Enhancing Autonomous Navigation with Few-Shot Traffic Sign Recognition

SATVIK V[1], ROHIT M[1], NAVEEN K [1] AND ILAVENDHAN A[*]
School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai 600127, India

Corresponding author: Ilavendhan A

ABSTRACT Traffic sign recognition is vital for autonomous driving and road safety. However, traditional deep learning models face challenges due to changes in light, occlusion, and the implementation of new or rare road signs. Furthermore, deep learning models require large labeled datasets and can take significant time to retrain a model for previously unobserved signs. While traditional deep learning models are ineffective for real-time traffic sign recognition, we explore the use of Few-Shot Learning (FSL) through Prototypical Networks as an alternative to classifying traffic signs based on their similarity. Traffic signs will be classified with our models using ConViT, EfficientVIT, ResNet18, MobileNetV3, and CNN models. We evaluated our FSL approach using the German Traffic Sign Recognition Benchmark (GTSRB) dataset, achieving the best accuracy for training of 99.98% (MobileNetV3) and testing of 97.46%(ResNet18). In contrast to traditional means, our model can classify newly introduced signs for the first time without needing retraining. Our model is adaptable to real-world situations and eliminates the need for additional datasets and training, lowering costs and dependency on additional datasets while retaining high performance. FSL is a potentially scalable and efficient mode of operation for autonomous vehicles, allowing us to implement new signs into the model without having to gather extensive datasets or retrain the model with respect to the new sign.

INDEX TERMS Traffic Sign Recognition, Few-Shot Learning, Autonomous Navigation, Intelligent Transportation Systems

I. INTRODUCTION

Autonomous navigation has emerged as a key area of development in modern transportation systems, enabling vehicles to operate safely and independently with minimal or no human input. Self-driving vehicles use multiple technologies including: sensors, artificial intelligence, and data analysis to engage with the world around it and make choices in reaction. One of those technologies is computer vision which allows the vehicle to interpret visual cues from the environment, including road markings, vehicles, pedestrians, and traffic signs. Computer vision is the backbone of several various perception tasks that must be interpreted to drive safely with autonomy [1][2].

Traffic sign recognition is one of the most important computer vision problems in this context. Traffic signs carry valuable information about the speed limit, warnings, regulations, and directions that are crucial for guiding vehicle behaviour. For fully autonomous systems and driver-assistance systems to navigate the world safely and legally, it is necessary for the system to recognize and understand traffic signs. An example of a deep learning-based traffic sign recognition system is shown in Figure (1) , where the model processes the input image and predicts the appropriate sign class. Accurate and reliable identification of traffic signs is increasingly valuable as we approach a time when autonomous vehicles are operating in the real world [3].

Although the advancement in the research field of computer vision has improved many possibilities for traffic sign detection, traffic sign recognition still faces many challenges in real world situations. One challenge is the variance in environmental conditions: traffic signs are often photographed under very poor lighting, or shaded. Signs may also be blurred by rain, fog, or motion, which can limit a model's performance. There may also be partial occlusion of signs like a vehicle, tree, or infrastructure may obscure the sign face, complicating the detection and recognition process. Research has shown that models trained on clean images, or ideal conditions, often do not overgeneralize when presented with these more difficult conditions after deployment [4]. Another challenge associated with traffic sign recognition is the number of global designs of signs used in different countries and regions of the world. A model trained on one country's dataset may not recognize a sign because of a symbol or color used in a different nation's traffic signs, limiting the potential for deployment, for example, across borders [5].

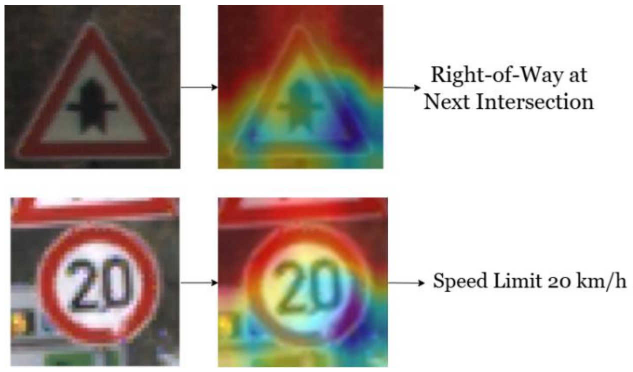

FIGURE 1. **Example of traffic sign recognition using a deep learning model.**

In addition to these difficulties, class imbalance in datasets remains a challenge that will continue to affect many machine learning approaches. Some signs like speed limits or stop signs are often abundant in datasets, whereas others, such as warning signs or construction zone signs are sparsely collected, leading to models with poor performance levels on the underrepresented classes. The underlying class imbalance affects model training stability and prediction accuracy when used in real-world applications. This is also evidenced in the literature whereby existing works often still rely heavily on vast quantities of labeled data based on traditional deep learning literature practices, and struggle to generalize across unseen or classes that are rarely seen [6]. These issues point more towards the need for more data-efficient, adaptive, or even continual learning in the presence of scarcity and variability in training data.

Traffic sign identification methods based on deep learning, like CNNs or vision transformers, commonly depend on very large, balanced, and well-annotated datasets. They can obtain high accuracy in controlled settings with enough training images for each class, but their accuracy will often drop substantially when they encounter infrequent signs, previously unseen classes, or domain shifts from environmental or geographic changes. The requirement for thousands of labeled images for each class can make it very difficult for traffic sign identification systems to be deployed in locations where there are not many readily available or easily accessible images of traffic signs. As Khan et al. point out, this reliance on data limits the scalability and flexibility of deep learning systems in rapidly changing, dynamic, real-world scenarios [7].

To address these limitations, one of the most promising methods is Few-Shot Learning (FSL). The goal of FSL is to train models that can generalize to new classes with only a limited number of labeled examples. Instead of collecting an exhaustive amount of data, FSL methods try to learn transferable representations or similarity-based matching approaches that enable learning of new tasks rapidly. Such ability can be useful in traffic sign recognition, where the ubiquitous experience of encountering an unfamiliar sign or a newly introduced sign is prevalent. Recent studies have explored how FSL methods could reduce the amount of labeled data required, while maintaining strong performance on classification tasks, by integrating FSL methods with visual recognition architectures [8][9]. These studies suggest that FSL can support the goal of making traffic sign recognition systems more data efficient, adaptable, and robust for autonomous vehicles.

The objective of this research is the usability of a traffic sign recognition system that can achieve high accuracy with low training data. In particular, the study concentrates on the feasibility of adapting several Few-Shot Learning methodologies with deep learning-based models to facilitate the real-time classification of traffic signs. Various models are examined, including ResNet18, MobileNetV3, EfficientNetV2, and ConVIT which were all trained and validated using the German Traffic Sign Recognition Benchmark (GTSRB) dataset. The few-shot learning scenarios are simulated with limited labeled examples maintained for certain classes, specifically noting the model performance for rarer and underrepresented signs. Broadly, this work aims to provide a lightweight and flexible recognition paradigm which is generalizable over many different environments eventually providing safety and scaling solutions for autonomous navigation systems.

II. RELATED WORKS

Traffic sign detection and classification have been extensively studied using various deep learning and transformer-based models. Recent advancements have significantly improved the accuracy, robustness, and real-time processing capabilities of these models, making them more suitable for autonomous driving and intelligent transportation systems. By leveraging sophisticated feature extraction techniques and optimized training methodologies, these models have demonstrated enhanced adaptability to various environmental conditions, including changes in lighting, weather, and occlusions.

A. TRANSFORMER-BASED APPROACHES

Vision Transformers have shown promising performance in traffic sign classification tasks, as demonstrated by Zheng and Jiang [20] , who evaluated their effectiveness compared to traditional convolutional models. Kaleybar et al. [10] proposed an Efficient Vision Transformer for traffic sign detection, integrating a locality inductive bias with a transformer module to improve both computational efficiency and detection accuracy. Their study explored several transformer-based architectures, including Pyramid Vision Transformer (PVT), Swin Transformer, and Locality in Locality (LNL) models, while using ResNet as a baseline. The integration of hierarchical feature learning with self-attention mechanisms enabled these models to improve feature extraction and robustness in diverse conditions, making them highly suitable for real-world traffic sign recognition tasks. Similarly, Farzipour et al.[11] introduced a hybrid approach that combined Convolutional Neural Networks (CNNs) with Vision Transformers, leveraging the local feature extraction strengths of CNNs alongside the global dependency learning capabilities of transformers.

Their model was rigorously evaluated on both the German Traffic Sign Recognition Benchmark (GTSRB) and the Persian Traffic Sign Dataset (PTSD), showcasing enhanced computational efficiency, improved sign recognition accuracy, and superior generalization to unseen traffic sign categories. Additionally, Mingwin et al.[19] explored the potential of Vision Transformers for traffic sign recognition, highlighting their ability to preserve spatial information while capturing high-level semantic details, thus improving classification accuracy. Further, Nejati Manzari et al. [24] introduced a Pyramid Transformer incorporating locality mechanisms to enhance the efficiency of traffic sign detection, addressing issues related to small object detection and occlusions. Furthermore, Gan et al.[25] proposed a cross-domain few-shot in-context learning method, enhancing traffic sign recognition performance using multimodal learning approaches, thereby enabling models to generalize better across different traffic sign datasets and domains.

