Project Report On

Traffic Sign Recognition on GTSRB dataset using CNNs and

Transfer Learning



Submitted in partial fulfillment for the award of Post Graduate Diploma in Big Data Analytics (PGDBDA)

From KnowIT(Pune)

Guided by:

Trupti Joshi Ma’am & Prasad Deshmukh Sir

Submitted By:

Omkar Rajale (230943025034)

Ujwala More (230943025051) Shantanu Aher (230943025055)

## CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Omkar Rajale (230943025034)

Ujwala More (230943025051) Shantanu Aher (230943025055)

Have successfully completed their project on

Traffic Sign Recognition on GTSRB dataset using CNNs and

Transfer Learning

Under the guidance of Trupti Joshi Ma’am and Prasad Deshmukh sir

## ACKNOWLEDGEMENT

This project Traffic Sign Recognition on GTSRB dataset using CNNs and Transfer Learning

was a great learning experience for us and we are submitting this work to CDAC KnowIT (Pune).

We all are very glad to mention the name Trupti Joshi Ma’am and Prasad Deshmukh Sir for his valuable guidance to work on this project. Her guidance and support helped us to overcome various obstacles and intricacies during the course of project work.

We are highly grateful to Mr. Vaibhav Inamdar Manager (KnowIT), CDAC, for his guidance and support whenever necessary while doing this course Post Graduate Diploma in Big Data Analytics (PGDBDA) through CDAC ACTS, Pune.

Our most heartfelt thanks goes to Mr. Prasad Deshmukh (Course Coordinator, PGDBDA) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day here in CDAC KnowIT, Pune.

TABLE OF CONTENTS

ABSTRACT

1. INTRODUCTION
2. DATA COLLECTION AND FEATURES
3. SYSTEM REQUIREMENTS
   1. Software Requirements
   2. Hardware Requirements
4. FUNCTIONAL REQUIREMENT
5. ARCHITECTURE
6. DEEP LEARNING ALGORITHMS
7. CONCLUSION AND FUTURE SCOPE
8. REFERENCES

**Abstract**

This project focuses on the development and implementation of a Convolutional Neural Network (CNN) and transfer learning for the classification of traffic signs from images. With the help of deep learning, our model aims to accurately detect and classify various types of traffic signs, including speed limits, stop signs, yield signs, and more.

The developed solution will be used to integrate into autonomous vehicles and real-world traffic management systems, contributing to improved road safety and efficiency.

### INTRODUCTION

The German Traffic Sign Recognition Benchmark (GTSRB) dataset serves as a crucial resource for advancing traffic sign recognition technology. This dataset encompasses a diverse array of images depicting various traffic signs encountered on German roadways, making it an invaluable asset for training and evaluating machine learning models. In our project, "Traffic Sign Recognition Using Deep Learning: A Study on the GTSRB Dataset," we delve into the realm of computer vision and machine learning, aiming to develop robust models capable of accurately identifying and classifying traffic signs in real-world scenarios. By harnessing the power of deep learning algorithms and leveraging the complexities of the GTSRB dataset, we seek to contribute to the evolution of traffic sign recognition systems, ultimately enhancing road safety and efficiency in transportation networks. Through meticulous experimentation and analysis, we aim to uncover insights that can inform the development of more intelligent and adaptive traffic management solutions, propelling innovation in this critical domain.

# Dataset Collection and Features

#### Data Sources

We used the German Traffic Sign Recognition Benchmark (GTSRB) dataset for our project, which is a popular tool for evaluating traffic sign recognition systems. This dataset was acquired via Kaggle, a well-known website that hosts datasets and competitions for data research. A wide range of photographs representing different traffic signs that are frequently seen on German roads are included in the GTSRB dataset. Supervised learning tasks are made possible by the labelling of each image with the appropriate traffic sign class. The collection offers a realistic depiction of traffic sign photos, complete with various lighting, atmospheric, and perspective circumstances. Our goal is to use this dataset to create and assess

Deep learning models for autonomous recognition of traffic signs. We were able to obtain a common and widely-used benchmark dataset by importing the dataset from Kaggle.

