0.49	0.53	sign
	ResNet50	- Pretrained Model	0.47	0.22	0.47	0.30	0	shape
			0.18	0.03	0.18	0.05	0	sign

Baseline Model (Model 1): We are using MLP (Multilayer Perceptron) with 2 hidden layers as our baseline model, we opted this model for its simplicity, flexibility, efficiency, and reliance on historical success. It serves as a pragmatic starting point for shape and sign classification tasks, enabling systematic exploration of more sophisticated approaches while maintaining a solid foundation in traditional machine learning principles. It captures basic patterns in the data without incorporating any advanced techniques.

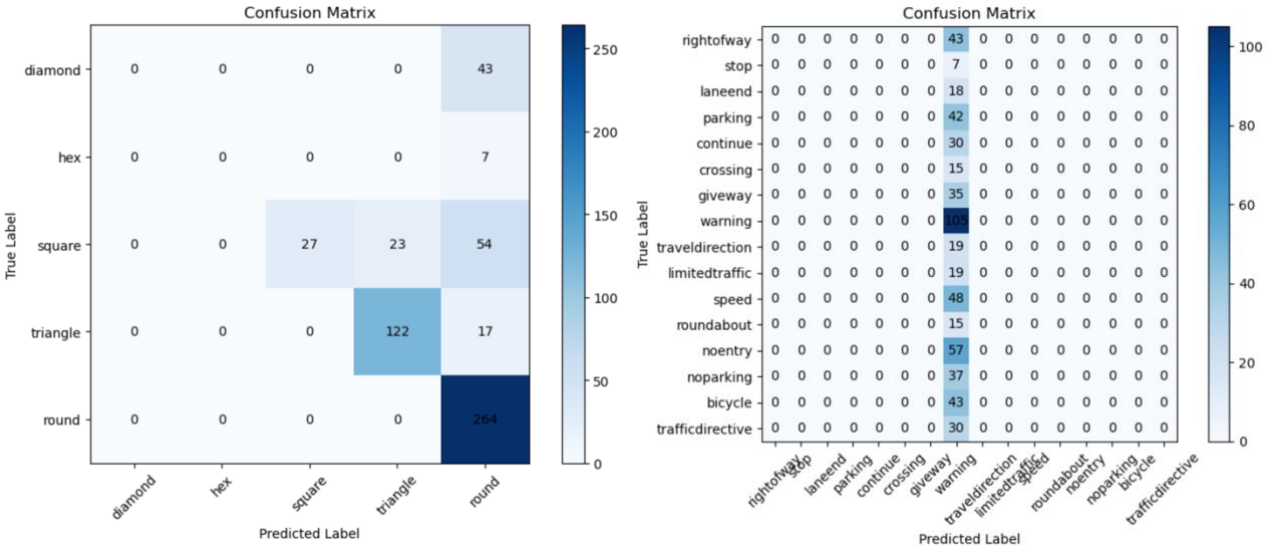
Flattened the images to convert them from 2D arrays to 1D arrays, making them suitable for input into a Multi-Layer Perceptron (MLP) model. This transformation allows each pixel in the image to be treated as a separate feature, enabling the MLP to learn patterns directly from the pixel values.

It defines a sequential model consisting of three dense layers. The architecture is as follows:

- **Input Layer**: input_dim (28x28), which corresponds to the number of features in the input data.
- **Hidden Layers**: Two hidden layers with 128 and 64 neurons respectively
- **Output Layer**: The output layer has num_classes neurons, representing the number of classes in the classification problem.

The performance metrics of the MLP model on the test data reveal a significant difference in accuracy between predicting shapes and signs. For the shape classification task, the model achieved an accuracy

of 69.48%, indicating a reasonably good performance. However, for the sign classification task, the accuracy dropped substantially to 18.65%, suggesting that the model struggles significantly with this



task. These results highlight the need for further investigation and potential model improvements for better handling of sign classification.

Improved Classification Model : Convolutional Neural Network

A Convolutional Neural Network (CNN) is a deep learning algorithm designed to analyze visual data. It employs a hierarchical architecture that learns to extract features from images through convolutional layers. CNNs are particularly effective for tasks like image classification, object detection, and image segmentation due to their ability to automatically learn spatial hierarchies of features from raw pixel values.