B. YOLO-BASED OBJECT DETECTION APPROACHES

YOLO-based models have exhibited remarkable performance in real-time traffic sign detection due to their efficient single-shot detection framework. Kaur et al.[12] proposed an improved YOLOv8 model for traffic sign detection, optimizing the model's multi-scale detection capabilities by refining anchor box selection and enhancing

TABLE I
COMPARISON OF TRAFFIC SIGN DETECTION MODELS

Reference no	MODEL	Real-time capable	Handles small signs	Multi-scale detection	Handles occlusion	Lightweight model	Few-shot learning	Key features	Datasets used
[10]	Efficient ViT	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	Vision Transformer with locality inductive bias	GTSRB, PTSD
[11]	Local Vision Transformer	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	Transformer with locality for traffic sign detection	GTSRB
[12]	Improved YOLOv8	**Yes**	**Yes**	**Yes**	**Yes**	**No**	**No**	YOLOv8 with multi-scale detection	GTSRB, Indian TSR Dataset
[13]	YOLOv8-based Model	**Yes**	**Yes**	**Yes**	**Yes**	**No**	**No**	YOLOv8 with four-category classification	GTSRB
[14]	YOLOv5-based Detection	**Yes**	**No**	**Yes**	**Yes**	**No**	**No**	Faster YOLOv5 model with real-time detection	LISA, GTSRB
[15]	Multi-Focal YOLOv8n	**Yes**	**Yes**	**Yes**	**Yes**	**No**	**No**	Multi-focal mechanism for Indian road conditions	Indian TSR Dataset
[16]	CNN-based TSR	**No**	**No**	**No**	**No**	**Yes**	**No**	CNN-based recognition with preprocessing techniques	LISA
[17]	Deep CNN TSR	**No**	**No**	**No**	**Yes**	**No**	**No**	CNN optimized for video-based sign recognition	GTSRB
[18]	Unified Framework	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	Combines AlexNet, VGG-19, ResNet-50, EfficientNet v2	Indian TSR Dataset

[19]	Vision Transformer TSR	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	Vision Transformers for feature extraction	GTSRB
[21]	ViT-based TSR	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	ViT-based multimodal traffic sign recognition	GTSRB, Mapillary TSR
[22]	TrafficSign Net	**Yes**	**Yes**	**Yes**	**Yes**	**No**	**No**	Custom CNN optimized for real-time conditions	GTSRB, Belgian TSR Dataset
[23]	ViT-based TSR	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	Transformer-based sign classification	GTSRB, LISA
[24]	Pyramid Transformer TSR	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	Pyramid-based Transformer architecture	PTSD
[25]	Few-shot Learning with Transformer s	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	Cross-domain few-shot learning with in-context learning	GTSRB, PTSD, Mapillary TSR
[26]	Optimized CNN	**Yes**	**No**	**Yes**	**Yes**	**Yes**	**No**	CNN optimized for edge devices	GTSRB
[27]	Multi-Modal Transformer	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	Transformer with multimodal fusion	GTSRB, Mapillary TSR
[28]	Feature Combination + RF	**No**	**No**	**No**	**No**	**Yes**	**No**	Uses Random Forests with feature fusion	LISA, GTSRB
[29]	Cascaded R-CNN	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	Cascaded R-CNN with multiscale attention	GTSRB

spatial feature representation. Their comparative analysis against ResNet50 highlighted YOLOv8's superior performance in handling complex traffic scenarios with varying sign sizes and orientations. Gupta et al.[13] extended this work by implementing YOLOv8 for classifying traffic signs into four distinct categories—traffic lights, stop signs, speed limits, and crosswalks. Their study demonstrated the model's robustness under challenging lighting conditions, extreme weather variations, and occlusions caused by surrounding objects such as trees and vehicles. Ashwath et al.[14] focused on YOLOv5 for real-time detection, showing that its optimization strategies, including feature pyramid networks and advanced anchor clustering techniques, significantly improved both detection speed and accuracy over earlier YOLO versions.

For Indian road conditions, Suresha et al.[15] introduced a specialized multi-focal YOLOv8n model integrated with an attention mechanism to enhance detection in challenging environments, including low-light conditions and occluded traffic signs. This region-specific approach enabled better adaptation to the variations in Indian traffic sign structures and designs.

C. CNN-Based Traffic Sign Detection Approaches

Convolutional Neural Networks (CNNs) have remained a fundamental approach for traffic sign recognition, owing to their ability to learn hierarchical spatial representations. Jency et al. [16] developed a CNN-based traffic sign recognition system using the LISA dataset, implementing preprocessing techniques such as contouring, fit ellipse, and Laplacian of Gaussian filtering to enhance edge detection and feature extraction. Their study demonstrated that carefully selected preprocessing methods could significantly boost recognition accuracy. Urs et al.[17] proposed a deep CNN-based approach tailored for real-time sign detection in video feeds rather than static images, making it more effective for dynamic driving environments. Their model leveraged a combination of rotation, scaling, and color normalization techniques to enhance feature learning and ensure robustness across diverse lighting conditions. Additionally, a specialized deep learning model optimized for real-time traffic sign detection in autonomous vehicles was proposed by [22], focusing on improving both accuracy and processing speed through a combination of depth wise separable convolutions and efficient pooling strategies. A recent study by Li and Wang [26] proposed an optimized deep learning model tailored for traffic sign recognition in autonomous vehicles. Their approach leverages advanced neural network techniques to enhance detection accuracy while maintaining computational efficiency. The model effectively addresses challenges posed by varying environmental conditions, improving the robustness of recognition systems in real-world driving scenarios. Furthermore, Ellahyani et al. [28] proposed a feature combination method utilizing Random Forest classifiers, incorporating advanced segmentation techniques such as color thresholding and Histogram of Oriented Gradients (HOG) features to further enhance detection accuracy.

D. MULTI-MODEL AND REGION-SPECIFIC APPROACHES

Several studies have focused on combining multiple architectures to optimize traffic sign detection for specific environments, addressing regional variations in traffic sign design and road conditions. Uikey et al. [18] proposed a unified framework that integrates multiple deep learning architectures, including AlexNet, VGG-19, ResNet-50, and EfficientNet v2, for Indian traffic sign classification. Their study highlighted the challenges posed by unstructured road conditions, poor lighting, and variations in traffic sign shape and color, necessitating the need for models tailored to region-specific datasets. Another study by Zhang et al.[27] proposed a fusion-based deep learning model combining CNN, transformer, and attention mechanisms to enhance generalization across diverse traffic sign datasets, ensuring adaptability to multiple traffic environments. Additionally, Zhang et al.[29] introduced a cascaded R-CNN model with multiscale attention to improve detection accuracy for small-sized traffic signs, effectively addressing imbalanced datasets and enhancing feature extraction for low-contrast traffic signs.

These approaches have demonstrated significant improvements in various aspects of traffic sign recognition, with models being optimized for real-time performance, small sign detection, and handling occlusions. As shown in TABLE I, several models, such as EfficientViT, YOLOv8-based models, and Vision Transformers, have been developed to address specific challenges in traffic sign detection, including multi-scale detection, real-time capabilities, and few-shot learning. Each model's unique features, strengths, and the datasets used for evaluation are summarized in the table, providing a comprehensive comparison of their capabilities.

E. RESEARCH GAP AND OUR CONTRIBUTION

Despite significant advancements in deep learning techniques for traffic sign detection, many existing approaches continue to face important limitations. A major challenge is the heavy reliance on large-scale labeled datasets, which are often expensive and time-consuming to collect, especially when dealing with rare or newly introduced traffic signs. In addition, many deep learning models require substantial computational resources during training and inference, which poses difficulties for deployment on edge devices with limited processing power, such as those commonly used in autonomous vehicles. These limitations are further compounded by issues related to generalization. Models trained on region-specific datasets often struggle to perform effectively when exposed to traffic signs from different countries or environments, due to variations in sign design, color schemes, and linguistic symbols.

To overcome these challenges, this work proposes a Few-Shot Learning (FSL) approach for traffic sign classification that enables recognition of new or rare sign categories using only a limited number of labeled examples. The proposed method combines various lightweight and efficient deep learning architectures, including ConVIT, EfficientViT v2, ResNet18, and MobileNetV3. These models are chosen for their balance between performance and computational efficiency, making them suitable for real-time applications on resource-constrained devices. Additionally, the system incorporates strategies to enhance learning with minimal data, allowing for rapid adaptation to previously unseen traffic signs without the need for retraining on large datasets.