#### Dataset Size

* The German Traffic Sign Benchmark is a widely used benchmark multi-class, single-image classification dataset.
* GTSRB dataset consists of:
  + More than 40 classes
  + More than 50,000 images in total
  + The dataset is split into a training set of 39,209 images and a test set of 12,630 images.

#### Features/Attributes

Here is an overview of the key features (attributes) within our dataset: 1**. Speed Limit**:

Attributes:

It contains various speed limits like 20km/h, 30km/h, 50km/h, 60/h, 70/h, 80/h, 100km/h, 120km/h,

“End of Speed Limit (80km/h) ”

**2. Roads:**

Attributes :

It contains various sign for various roads.

**3. Direction:**

Attributes:

The traffic sign displaying the right and left direction sign are included.

**4.Crossings:**

Attributes:

Various crossing signs are mentioned in it.

**5.Caution/Dangerous:**

Attributes:

The signs which depicts caution, danger or gives the warnings are included.

**6. Traffic Signals:**

Attributes:

It includes the traffic signals.

2. **SYSTEM REQUIREMENTS**

**Hardware Requirements**

1. Computer: A computer with sufficient processing power and memory to run data processing and analysis tasks. A modern multicore processor and at least 8 GB of RAM are recommended.

2.Storage: Adequate storage space to store the generated dataset and any additional datasets if required. An SSD (Solid State Drive) is recommended for faster data access.

3.Internet Connection: A stable internet connection for downloading and installing software packages

and libraries, as well as for any online resources needed during the project.

**Software Requirements**

1. Operating System: Windows 10 or higher
2. Python: The project heavily relies on Python for data generation, analysis, and machine learning. Ensure Python is installed on your system.
3. Python Libraries: Install the following Python libraries and dependencies using package managers like pip

or conda:

NumPy: For numerical computing.

pandas: For data manipulation and analysis.

Matplotlib and Seaborn: For data visualization.

Tensorflow: For numerical computations and large-scale machine learning.

Keras: Designed to provide a user-friendly and efficient interface for building and training neural networks.

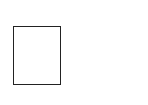
Gradio: Build web-based user interfaces (UIs) for your machine learning models, APIs, or any arbitrary

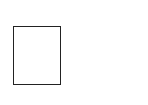
Python function

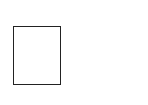
Other libraries specific to our project's needs.

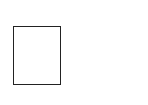
### 3. FUNCTIONAL REQUIREMENTS

Python 3:

 Python is a general purpose and high level programming language.

 It is use for developing desktop GUI applications, websites and web applications.

 Python allows to focus on core functionality of the application by taking care of common programming tasks.

 Python is derived from many other languages, including ABC, Modula3, C, C++, Algol68, Small Talk, and Unix shell and other scripting languages.