The CNN model is used for shape and sign traffic signal data due to its powerful feature extraction capabilities, robustness to variations in image data, ability to prevent overfitting, and flexibility in configuration. These attributes make it an ideal choice for achieving high accuracy in traffic signal classification tasks:

- **Spatial Hierarchies:** CNNs are excellent at capturing spatial hierarchies in images, which means they can detect edges, textures, shapes, and more complex patterns. This is essential for accurately identifying different traffic signs and shapes that have unique visual characteristics.
- **Automatic Learning:** Unlike traditional methods that require manual feature extraction, CNNs automatically learn the most relevant features directly from the pixel data.

Basic CNN Model (Model 2) :

It consists of multiple layers arranged sequentially:

- **Convolutional Layers:** Three convolutional layers were used with increasing filter sizes (32, 64, and 128) and kernel size (3x3). These layers learn spatial hierarchies of features in the input images.
- **Pooling Layers:** After each convolutional layer, a max-pooling layer with a pool size of (2, 2) is added. This layer reduces the spatial dimensions of the feature maps, aiding in translation invariance and reducing computational complexity.
- **Flatten Layer:** This layer flattens the output from the previous layers into a one-dimensional vector, preparing it for input to the fully connected layers.
- **Fully Connected Layers:** Two dense (fully connected) layers follow the flattened output. The first dense layer consists of 128 neurons and uses the specified activation function. The final layer, with the number of neurons equal to the number of classes, employs the softmax activation function to output class probabilities.

The model achieved an accuracy of 47.3% on the validation data and 47.4% on the test data, indicating moderate performance. However, for sign classification, the model's accuracy was only 18.65% on both validation and test data, highlighting a significant challenge in effectively classifying signs.

Adding Batch Normalization to Improve Model Performance (Model 3):

To enhance the performance and stability of our CNN model, we have integrated batch normalization into the network architecture. Batch normalization normalizes the inputs to each layer, reducing internal covariate shift, improving gradient flow, and allowing for higher learning rates. This adjustment has led to significant improvements in training efficiency and model accuracy.

Performance metrics show remarkable improvements:

- For shape classification, the model achieved 99.1% accuracy on validation data and 98.92% on test data.
- For sign classification, the model achieved 99.47% accuracy on validation data and 98.93% on test data.

These metrics demonstrate that batch normalization not only improves the model's training stability but also enhances its generalization capabilities, resulting in high performance on both validation and test datasets.

- **Stabilized Learning Process:** Batch normalization reduces internal covariate shift, ensuring that the distribution of inputs to each layer remains stable throughout training. This stabilization results in a more consistent and reliable learning process, enabling the model to converge more quickly and effectively.
- **Improved Gradient Flow:** By normalizing the inputs to each layer, batch normalization helps maintain the scale of gradients within a manageable range. This prevents issues like exploding or vanishing gradients, which are common in deep networks, thereby enhancing the gradient flow and facilitating more efficient weight updates.
- **Higher Learning Rates:** The normalization of inputs allows the use of higher learning rates without risking instability in training. This accelerates the training process, enabling faster convergence to optimal performance.
- **Regularization Effect:** Batch normalization introduces a slight regularization effect by adding noise to the learning process through batch statistics. This helps prevent overfitting, as the model becomes less sensitive to minor fluctuations in the training data.

CNN Model Optimisation (Model 4):

Further Enhancements with Regularization, Dropout, and Padding:

To further improve the model's robustness and generalization capabilities, we can incorporate additional techniques such as L2 regularization, dropout, and padding. Although these methods initially result in a slight decrease in accuracy, they offer several long-term benefits:

1. L2 Regularization: Applying L2 regularization penalizes large weights, encouraging the model to maintain simpler, more generalizable patterns. This helps in preventing overfitting, where the model performs well on training data but poorly on unseen data.
2. Dropout: Introducing dropout randomly deactivates a fraction of neurons during each training iteration. This prevents the model from becoming overly reliant on specific neurons, thereby improving its ability to generalize, and reducing the risk of overfitting.
3. Padding: Using padding ensures that the spatial dimensions of the input are preserved after convolution operations, allowing the model to learn features more effectively, especially near the edges of the input images.