The architecture is designed to support scalable and robust performance in dynamic and diverse traffic environments. By minimizing the dependency on extensive labeled datasets and reducing computational overhead, the proposed approach aims to improve the practicality and adaptability of traffic sign recognition systems. This contributes to the broader goal of making autonomous navigation systems more reliable, accessible, and deployable across a wide range of real-world scenarios, including those with limited data availability or changing traffic regulations.

III. METHODOLOGY

This section describes the dataset used, the Few-Shot Learning framework, the architecture of Prototypical Networks, and the training strategy employed in our study.

A. DATASET DESCRIPTION

The German Traffic Sign Recognition Benchmark (GTSRB) [30] is a widely used dataset for traffic sign classification, containing 43 distinct classes of traffic signs. The dataset consists of 39,209 training images and 12,630 testing images, with images captured under diverse real-world conditions, including variations in lighting, occlusion, and perspective distortion. Each traffic sign class has a different number of samples, leading to class imbalance, which makes classification more challenging.

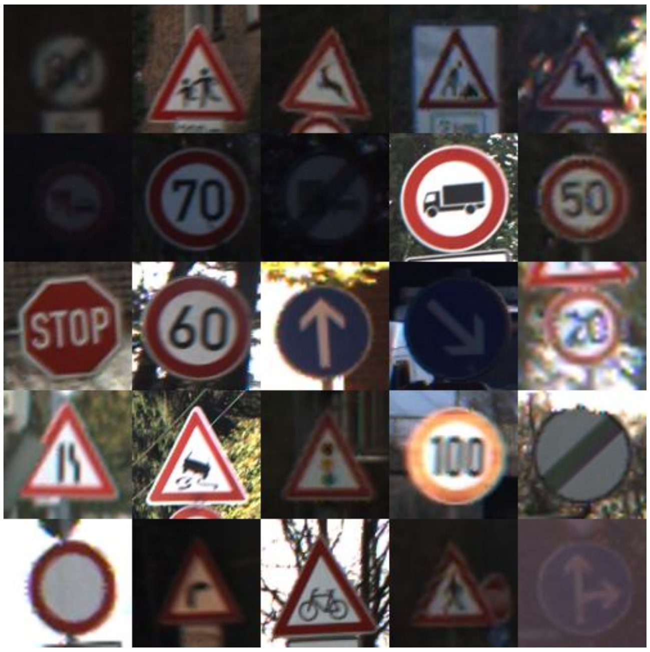

FIGURE 2. Sample Traffic Signs from the GTSRB Dataset

FIGURE 2) shows a collection of traffic signs from the German Traffic Sign Recognition Benchmark (GTSRB) dataset. The images include different types of signs, such as speed limits, warning signs, and directional arrows. Some pictures were taken in low light or shadowed areas, making the signs harder to see. Others have motion blur or different angles, which adds to the challenge of recognizing them correctly. These variations help train models to identify traffic signs accurately, even in real-world conditions where lighting and visibility may not always be perfect.

For preprocessing, all images are resized to a fixed resolution to ensure uniformity across the dataset. The images are then converted to tensors for compatibility with deep learning frameworks. Finally, normalization is applied using standard mean and standard deviation values, which helps stabilize training and improves model convergence.

B. FEW-SHOT LEARNING (FSL) APPROACH

Deep learning models have achieved remarkable success in traffic sign recognition, but they usually require large amounts of labeled training data. In real-world scenarios, this is a major limitation. Traffic signs vary across regions, and new signs are frequently introduced. Collecting and labeling thousands of images for each new sign is expensive and time-consuming. Moreover, standard deep learning models need to be retrained from scratch when new traffic signs appear, making them inefficient for real-time applications.

Few-Shot Learning (FSL) provides a more practical solution by allowing models to recognize new classes using only a few labeled examples. Instead of memorizing class-specific features, FSL-based models learn to compare and differentiate samples based on similarity. One of the most effective ways to achieve this is through metric-based learning, where the model maps images to a feature space and classifies them by comparing distances rather than using predefined class labels.

For this study, we use Prototypical Networks, a well-known metric-based FSL approach. This method works by computing a representative feature, called a prototype, for each class. When a new traffic sign appears, the model determines its class by measuring its similarity to these prototypes. This approach is particularly well-suited for traffic sign recognition since signs within the same category often share common visual patterns, making distance-based classification an efficient strategy.

C. PROTOTYPICAL NETWORKS

Prototypical Networks classify images by comparing them to class prototypes rather than learning explicit decision boundaries. The process consists of three main steps: feature extraction, prototype computation, and distance-based classification.

1) Feature Extraction: The model first processes input images through a feature extractor that learns to identify relevant patterns. These extracted features capture essential characteristics such as shape, color, and structure, which help differentiate traffic signs. Instead of directly associating images with predefined class labels, the feature extractor transforms them into compact numerical representations in a high-dimensional space. In this learned space, similar images are positioned closer together, enabling efficient classification based on similarity rather than traditional label assignment.

2) Prototype Computation: A prototype is a representative feature vector for each class, computed as the mean of all support samples belonging to that class. This prototype acts as a class reference point. Given a set of support images S_c for class c , the prototype is defined as:

$$p_c = \frac{1}{|S_c|}\sum_{x_i \in S_c} f(x_i) \qquad (1)$$

where $\boldsymbol{f(x_i)}$ is the feature embedding of image x_i , and S_c is the number of samples in class c.

As expressed in (1) , computing the prototype as the mean embedding ensures that the model captures a stable representation of each class, even when only a few labeled examples are available. This helps maintain consistency in

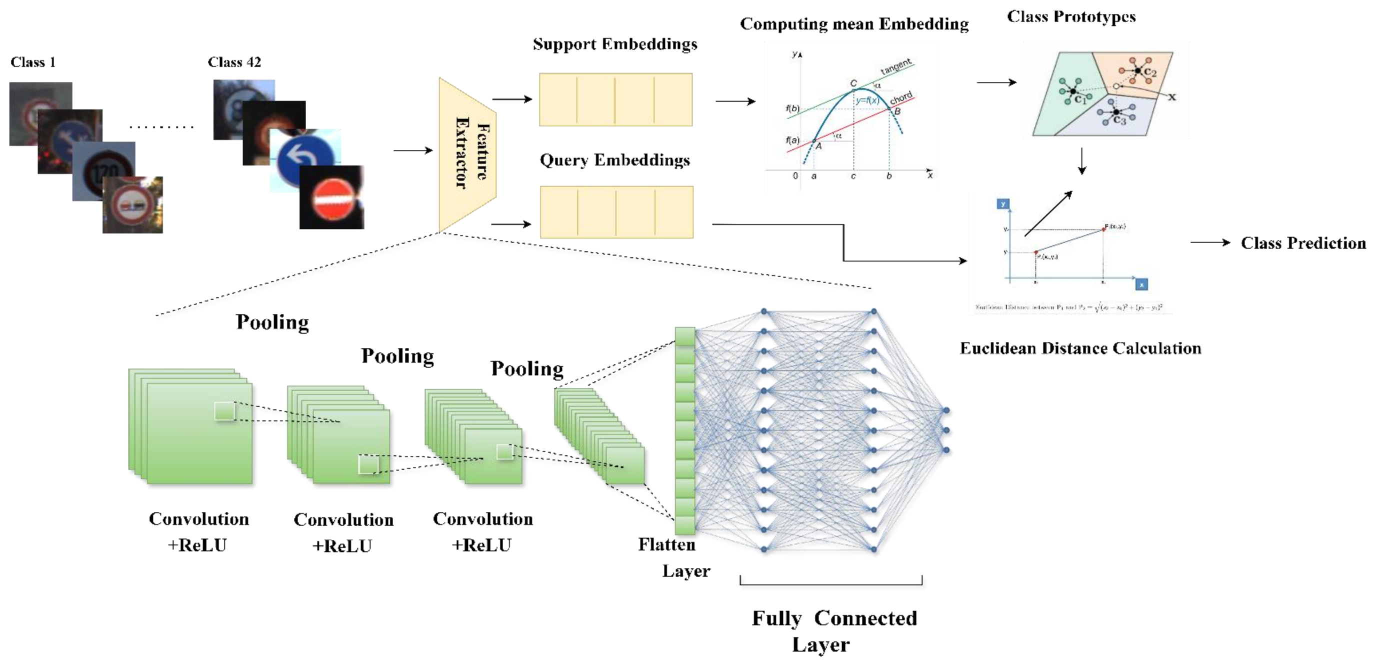

FIGURE 3. **Overview of traffic sign classification using a Few-Shot Learning (FSL) model based on Prototypical Network**

.classification and reduces sensitivity to variations in individual support samples

3) Distance-Based Classification: Once the prototypes are established, the model classifies new images by measuring their similarity to these prototypes. The classification decision is based on Euclidean distance [43], which quantifies how close a query image is to each class prototype. Given a query image x_q, the distance to a class prototype p_c is computed as:

$$d(x_q, p_c) = |f(x_q) - p_c|^2 \tag{2}$$

where $f(x_q)$ represents the feature vector of the query image, and p_c is the prototype of class c. As shown in (2) , the model assigns the query image to the class with the smallest Euclidean distance, ensuring that classification is performed based on similarity in the learned feature space.