# ARCHITECTURE

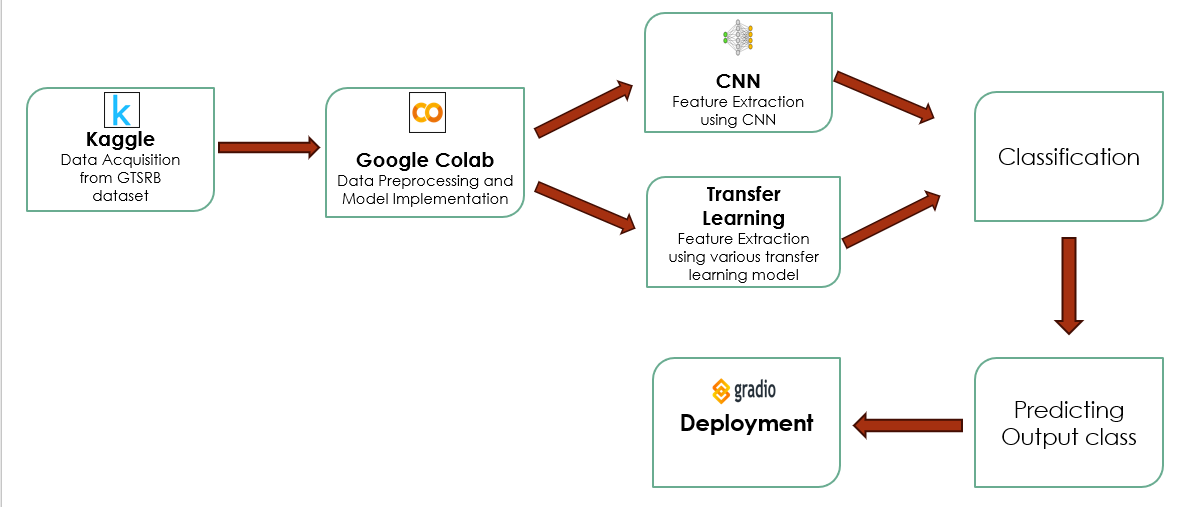
****

Fig: System Architecture for German Traffic Sign Recognition System

# DEEP LEARNING ALGORITHMS

#### Basic CNN:

**Overview:**

This system harnesses the power of Convolutional Neural Networks (CNNs) to automatically recognize and understand German traffic signs. Imagine driving down the road and having your car instantly identify speed limits, stop signs, or directions, enhancing safety and navigation.

**Explanation:**

Convolutional neural networks(CNN) is a powerful deep learning algorithm capable of dealing

with millions of parameters and saving the computational cost by inputting a 2D image and

convolving it with filters/kernel and producing output volumes.



**CNN layers**

The CNN architecture consists of a number of layers (multi-building blocks)

**Convolutional Layer:**

In CNN architecture, the most significant component is the convolutional layer. It consists of a

collection of convolutional filters (kernels). The input image, expressed as N-dimensional metrics,

is convolved with these filters to generate the output feature map.

**Kernel :**

A grid of discrete numbers or values describes the kernel. Each value is called the kernel weight. Random numbers are assigned to act as the weights of the kernel at the beginning of the CNN training process.

Thus, the kernel learns to extract significant features.

Convolutional Operation: Initially, the CNN input format is described. The vector format is the input of the traditional neural network, while the multi-channeled image is the input of the CNN. RGB image format is three-channeled.

**2. Pooling Layer:**

Pooling layer in a Convolutional Neural Network (CNN) is a critical component that reduces the spatial dimensions (i.e., width and height) of the input volume for the next convolutional layer. It does this while retaining the most important information. Pooling helps in decreasing the computational power required to process the data through dimensionality reduction. Furthermore, it also helps in extracting dominant features which are rotational and positional invariant, thus aiding in the process of classification.

There are several types of pooling, but the two most common are:

**Max Pooling:** In these pooling, where the maximum element is selected from the region of the feature

map covered by the filter. By doing so, it captures the most prominent feature in the particular patch of the input.

**3.Activation Layer**

An activation layer in a CNN is a layer that serves as a non-linear transformation on the output of the convolutional layer. It is a primary component of the network, allowing it to learn complex relationships between the input and output data.

The activation layer can be thought of as a function that takes the output of the convolutional layer and maps it to a different set of values. This enables the network to learn more complex patterns in the data and generalize better.

**ReLu:** It is the most commonly used activation function in most convolutional networks. It is a non-linear transformation that outputs 0 for all negative values and the same value as the input for all positive values. This allows the network to imbibe more complex patterns in the data.

Formula:

f(x) = max(0,x)

Range: [0,inf)

**Sigmoid:** It is another commonly used activation function, which outputs values between 0 and 1 for any given input. This helps the network to understand complex relationships between the input and output data but is more computationally expensive than ReLu.

Formula:

f(x) = 1 / (1 + exp(-x))

Range: (0,1)

**Tanh:** It is the least commonly used activation function, which outputs values between -1 and 1 for any given input.

**Softmax:** Used in the output layer of a neural network model for multi-class classification tasks. It converts the output scores from neurons into probabilities by taking the exponential of each output and then normalizing these values by dividing by the sum of all the exponentials.