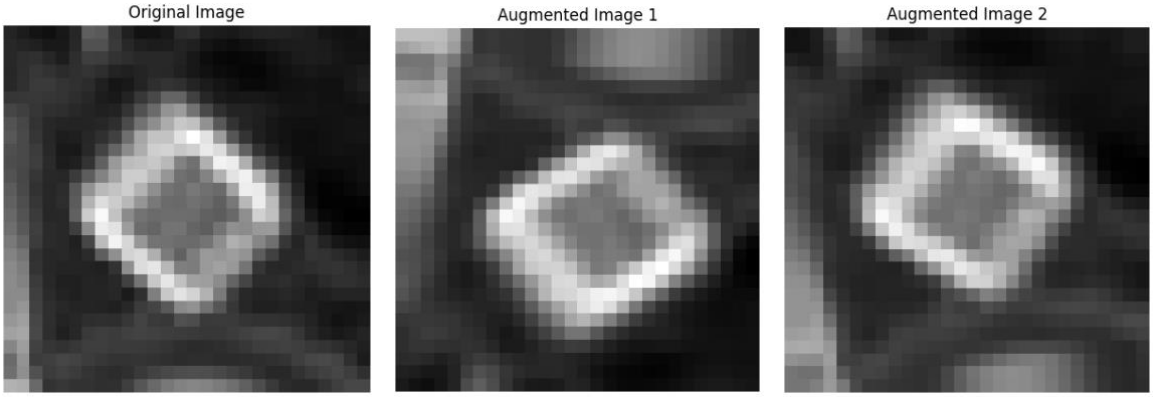
Performance metrics after incorporating these techniques:
- Shape classification: 75.9% accuracy on validation data and 78.1% on test data.
- Sign classification: 78.86% accuracy on validation data and 79.22% on test data.

Compared to the results after batch normalization alone (99.1% and 98.92% for shape, 99.47% and 98.93% for sign), these techniques resulted in lower accuracy. However, they are essential for building a more robust and generalizable model. The integration of L2 regularization, dropout, and padding helps in mitigating overfitting, ensuring the model performs well on unseen data.

CNN with Augmented Data (Model 5): This process helps to artificially increase the size and diversity of the training dataset, improving the model's ability to generalize.

- ImageDataGenerator is used with various augmentation parameters, such as rotation, width shift, height shift, horizontal and vertical flips, zoom, and rescaling. These transformations enhance the diversity of the training data by creating variations of the original images.
- The augmentation process is applied separately to the shape and sign training datasets. For each class label, new augmented images are created and combined with the original images to form a larger and more diverse training set.
- The total number of augmented images generated for both shape and sign data is calculated, indicating the effectiveness of the augmentation process in increasing the dataset size.
- Applied resizing to make images of 28x28

Data augmentation is crucial in machine learning to improve model performance by providing a more extensive and varied dataset. This process helps the model generalize better and reduces the risk of overfitting.



Combined Dataset for Shapes (Original and Augmented):

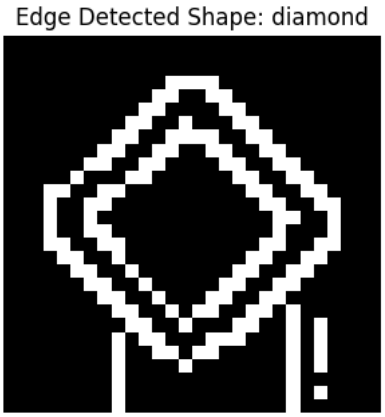
Shape Type	Original	Augmented	Combined
diamond	196	392	588
hex	29	58	87
square	480	960	1440
triangle	648	1296	1944
round	1227	2454	3681
Total	2580	5160	7740

After augmentation, the shape classification model achieved 94.6% accuracy on validation data and 93.54% on test data, while the sign classification model achieved 85.44% and 85.97% accuracy respectively. These results highlight the effectiveness of data augmentation in enhancing model performance for image classification tasks.

CNN with Edge Detection (Model 6): We applied edge detection preprocessing for CNN models, particularly beneficial for shape and sign recognition tasks. By applying the Canny edge detection algorithm to images, the resulting edge-detected images enhance features relevant to shape and sign identification. These preprocessed images are then utilized for training, validation, and testing CNN models.

- Edge detection enhances feature representation by highlighting edges, corners, and contours, aiding in shape and sign identification.

- It improves model robustness across diverse lighting conditions, backgrounds, and image distortions.
- Simplifying images to essential features through edge detection streamlines CNN learning, facilitating pattern recognition.
- Edge-detected images reduce computational complexity and memory usage during model training and inference.



Hyperparameter Tuning : We are optimizing a Convolutional Neural Network (CNN) model by performing grid search over various hyperparameters such as activation functions, optimizers, number of layers, and layer sizes. It leverages GridSearchCV to find the best combination of parameters for the given training data.
On applying **Grid Search Cross-Validation**, optimal hyperparameters for our model were:

activation	layer_size	num_layers	optimizer
relu	128	3	adam

EarlyStopping: Used early stopping in CNN model to prevent overfitting and optimize training efficiency. Early stopping allows the model to monitor its performance on a separate validation dataset during training. When the performance stops improving or begins to degrade, early stopping halts the training process. This prevents the model from continuing to learn from noise in the training data, thereby avoiding overfitting, where the model performs well on the training data but poorly on unseen data. By stopping the training process early, we can save computational resources and time while still achieving optimal model performance.
ReduceLROnPlateau: We used it in our model to automatically decrease the learning rate when the model stops improving, helping stabilize training, speed up convergence, prevent overshooting, and improve generalization.