4) Probabilistic Model for Classification: The probability of a query image x_q [42] belonging to class c is determined by applying a softmax function over the negative distances between the query sample and all class prototypes. This ensures that closer prototypes receive higher probabilities while farther ones receive lower probabilities. The probability is computed as:

$$P(y = c \mid x_q) = \frac{exp\left(-d\left(f(x_q), p_c\right)\right)}{\sum_{c'} exp\left(-d\left(f(x_q), p_{c'}\right)\right)} \tag{3}$$

where $\left(f(x_q), p_c\right)$ represents the Euclidean distance between the query sample's feature vector $\left(f(x_q), p_c\right)$ and the prototype p_c of class c. As expressed in (3) , the model assigns probabilities to each class, with the highest probability corresponding to the nearest prototype.

FIGURE 3) illustrates the process of traffic sign classification using Prototypical Networks in a Few-Shot Learning (FSL) framework. The model follows three key steps: feature extraction, prototype computation, and distance-based classification. Initially, input images are processed through a feature extractor, which in this diagram is represented by a Convolutional Neural Network (CNN). However, this step is model-agnostic and can be replaced with any suitable backbone, such as Vision Transformers (ViTs) or ResNet variants, depending on the application. The extracted embeddings are then used to compute class prototypes by averaging the feature representations of support samples. Finally, a query image is classified based on the Euclidean distance to these prototypes, assigning it to the closest class.

D. TRAINING STRATEGY

Few-Shot Learning requires a different training approach compared to standard deep learning methods. Instead of training a model to recognize all classes at once, it is trained through episodic learning, where each episode simulates a small classification task. This method helps the model generalize better to unseen classes by repeatedly learning how to differentiate between a limited number of categories using only a few examples.

1) Episodic Training Framework (N-way, K-shot Setup): In episodic training, each iteration presents the model with a small set of randomly chosen classes (N-way classification) [39] . Within each class, only a few labeled examples (K-shot learning) [40] are provided as support samples. The model uses these labeled samples to form class representations and then classifies a set of query images based on their similarity to these representations. FIGURE 4) illustrates an example of a support set and a query set used during Few-Shot Learning. The support set consists of labeled images from three classes, while the query set contains unseen images that the model must classify by comparing them with the class prototypes computed from the support set.

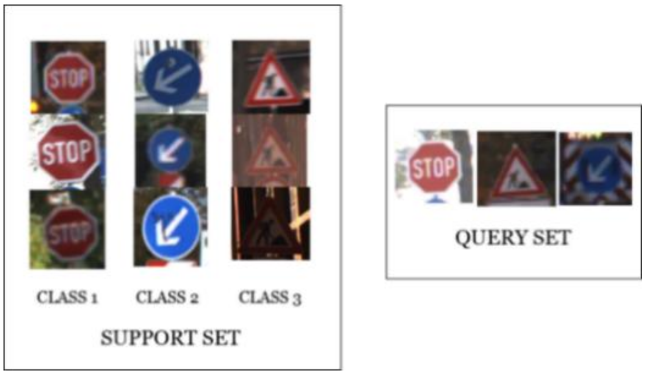

FIGURE 4. Example of a 3-way, 3-shot Few-Shot Learning episode

For this study, different model architectures were trained using varying episodic configurations, as shown in TABLE II. The models were trained under different N-way, K-shot setups to ensure a balanced evaluation based on their architectural design and computational efficiency. Higher N-way settings were used for certain models to encourage stronger generalization, while all models were trained with a fixed number of 10-shot support samples and 10 query samples per class.

TABLE II
MODEL CONFIGURATIONS FOR CLASSIFICATION TASKS

Model	Task Type	Shots per Class	Query per Class
Resnet 18	30-way	10-shot	10
EfficentNetV2	30-way	10-shot	10
CNN	30-way	10-shot	10
MobileNetV3	30-way	10-shot	10
ConViT	10-way	10-shot	10

2) Episodic Sampling(Task Creation in Each Episode): During training, each episode is structured to simulate Few-Shot Learning scenarios, ensuring the model learns to generalize from a limited number of labeled samples. Each episode consists of the following steps:

i. **Selection of N classes:** A subset of classes is randomly sampled from the dataset to form the classification task for the episode.
ii. **Selection of K support samples per class:** A fixed number of labeled examples are chosen for each class to compute prototypes.
iii. **Selection of Q query samples per class:** Additional unlabeled examples are selected for evaluation to assess the model's ability to classify new instances.
iv. **Prototype computation:** The model generates class prototypes based on the support set, representing the average feature embedding for each class.
v. **Query classification:** The query samples are classified by measuring their distance to the computed prototypes using a similarity metric such as Euclidean distance.

This episodic framework ensures that the model is exposed to a diverse set of classification tasks throughout training, improving its ability to recognize novel classes under Few-Shot Learning constraints.

3) Loss Function(Negative Log-Likelihood Loss): To train the model effectively, the negative log-likelihood (NLL) [41] loss is employed, ensuring that the predicted probability for the correct class is maximized. The loss function is defined as:

$$\mathcal{L} = -\log P\left(y = c \mid x_q\right) \quad (4)$$

where $\mathbf{P}\left(y = c \mid x_q\right)$ represents the probability assigned to the correct class c for a given query image x_q. As described in Equation (4) , minimizing this loss function encourages the model to refine its feature embeddings by positioning them closer to their corresponding class prototypes while increasing separation from non-matching class prototypes. This optimization process enhances the model's discriminative ability, improving classification performance in Few-Shot Learning settings.

Unlike conventional supervised learning, where a model is trained on a fixed set of classes with abundant labeled examples, episodic training continuously exposes the model to new classification tasks throughout the training process. This is particularly advantageous in Few-Shot Learning, where the goal is to recognize novel categories using only a few labeled instances.

The benefits of episodic training include:

- **Simulating real-world learning scenarios:** The model is trained to differentiate between a small number of classes per episode, mirroring real-world applications where new categories emerge with limited labeled data.

- **Enhancing generalization to unseen traffic signs:** Since the model repeatedly learns to adapt to new class distributions, it becomes more effective at recognizing novel traffic signs.
- **Preventing overfitting to specific class features:** By dynamically varying the support and query sets in each episode, episodic training reduces the risk of overfitting, promoting more transferable feature representations.

By leveraging episodic training, the model develops a well-structured feature space that facilitates Few-Shot classification with minimal supervision. This approach is particularly well-suited for real-world traffic sign recognition, where new or rare sign categories must be accurately classified without requiring large-scale labeled datasets.

E. TESTING STRATEGY

The testing phase was designed to evaluate how well the trained models generalized to unseen traffic signs. The predefined test set from the GTSRB dataset was used for a consistent and fair evaluation.

During testing, prototypes were computed for each class using all available images belonging to that class. Each test image was then processed by the model to extract its feature representation. To classify a test image, the Euclidean distance between its feature representation and all class prototypes was computed, and the image was assigned to the class with the shortest distance. The flow of the overall classification process is depicted in FIGURE 5), where the support and query images are passed through a feature extractor, and the embeddings are compared to determine the predicted class.

F. MODEL ARCHITECTURES

The choice of model architecture plays a crucial role in Few-Shot Learning, particularly in scenarios like traffic sign recognition, where distinguishing between visually similar classes requires rich feature representations. To explore different architectural paradigms, we employed a diverse set of models, ranging from traditional convolutional networks to modern vision transformers. Each model serves as the feature extractor for the Prototypical Network, transforming input images into meaningful representations before classification.

To ensure a fair comparison, all models were trained under the episodic training framework, using the configurations described earlier. The selected architectures vary in complexity, computational efficiency, and the ability to capture both local and global features. The details of each model, including architecture type, feature dimension and parameter count, are summarized in TABLE III.