**Optimizers:**

Optimizers are a vital part of the training process for deep learning models. They are responsible for adjusting the model's parameters, such as weights and biases, in order to minimize the loss function and improve performance.

**Adam:**

Combines the strengths of both momentum and adaptive learning rate concepts.

Momentum: Accumulates past gradients to smooth out updates and avoid getting stuck in local minima.

Adaptive learning rate: Adjusts the learning rate for each parameter individually based on its historical gradients.

**RMSProp:**

Addresses the issue of vanishing gradients in SGD, which can slow down learning in some scenarios.

Maintains a running average of squared gradients for each parameter.

Divides the learning rate by the square root of the average squared gradient to prevent large updates for parameters with consistently large gradients.

**4.Fully Connected Layers:**

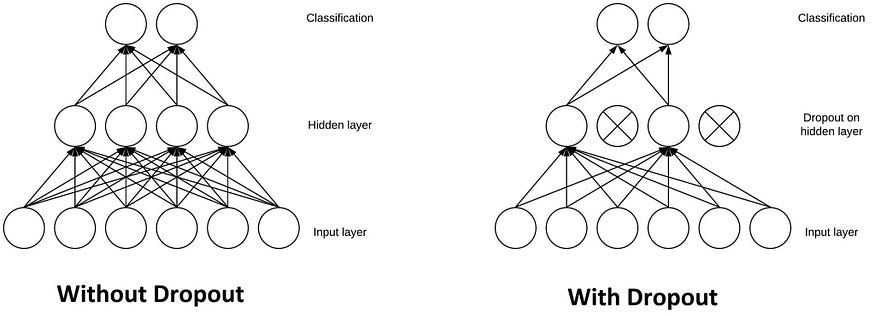
The Fully Connected (FC) layer is a critical component of Convolutional Neural Networks (CNNs) and many other types of neural networks. Positioned after the convolutional and pooling layers in a CNN, the FC layer plays a pivotal role in the network's ability to make predictions and classifications based on the extracted features.

The fully connected layer integrates various features extracted in the previous convolutional and pooling layers and maps them to specific classes or outcomes. Each input from the previous layer connects to each activation unit in the fully connected layer, enabling the CNN to simultaneously consider all features when making a final classification decision.

Not all layers in a CNN are fully connected. Because fully connected layers have many parameters, applying this approach throughout the entire network would create unnecessary density, increase the risk of overfitting and make the network very expensive to train in terms of memory and compute.

**5.Dropout:**

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model’s performance when used on a new data. To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3/ 30% of the nodes are dropped out randomly from the neural network.



**Output Layer:**

The Output Layer is the final layer in a neural network architecture and is responsible for producing the model's output, making it crucial for achieving the specific objectives of a given task, such as classification, regression, or any other predictive modelling. This layer takes the processed information from the previous layers (which have gone through various transformations via convolutional, pooling, and fully connected layers, among others) and converts it into a form that matches the desired output format of the problem being solved.

**Here's how it works:**

**Images as Input:** Photos of traffic signs are fed into the system.

**CNN Feature Extraction:** The CNN analyzes the image, extracting visual patterns and key features.

**Sign Recognition:** Based on the learned features, the system identifies the specific sign and its meaning.

**Benefits:**

**Accuracy and Speed:** CNNs excel at recognizing complex patterns, leading to highly accurate and fast

sign identification.

**Robustness:** The system can handle variations in lighting, weather, and sign conditions.

**Scalability:** Adaptable to different types of signs and applications.

**Transfer learning Models**

Transfer learning is a technique in machine learning where a pre-trained model on one task is used as a starting point for a new, related task. It's like building on existing knowledge to learn something new faster and more efficiently.

Key idea:

**1.Train a model on a large, general task**: Imagine training a model to recognize objects in millions of images. It learns essential features like edges, shapes, and textures.

**2.Use that model as a starting point:** Now, train a different model to recognize specific objects like cats in new images. You only need to fine-tune the last layers, leveraging the pre-trained features from the first model.