Edge detection significantly enhanced feature representation, aiding in shape and sign identification while improving robustness across diverse conditions. Hyperparameter tuning via grid search optimized model architecture and parameters, while EarlyStopping prevented overfitting, optimizing training efficiency. ReduceLROnPlateau dynamically adjusted the learning rate, stabilizing training and improving generalization. Our model achieved impressive accuracy rates: 94% on collected shape data and 96% on test data, demonstrating robust performance. For sign recognition, accuracy rates stood at 80% and 84%, respectively. These results affirm the efficacy of our approach in leveraging advanced techniques for superior CNN performance in shape and sign recognition tasks.

Custom ResNet32-Inspired Neural Network for Image Classification (Model 7) :

This custom neural network is designed for image classification and draws inspiration from the ResNet50 architecture. ResNet (Residual Networks) is known for its ability to train very deep networks without the vanishing gradient problem, thanks to its residual blocks. Here, we have created a simpler version of ResNet50 that is computationally less demanding, making it suitable for smaller datasets and limited computational resources. The model is designed to handle grayscale images of size 28x28 and includes several key components and design principles from ResNet50, adapted to be computationally manageable.

Explanation of the Architecture

Identity Block: The identity block is a fundamental building block of ResNet. It allows the input to be passed through directly (via a shortcut connection) and added to the output of a series of convolutional layers. This helps in preserving the gradient during backpropagation, addressing the vanishing gradient problem, and enabling the training of deeper networks.

Convolutional Block: The convolutional block is another crucial component of ResNet. It is like the identity block but includes a convolution operation in the shortcut path to change the dimensions of the input to match the output. This allows for down sampling and expanding the depth of the network.

Create Model: It includes multiple stages, each with convolutional and identity blocks to extract features at different levels of abstraction. The network ends with global average pooling and dense layers for classification.

Explanation of Adaptations

- Reduced Number of Layers: The model has fewer layers compared to ResNet50 to reduce computational complexity.
- Dropout Layers: Dropout layers are added to prevent overfitting.
- Global Average Pooling: This layer is used instead of flattening, which reduces the number of parameters and helps in generalization.
- Simplified Blocks: Each stage includes only a few blocks, balancing between model depth and computational feasibility.

This custom model retains the essential elements of the ResNet architecture, such as residual connections and batch normalization, but adapts them to a smaller scale suitable for limited computational resources. It is designed to handle smaller input sizes and fewer parameters, making it practical for use cases where computational resources are a constraint. The model should perform well on tasks like image classification for smaller datasets.

The custom ResNet32-inspired neural network, designed for image classification on 28x28 grayscale images, demonstrates promising accuracy on test data, achieving 78%. However, its performance drops notably when applied to collected data, reaching only 52% accuracy for shape classification. The model's performance on sign classification is also noteworthy. On the test data, the model achieves an accuracy of 58% for sign classification. However, its performance decreases substantially to 26% accuracy when applied to collected sign data. The model seems to struggle with generalization, indicating a gap between its performance on test data and real-world collected data.

ResNet50 Model:

ResNet50 is a deep convolutional neural network architecture that consists of 50 layers. It is renowned for its effectiveness in training very deep neural networks by introducing skip connections or shortcuts. The ResNet50 model used in our experiment is pretrained on a large dataset, likely ImageNet, which enables it to learn rich and generalizable features from a diverse range of images.