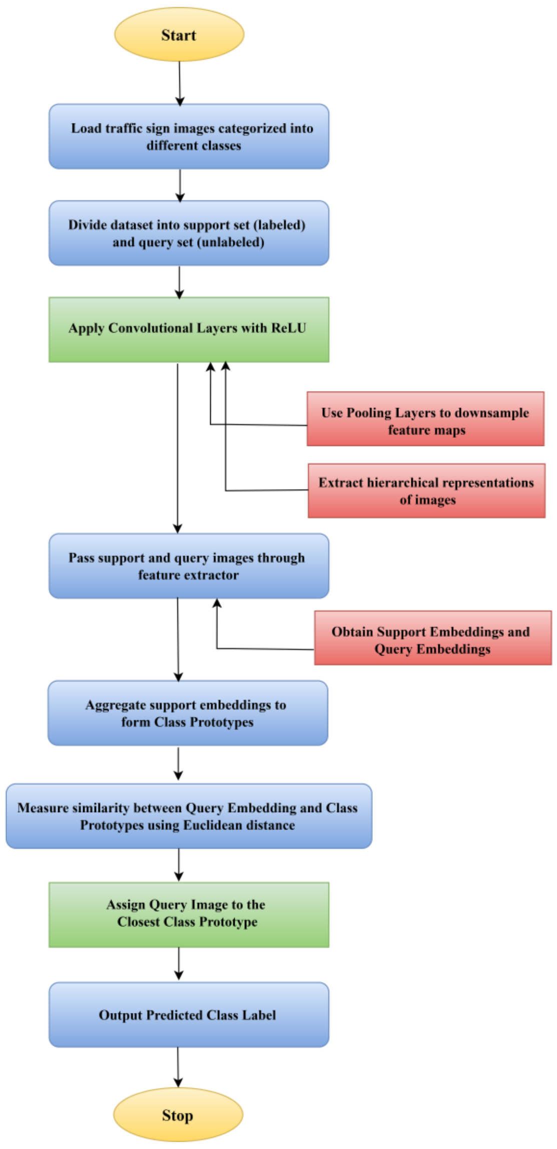
Start
Load traffic sign images categorized into different classes
Divide dataset into support set (labeled) and query set (unlabeled)
Apply Convolutional Layers with ReLU
Use Pooling Layers to downsample feature maps
Extract hierarchical representations of images
Pass support and query images through feature extractor
Obtain Support Embeddings and Query Embeddings
Aggregate support embeddings to form Class Prototypes
Measure similarity between Query Embedding and Class Prototypes using Euclidean distance
Assign Query Image to the Closest Class Prototype
Output Predicted Class Label
Stop

FIGURE 5. Workflow for traffic sign classification using Prototypical Networks

TABLE III
COMPARISON OF MODEL ARCHITECTURES FOR FEW-SHOT LEARNING

Model	Type	Feature Dimension	Img Size	Params
CNN	CNN	64	84	113k
Resnet 18	Deep CNN	512	84	11.18M
MobilenetV3	Lightweight CNN	960	84	2.97M
EfficientVit	Hybrid CNN-TF	64	84	15.00M
ConVit-Tiny	Vision TF	768	224	5.52M

The following subsections outline the core model architectures used in this study.

1) CNN: The CNN-based is a simple and lightweight model used to extract features from traffic sign images. It is made up of four main blocks, and each block includes a convolutional layer, batch normalization, a ReLU activation function, and a max pooling layer. This kind of structure helps the model learn important patterns from the image while reducing its size step by step. All four convolutional layers use 64 filters and a 3×3 kernel with padding, which helps keep the size of the feature maps consistent before pooling. Batch normalization helps the model train more smoothly, while ReLU adds non-linearity so the model can learn more complex patterns. Max pooling with a 2×2 window is used to reduce the size of the image at each stage, which also helps the model focus on the most important features.

The last convolutional layer changes the number of output channels to match the feature dimension we want, which is 64 in this case. If the input image is 84×84 pixels, then after all four pooling layers, the output feature map becomes 5×5 in size. The final output from the CNN is a feature map of size 64×5×5, which is then flattened into a one-dimensional feature vector of length 1 to decide which class the image belongs to. The CNN-based ProtoNet is easy to implement and fast to run, which makes it a good choice for systems that have limited processing power, like IoT devices.

To better understand how the model interprets images, the following Figure (6) shows the activation maps at various stages of the convolutional process. These feature maps represent the learned activations after passing through the CNN layers. They illustrate how the network focuses on different parts of the image to capture the relevant details, such as numbers and symbols on traffic signs.

This model was chosen because it provides a clear and efficient way to test the core idea of Few-Shot Learning without relying on a complex or heavy architecture. Its simplicity makes it ideal for understanding the effect of the learning framework itself, and it acts as a reliable baseline for comparing more advanced models. While the CNN-based is efficient and works well in controlled environments, it may not capture fine-grained differences between similar classes as effectively as deeper architectures. Its performance can also be limited when applied to large-scale or high-variance datasets, but it remains a strong candidate for quick inference and low-power applications.

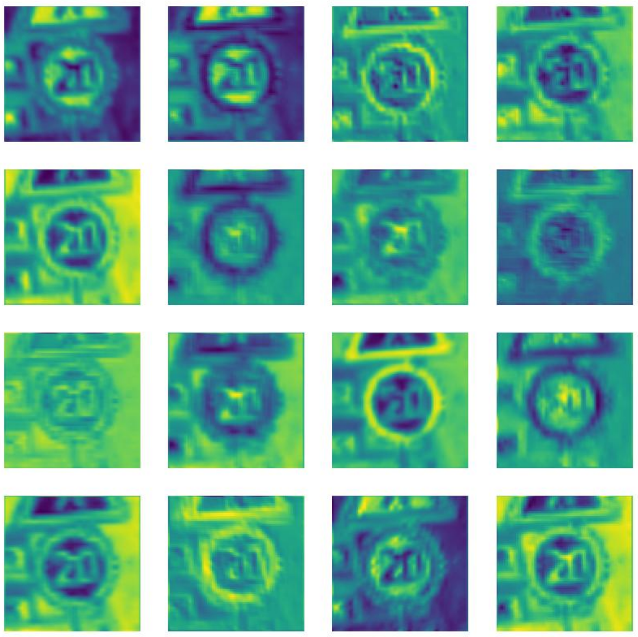

FIGURE 7. **CNN Feature Maps (Activation) at various layers.**

2) ResNet18 : The ResNet18 [31] Protonet based uses a pre-trained ResNet18 model to extract features from traffic sign images. As shown in

FIGURE 8), ResNet18 is a deeper convolutional neural network that includes residual connections, which help in training by allowing the model to pass information across layers more easily. These connections reduce the risk of vanishing gradients and allow the network to learn more complex patterns from the data, especially in deeper layers. In this setup, the final fully connected classification layer of ResNet18 is removed, and only the convolutional layers are used for feature extraction. The model is initialized with weights pre-trained on the ImageNet dataset, which allows it to start with a strong ability to recognize general visual features. As the image passes through the network, ResNet18 extracts increasingly abstract features, moving from edges and textures in the early layers to more detailed patterns in the deeper layers. By the end of the model, each input image is converted into a feature map with a spatial size of 1×1 and a depth of 512. This output is then flattened into a one-dimensional feature vector of length 512, which is used in the prototypical network for distance-based classification.

This feature vector acts as a compact representation of the image and is used to compare with class prototypes during Few-Shot Learning. The advantage of using a pre-trained ResNet18 is that it can capture more meaningful and fine-grained details than a simple CNN, especially when classes

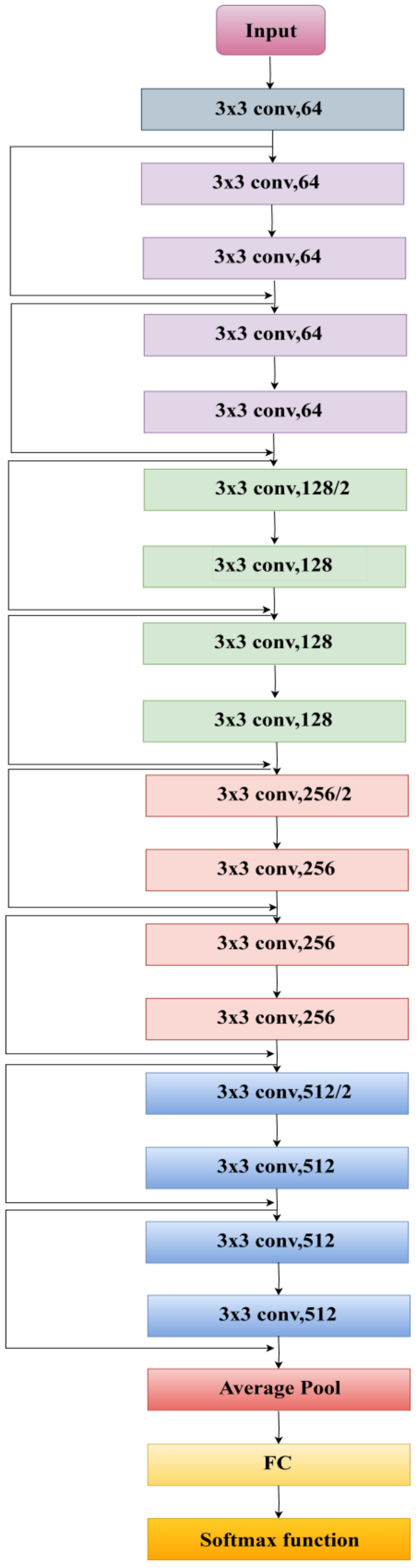

FIGURE 8. **ResNet18 Architecture for Traffic Sign Classification**

have small visual differences. Since the model is already trained on a large and diverse dataset, it requires less data and time to adapt to new tasks, which fits well with the goals of Few-Shot Learning.