Transfer learning models are pre-trained on a huge dataset that is Imagenet

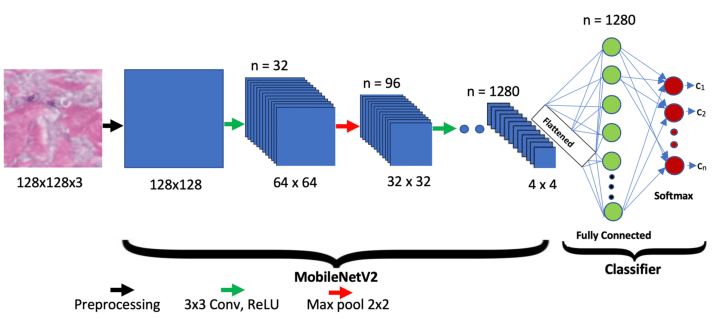
ImageNet, with its 1.4 million images and 1000 object categories, provides a strong foundation for transfer learning in image recognition tasks.

**Advantages of transfer learning**   
  
• Quicker training: Using pre-trained weights cuts down on training time considerably as compared to building a model from the ground up, particularly for difficult jobs requiring a little amount of data.

• Superior performance: Transfer learning frequently results in superior performance on the intended task, particularly when working with tiny datasets where it may not be possible to train from start.   
  
• Lower resource requirements: Pre-trained models can be trained on hardware with less capacity since they can use less memory and compute resources.

**MobileNetV2**

MobileNet V2 is a lightweight convolutional neural network (CNN) architecture well-suited for transfer learning applied to the German Traffic Sign Recognition dataset.



The characteristics are:

* Designed for mobile and embedded devices: Employs depth-wise separable convolutions and linear bottleneck layers to achieve high accuracy with reduced computational cost and memory footprint.
* Pre-trained on ImageNet: Like other transfer learning models, MobileNet V2 leverages pre-trained knowledge from the vast ImageNet dataset, learning fundamental visual features.

Transfer Learning for Traffic Sign Recognition:

* Freeze initial layers: The pre-trained weights of the initial layers in MobileNet V2 are frozen, preserving their generic feature extraction capabilities.
* Fine-tune final layers: The final layers of the network are either modified or replaced with new layers specific to the German Traffic Sign Recognition dataset (speed limit, roads, directions, etc.).
* Training on German Traffic Sign Recognition dataset: The modified MobileNet V2 is then trained on the Intel dataset, allowing the final layers to adapt the learned features for classifying these specific image categories.

**Benefits:**

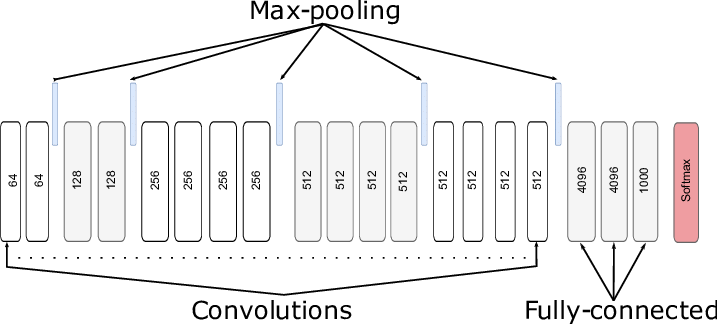
**Speed and Efficiency:** Runs even complex tasks smoothly on mobile devices and resource-constrained environments.

**Accuracy:** Delivers excellent image recognition performance comparable to larger models.

**Flexibility:** Adaptable to various tasks and domains through fine-tuning or transfer learning.

**VGG19:**

VGG19 is a powerful CNN architecture with 19 layers from which 16 are the convolutional layers and 3 fully connected layers known for its high accuracy in image recognition tasks. Imagine achieving top-notch performance in identifying objects, classifying scenes, or understanding content in images - that's what VGG19 excels at. However, its large size and complex structure make it less suitable for resource-constrained environments like mobile devices.