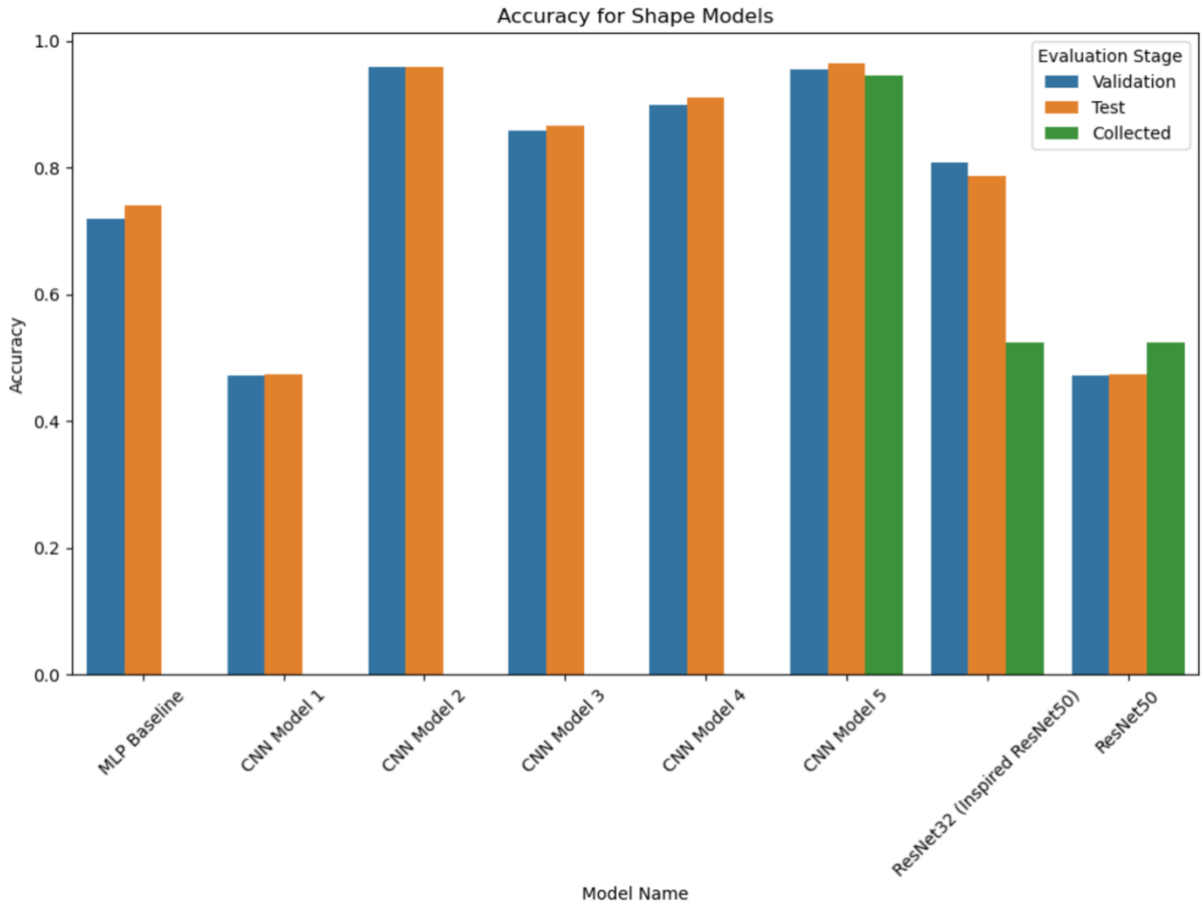
Comparison for both models **ResNet50 and ResNet50-Inspired (Custom)**:

- **Shape Classification:**
  - Custom Model: Test Accuracy - 78%, Collected Data Accuracy - 52%
  - Pretrained ResNet50: Test Accuracy - 47%, Collected Data Accuracy - 52%

- **Sign Classification:**
  - Custom Model: Test Accuracy - 58%, Collected Data Accuracy - 26%
  - Pretrained ResNet50: Test Accuracy - 18%, Collected Data Accuracy - 26%

In summary, our custom ResNet32-inspired model exhibits superior performance compared to the pretrained ResNet50 model on both shape and sign classification tasks, particularly on test data. However, there's no significant difference in performance on collected data between the two models, suggesting the need for further optimization in generalization to real-world scenarios.

Visualizing the accuracy of all models: The visualization of accuracy for all models offers a comprehensive overview of their performance. It reveals the range of accuracies across different models, highlighting any outliers that significantly deviate from the norm. By tracking temporal trends, we can discern patterns in accuracy over time, informing decisions on model selection and refinement. Additionally, analyzing the relationship between model complexity and performance sheds light on whether more intricate models consistently yield higher accuracies. Ensuring consistency across various datasets is crucial, as robust models should perform reliably in diverse scenarios.



The visualization indicates that the CNN Optimized model stands out for its strong performance. Its high accuracy suggests that the optimization strategies applied to the convolutional neural network have effectively enhanced its predictive capabilities. This underscores the importance of fine-tuning model architectures and parameters to achieve optimal results. Further analysis of the CNN Optimized model's performance could provide valuable insights into the specific techniques or features contributing to its success, guiding future model development and optimization efforts.