ResNet18 was chosen because it strikes a good balance between depth, performance, and computational efficiency. It is deep enough to extract powerful features but still light enough to run on most systems without major hardware demands. Its pre-trained weights help improve generalization, especially when labeled data is limited. However, compared to simpler models, it has a larger number of parameters and requires more memory and computation. While not the most lightweight option, it remains a reliable and widely used architecture that performs well across many visual recognition tasks.

3) MobileNetV3 *:* The MobileNetV3-based ProtoNet uses the MobileNetV3 architecture [31] as the feature extractor for the Few-Shot Learning task. MobileNetV3 is a lightweight convolutional neural network that is specifically designed to run efficiently on mobile and edge devices. As shown in FIGURE 9) , it uses a combination of depth wise separable convolutions and attention mechanisms to reduce the number of parameters while still learning strong and meaningful features from images.

In this model, the final classification layers of MobileNetV3 are removed, and the remaining network is used to extract feature embeddings from the input images. As images pass through the network, the model applies several efficient convolutional blocks that focus on preserving important details while keeping the computation low. At the end of the network, the output is a compact feature vector that captures the main visual patterns in the image. This feature vector is then flattened and used to compare with class prototypes in the Prototypical Network setup.

The output feature vector acts as a unique representation of each input image and helps in identifying the closest matching class during classification. MobileNetV3 is particularly effective in scenarios where model size and speed matter, making it suitable for real-time systems and low-power devices like IoT platforms. Its ability to learn useful features with fewer resources makes it a strong candidate for Few-Shot Learning, especially when large models are not practical.

This model was chosen because of its excellent balance between performance and efficiency. It is much smaller and faster than deeper networks like ResNet, yet still powerful enough to capture meaningful patterns needed for classification. MobileNetV3 is ideal for deploying Few-

Shot Learning models on embedded systems or applications where speed and low energy usage are important. However, since it is a lightweight model, it may not perform as well as deeper architectures on very complex tasks. In situations where higher accuracy is required and resources are available, larger models may still be preferred. Nonetheless, MobileNetV3 remains a practical and effective choice for real-time Few-Shot Learning tasks.

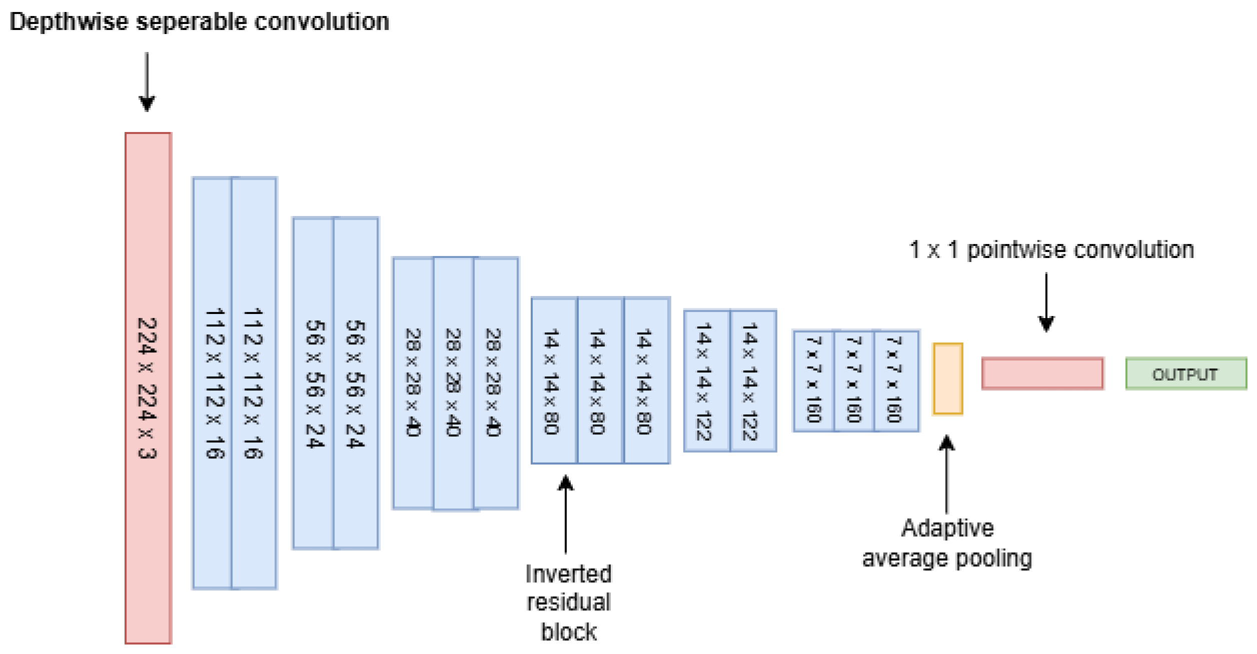

FIGURE 9. **MobileNetV3 Architecture for Traffic Sign Classification**

*4) **EfficientViT :*** The EfficientViT based uses a hybrid architecture that combines the strengths of both convolutional neural networks and vision transformers to extract features from traffic sign images. EfficientViT is designed to be both fast and accurate, making it suitable for tasks where real-time performance is important without sacrificing the quality of feature representation. It applies standard convolutional layers to capture local features and then uses lightweight attention mechanisms to understand relationships across the whole image.

The architecture begins with a standard convolutional layer followed by a series of optimized building blocks, including Fused-MBConv for early layers and MBConv with squeeze-and-excitation (SE) modules for deeper layers. EfficientNetV2 is designed for high performance with reduced computational complexity, making it ideal for real-time applications such as traffic sign recognition on edge devices.

In this setup, the classification head of the original EfficientViT model is removed, and the remaining layers are used as the feature extractor. As an image flows through the network, convolutional layers first focus on capturing textures, edges, and shapes, while the transformer-based blocks help model more global information. This combination allows the network to learn detailed and context-aware features. The final output is a compact embedding that summarizes the image and is used in the Prototypical Network to calculate distances between the image and each class prototype.

The feature vector produced by the EfficientViT acts as a rich representation of the input image and helps the model make accurate predictions, even with only a few training examples. The attention mechanism included in the model improves its ability to differentiate between visually similar traffic signs by focusing on important regions of the image, rather than treating all pixels equally. EfficientViT was chosen because it provides a balance between the strong local pattern recognition of CNNs and the global understanding offered by transformers. It is more lightweight compared to traditional vision transformers but still offers some of their benefits in terms of capturing complex spatial relationships. This makes it a good choice for Few-Shot Learning, where detailed and generalizable features are important. However, since it uses attention blocks, it can still be heavier than standard CNNs in terms of computation. While it may not be as fast as very lightweight models like MobileNet, EfficientViT offers a middle ground that works well in both accuracy and speed, especially for real-world tasks like traffic sign recognition.

5) ConViT-Tiny: The ConViT-based uses the ConViT-Tiny model as its feature extractor for the Few-Shot Learning setup. ConViT, which stands for Convolutional Vision Transformer, is a type of vision transformer that blends the structure of transformers with the locality and inductive biases of convolutional layers. Unlike pure transformer models, ConViT introduces soft convolutional layers at the beginning, which help the model better understand spatial information in images during the early stages of processing.

The network begins with patch embeddings and processes both local and global features through gated positional self-attention (GPSA) and standard self-attention (SA) layers. The right side highlights the GPSA mechanism, which adaptively combines learned positional relationships with content-based attention. This design allows ConViT to benefit from the locality of convolutions while leveraging the flexibility of transformers, making it suitable for visual recognition tasks like traffic sign classification.

In this model, the final classification layer of ConViT is removed, and the rest of the architecture is used to generate feature embeddings. As images pass through the encoder, the model first applies convolutional operations to capture local features and then switches to transformer blocks that focus on capturing long-range dependencies and global relationships in the image. The final output is a feature vector of length 768, which is used by the Prototypical Network to compare against class prototypes and make predictions. The use of attention mechanisms in ConViT allows it to focus on different parts of an image and understand how various features relate to each other. This can be especially useful in traffic sign recognition, where small differences in shape or color can separate one class from another. The combination of convolution and self-attention gives ConViT the ability to extract both fine-grained local details and broader contextual patterns.

ConViT was chosen because it brings together the advantages of CNNs and transformers in a single architecture. Its early convolution layers make it more efficient and easier to train compared to full transformers, while its attention-based structure gives it the flexibility to model complex visual patterns. However, since ConViT still relies on transformer blocks, it tends to be more computationally intensive than traditional CNNs or lightweight models like MobileNet. This makes it more suitable for tasks where accuracy is a higher priority than inference speed. Overall, ConViT provides a powerful and modern backbone for Few-Shot Learning tasks, especially when the goal is to handle fine distinctions between classes with limited data.

By employing a diverse range of architectures, this study examines how different feature extractors impact the performance of Prototypical Networks in Few-Shot Learning. The results from these models will provide insights into the trade-offs between efficiency, model complexity, and classification accuracy in real-world traffic sign recognition.