Transfer Learning for Traffic Sign Recognition:

* Freeze initial layers: The initial layers of the VGG19 architecture, which capture basic features like edges and textures, are frozen. By freezing these layers, their weights remain fixed during the subsequent training on the German Traffic Sign Recognition dataset.
* Fine-tune final layers: The final layers of the VGG19 model, responsible for making predictions, are either modified or replaced with new layers suitable for the German Traffic Sign Recognition task. These final layers are then fine-tuned during training on the German Traffic Sign Recognition dataset while keeping the initial layers frozen.
* Training on German Traffic Sign Recognition dataset: The modified VGG19 model, with its initial layers frozen and final layers fine-tuned, is trained on the German Traffic Sign Recognition dataset. During training, the model learns to classify images of traffic signs into their respective categories, leveraging the pre-trained knowledge from ImageNet and adjusting its predictions based on the characteristics of the traffic sign dataset.

**Benefits:**

• **Accuracy:** Achieves state-of-the-art performance in various image recognition tasks.

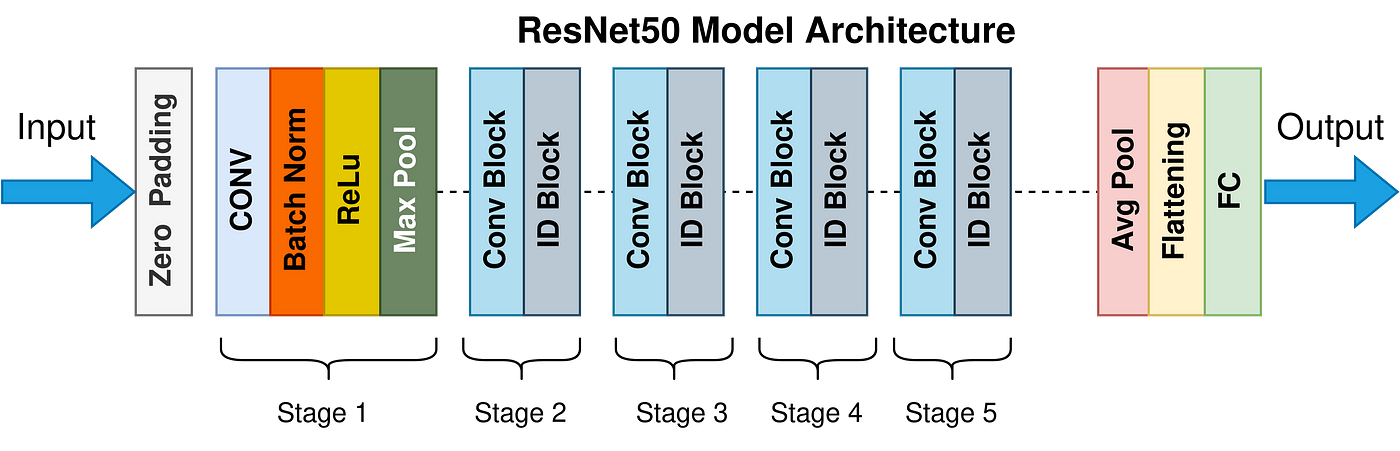
• **Flexibility:** Adaptable to diverse applications through transfer learning and fine-tuning.

• **Well-understood:** Extensive research and documentation make it a popular choice for researchers and developers.

**ResNet50:**

**Overview:**

The key idea behind ResNet-50 is the incorporation of residual blocks, which enable the training of very deep networks by addressing the vanishing gradient problem. ResNet-50 has 48 convolutional layers along with additional layers for handling pooling, normalization, and fully connected operations.



Freeze Initial Layers: The initial layers of the ResNet50 architecture, which capture basic features like edges and textures, are frozen. By freezing these layers, their weights remain fixed during the subsequent training on the German Traffic Sign Recognition dataset. This ensures that the model retains the general feature extraction capabilities learned from ImageNet.