IV. EXPERIMENTATION

This section details the implementation and training setup used to evaluate the effectiveness of Few-Shot Learning for traffic sign recognition. It covers the framework and hardware specifications, the training procedure, including dataset preparation, episodic sampling, and hyperparameter configurations, as well as the evaluation strategy, outlining the key metrics used to assess model performance. The results are then presented through quantitative analysis and visualizations to highlight the effectiveness of the approach.

A. IMPLEMENTATION DETAILS

The Few-Shot Learning experiments were implemented using PyTorch, an open-source deep learning framework that provides flexibility in neural network development. All models were trained and evaluated on Google Colab, utilizing an NVIDIA T4 GPU, which offers a balance of computational efficiency and accessibility.

1) Framework and Libraries: The implementation relied on various libraries for model training, data preprocessing, and evaluation:

- PyTorch - The primary framework for building and training neural networks.
- TorchVision - Used for dataset handling and image transformations.
- timm (Torch Image Models) - Provided access to transformer-based architectures such as ConViT
- NumPy & Pandas – Used for dataset manipulation and statistical analysis.
- Matplotlib & Seaborn – Utilized for result visualization, including loss curves and performance trends.

2) Hardware Setup: All experiments were conducted on Google Collab, leveraging the NVIDIA T4 GPU, which includes 16 GB of VRAM. This hardware was sufficient for training both CNN-based and transformer-based models in the Few-Shot Learning setting. The training time varied depending on model complexity, with lightweight architectures like MobileNetV3 completing training faster than Vision Transformer-based models such as ConViT-Tiny.

3) Dataset Loading Process: The dataset was loaded using PyTorch’s Data Loader, ensuring efficient batch processing. Images were pre-processed before training, with the following steps applied:

- **Resizing** – Adjusting images to the required input dimensions (84×84 for CNN-based models, 224×224 for transformer-based models).

- **Conversion to Tensors** – Transforming images into PyTorch tensors for compatibility.
- **Normalization** – Standardizing pixel values using mean and standard deviation normalization.
- **Batch Sampling** – Implementing episodic sampling, where each training episode dynamically selects support and query sets for Few-Shot Learning.

By structuring the dataset in this manner, the models were trained under episodic learning, allowing them to develop generalizable representations from limited labeled samples.

B. TRAINING PROCEDURE

The training process followed an episodic learning framework, ensuring that models learned to generalize across different Few-Shot Learning tasks. The hyperparameters were carefully selected after extensive experimentation with different configurations, and the values that yielded the best accuracy were chosen. The final hyperparameters used for training are summarized in TABLE IV. This configuration was applied consistently across all models to ensure a fair comparison of performance.

TABLE IV
FINAL HYPERPARAMETERS USED FOR TRAINING

Hyperparameters	Value
Learning Rate	**0.0005**
Batch Size	**1**
Epochs	**5**
Optimizer	**Adam**
Number of Episodes	**1000**

V. RESULTS AND DISCUSSION

A. EVALUATION STRATEGY

To assess the performance of different models in the Few-Shot Learning framework, multiple evaluation metrics were employed. These metrics provide insights into classification accuracy and the ability of the models to correctly identify traffic signs under varying conditions.

1) Accuracy: Accuracy [38] is a fundamental metric in classification tasks, measuring the proportion of correctly classified instances over the total number of instances. It is computed as shown in (5).

$$Accuracy = \frac{Number\ of\ Correct\ Prediction}{Total\ number\ of\ Samples} \quad (5)$$

In Few-Shot Learning, accuracy evaluates how well the model generalizes to new, unseen classes with limited labeled examples.

2) Precision: Precision measures [35][36] the proportion of correctly predicted positive instances out of all predicted positive instances. It is defined in (6) :

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)} \quad (6)$$

Where,

- True Positives (TP): The number of correctly classified instances belonging to the positive class
- False Positives(FP): The number of instances that were incorrectly classified as positive but actually belong to a different class.

A higher precision value indicates that the model makes fewer false positive predictions, meaning it is more confident in the instances it classifies as positive.

3) Recall: Recall (or Sensitivity) [35][36] measures the proportion of correctly predicted positive instances out of all actual positive instances. It is computed using (7):

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)} \quad (7)$$

Where,

- True Positives (TP): The number of correctly classified instances belonging to the positive class
- False Negatives (FN): The number of instances that actually belong to the positive class but were incorrectly classified as a different class.

A higher recall value means that the model successfully identifies most of the actual positive instances, reducing the number of false negatives.

4) F1-score: F1-Score is the harmonic mean of Precision and Recall, [37][38] providing a balanced measure when dealing with imbalanced datasets. It is given by (8)*:*

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (8)$$

These metrics ensure a more detailed evaluation, particularly in cases where class imbalance or misclassifications could affect the overall performance. The results based on these metrics are presented in the following section.

B. PERFORMANCE EVALUATION

After establishing the evaluation strategy, the models were assessed based on their training and testing performance. This section presents a detailed analysis of their performance using accuracy, precision, recall, and F1-score. The first part of this evaluation focuses on training performance, highlighting how well each model learned from the dataset.

1) Training Performance Analysis: The training performance of different models was evaluated using training accuracy, precision, recall, and F1-score, as presented in TABLE V. These metrics indicate how well each model learned from the dataset before being tested on unseen data. Training accuracy represents the percentage of correctly classified samples during training. A higher value means the model has effectively captured patterns in the dataset. All models achieved over 99% training accuracy. Among them, MobileNetV3 performed the best, reaching **99.98%,** showing that it learned almost perfectly. ResNet18 **(99.94%)** and EfficientB2 **(99.93%)** followed closely, proving their strong feature extraction abilities. ConViT **(99.87%)** also performed well, while CNN had the lowest accuracy at **99.24%,** reflecting its simpler architecture.

TABLE V
TRAINING PERFORMANCE OF DIFFERENT MODELS

Model	Training Accuracy	Precision	Recall	F1-score
CNN	99.24	0.9925	0.9924	0.9924
ConViT	99.87	0.9987	0.9987	0.9987
EfficientB2	99.93	0.9993	0.9993	0.9993
MobileNetV3	99.98	0.9998	0.9998	0.9998
ResNet18	99.94	0.9994	0.9994	0.9994

While high training accuracy is a good sign, it does not always mean the model will perform well on new, unseen data. Sometimes, a model may memorize the training data too well, making it less effective when tested in real-world situations. This issue, known as overfitting**,** highlights the importance of looking at test accuracy to understand how well the model generalizes.

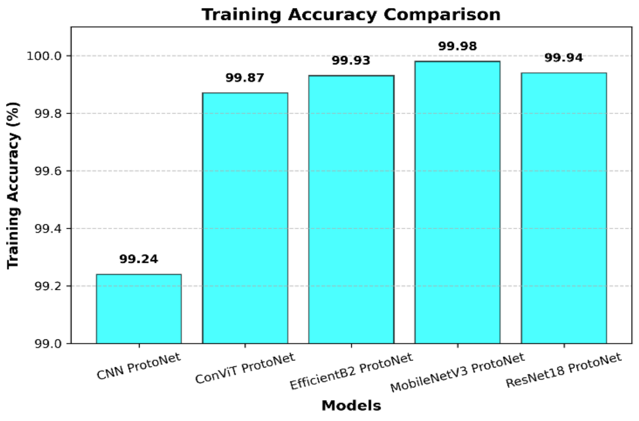

FIGURE 10. Training Accuracy Comparison of different models

To better visualize the training accuracy of each model, FIGURE 10) presents a bar chart comparing their performance. The graph clearly shows that deeper architectures, such as MobileNetV3 and ResNet18, achieved the highest training accuracy, reinforcing their strong learning capabilities.

Apart from accuracy, other metrics like precision, recall, and F1-score (from Table IV) provide a more detailed look at training performance. MobileNetV3 ProtoNet achieved the highest scores across all three metrics (**0.9998**), indicating that it classified training samples with almost perfect accuracy. ResNet18 and EfficientB2 also scored high **(~0.9994 and 0.9993**), confirming their ability to capture complex patterns. On the other hand, CNN had the lowest values (~**0.9924–0.9925**), showing its limitations in comparison to the other models.

Although these results indicate that all models learned effectively, the near-perfect scores of some models suggest a possible risk of overfitting. To determine if these models perform well in real-world applications, it is essential to examine their test accuracy and generalization ability.

2) Test Performance Analysis: While training accuracy shows how well a model learns from the dataset, its true effectiveness is measured by how well it performs on unseen data. The test performance of different models was evaluated using test accuracy, precision, recall, and F1-score, as shown in TABLE VI. These metrics help determine how well the models generalize beyond the training data.