Fine-Tune Final Layers: The final layers of the ResNet50 model, responsible for making predictions, are either modified or replaced with new layers suitable for the German Traffic Sign Recognition task. These final layers are then fine-tuned during training on the German Traffic Sign Recognition dataset while keeping the initial layers frozen. Fine-tuning allows the model to adapt its learned features to better recognize and classify traffic signs.

Training on German Traffic Sign Recognition Dataset: The modified ResNet50 model, with its initial layers frozen and final layers fine-tuned, is trained on the German Traffic Sign Recognition dataset. During training, the model learns to classify images of traffic signs into their respective categories, leveraging the pre-trained knowledge from ImageNet and adjusting its predictions based on the characteristics of the traffic sign dataset.

**Benefits:**

**Accuracy:** Achieves state-of-the-art performance in various image recognition tasks.

**Efficiency:** Skip connections and bottleneck layers reduce computational cost compared to similar-depth networks.

**Flexibility:** Adaptable to diverse applications through transfer learning and fine-tuning.

**CONCLUSION AND FUTURE SCOPE**

**Conclusion: Comparison of CNN , Transfer Learning models for German Traffic Sign Dataset**

Our findings reveal that the Basic Convolutional Neural Network (CNN) outperformed the Transfer Learning models, achieving an impressive accuracy of 94.16% in predicting traffic signs. This result demonstrates the effectiveness of the CNN architecture in this specific context.

The superiority of the Basic CNN model can be attributed to several factors. Firstly, the simplicity and straightforwardness of the CNN architecture allowed it to effectively capture the relevant features and patterns present in the traffic sign images. Additionally, by training the CNN model specifically on the German Traffic Sign Dataset, it could tailor its learned representations to the nuances and characteristics of the dataset, leading to improved performance.

While Transfer Learning models such as VGG19, MobileNetV2, and ResNet50 are powerful architectures pre-trained on large-scale datasets like ImageNet, our experimentation revealed that they were not as well-suited for the German Traffic Sign Dataset as the Basic CNN model. Despite their sophisticated architectures and feature extraction capabilities, these models did not surpass the performance of the CNN model in this scenario.

In conclusion, our project highlights the importance of selecting the appropriate model architecture for a given dataset and task. While Transfer Learning models offer significant advantages in certain contexts, the results of our study underscore the effectiveness of Basic Convolutional Neural Networks for the German Traffic Sign Dataset. This insight provides valuable guidance for future research and applications in traffic sign recognition and related fields.

**Inferential Outcomes:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **CNN** | **VGG19** | **MobileNetV2** | **ResNet50** |
| Training Accuracy | 0.9960 | 0.9960 | 0.9968 | 0.6577 |
| Testing Accuracy | 0.9416 | 0.9240 | 0.7734 | 0.3832 |

### Future Scope:

**Enhanced Speed and Efficiency:** Future iterations of the system should focus on improving speed and efficiency to ensure real-time performance even in high-traffic scenarios. This could involve optimizing algorithms, leveraging hardware acceleration (e.g., GPUs, FPGAs), and exploring novel architectures designed specifically for real-time inference.

**Integration with Autonomous Vehicles:** As autonomous vehicle technology continues to evolve, integrating real-time traffic sign recognition into autonomous driving systems will be crucial. This integration can enhance vehicle safety by enabling autonomous vehicles to accurately perceive and respond to traffic signs in real-world driving scenarios.

**Integration with Traffic Management Systems:** Integrating real-time traffic sign recognition with existing traffic management systems can enable more efficient traffic flow and enhanced safety measures. The system can provide valuable insights to traffic authorities by monitoring and analyzing traffic sign compliance and identifying potential hazards in real-time.

**References**

1. https://towardsdatascience.com/top-3-python-packages-to-generate-synthetic-data33a351a5de0c

1. scikit-learn. [https://scikit-learn.org/]

1. Confluent Python Client. [https://docs.confluent.io/platform/current/clients/confluentkafka-python/html/index.html]
2. Kaggle

[<https://www.kaggle.com/datasets/puneet6060/intel-image-classification/code>]

1. Google colab

[https://colab.google/ ]

1. Tenserflow

https://www.tensorflow.org/api\_docs