TABLE VI
TESTING ACCURACY AND EVALUATION METRICS

Model	Test Accuracy	Precision	Recall	F1-score
CNN	96.12	0.9477	0.9525	0.9484
ConViT	96.85	0.9546	0.9572	0.9550
EfficientB2	97.24	0.9568	0.9522	0.9521
MobileNetV3	96.80	0.9548	0.9569	0.9542
ResNet18	97.46	0.9625	0.9669	0.9639

Table VI reveals that ResNet18 achieved the highest test accuracy (**97.46%**), making it the best-performing model in terms of generalization. EfficientB2 followed closely with **97.24%**, showing that deeper architectures are effective in extracting meaningful features. ConViT and MobileNetV3 achieved similar test accuracies (**96.85%** and **96.80%**), while CNN had the lowest test accuracy at **96.12%**, suggesting that its simpler structure may not capture enough distinguishing patterns for Few-Shot Learning tasks.

To better understand the models' generalization ability, FIGURE 11) presents a bar chart comparing test accuracy across different models. It clearly shows that deeper CNN architectures, such as ResNet18 and EfficientB2, performed better than transformer-based models like ConViT and simpler architectures like CNN. Beyond accuracy, additional evaluation metrics—precision, recall, and F1-

score—provide a deeper analysis of how well the models handle different classification scenarios.

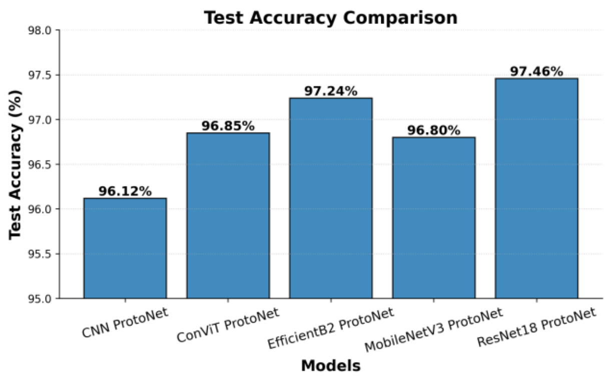

FIGURE 11. Testing Accuracy Comparison across models

Precision measures the proportion of correctly identified positive predictions out of all instances predicted as positive. In simpler terms, it reflects how many of the model's "positive" predictions were actually correct. As shown in FIGURE 12) , ResNet18 had the highest precision (0.9625), followed by EfficientB2 (0.9568). CNN had the lowest precision (0.9477), suggesting it made more false positive predictions compared to other models.

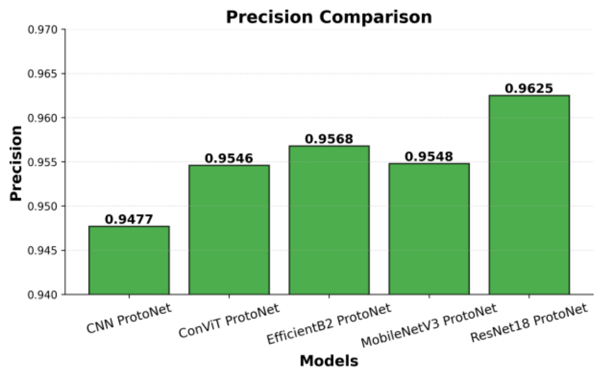

FIGURE 12. Precision Scores of Different Models

Recall, on the other hand, evaluates how effectively the model identifies all relevant positive instances. It tells us how many of the actual positive cases were correctly detected by the model. FIGURE 13) highlights that ResNet18 again achieved the highest recall (**0.9669**), meaning it successfully detected more positive cases than the other models. ConViT, MobileNetV3, and EfficientB2 performed closely, while CNN had the lowest recall (**0.9525**).

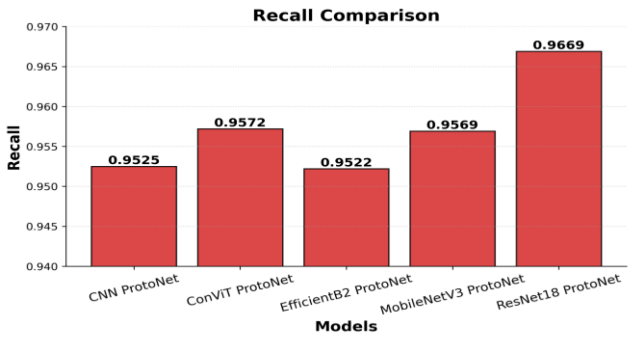

FIGURE 13. Recall Performance Across Models

Since precision and recall alone do not provide a complete picture, the F1-score, which balances both metrics, is also analyzed. As illustrated in FIGURE 14), ResNet18 had the highest F1-score (0.9639), confirming its strong generalization ability. Other models, including EfficientB2, ConViT, and MobileNetV3, performed competitively, while CNN had the lowest F1-score (0.9484).

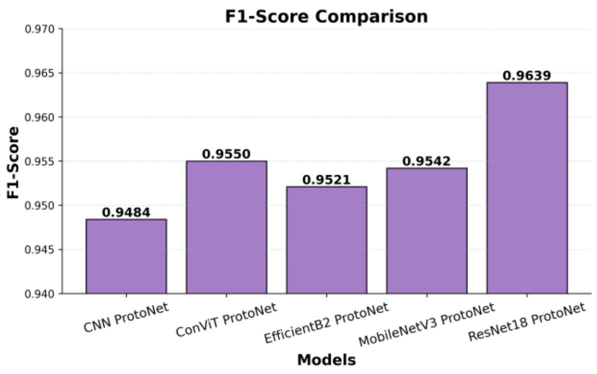

FIGURE 14. F1-Score Evaluation for Model Comparison

While MobileNetV3 achieved the highest training accuracy (99.98%), its test accuracy (96.80%) was lower than models like ResNet18 (97.46%) and EfficientB2 (97.24%). This drop suggests a tendency toward overfitting, where the model memorizes training patterns but struggles with unseen data. In contrast, ResNet18 and EfficientB2 maintained strong generalization, as their test accuracy remained high while avoiding excessive overfitting. CNN , with the lowest training accuracy (99.24%) and test accuracy (96.12%), indicates that its simpler architecture may limit feature extraction, leading to lower performance overall.

From these results, it is evident that deeper CNN-based models, such as ResNet18 and EfficientB2, consistently outperformed others in all test performance metrics. Despite achieving the highest training accuracy, MobileNetV3 did not generalize as well in testing, indicating a slight tendency toward overfitting. ConViT performed well, showing that transformer-based models can be effective in Few-Shot Learning, but they did not surpass deeper CNN architectures in this scenario.

Overall, ResNet18 emerged as the most effective model, achieving the best balance between accuracy, precision, recall, and F1-score. EfficientB2 and ConViT also delivered strong results, while CNN struggled to match the performance of deeper networks. These findings suggest that deep CNN architectures remain the most reliable choice for Few-Shot Learning in real-world traffic sign classification.

3) Prototype Distance Heatmap: To better understand how our model makes predictions, we created a heatmap that shows the distances between query images and class prototypes. This helps us see how well the model groups similar traffic signs together. In FIGURE 15) , the rows represent query images and the columns represent the 43 traffic sign classes (prototypes). The colour in each cell shows how far the query image is from each class prototype—darker colours mean the image is closer (more similar), and lighter colours mean it's farther away.

This heatmap is based on the ResNet18, which was our best-performing model. As you can see, there are strong vertical lines of dark colours in many places, which means the model is correctly matching query images to the right class. There are also a few lighter areas where the model might be getting confused between similar-looking signs. This kind of visual helps us understand where the model is doing well and where it might need improvement.

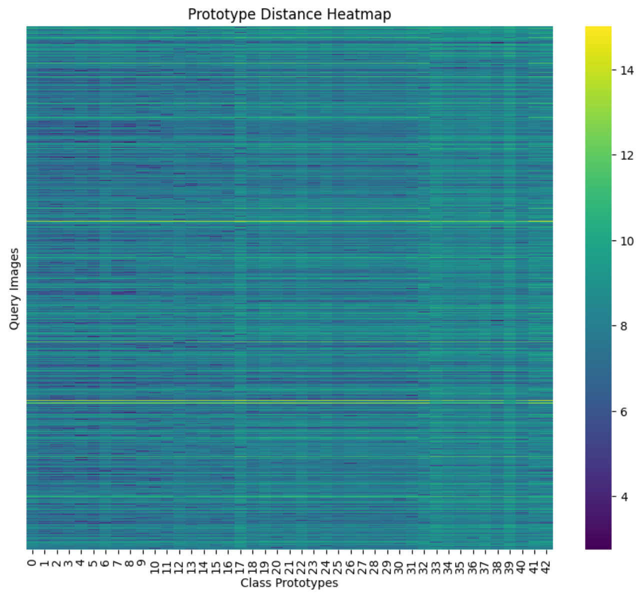

FIGURE 15. **Heatmap showing distances between query images and class prototypes.**

We saw similar patterns in other models like MobileNetV3 and ConViT-Tiny.

C. FSL IMPORTANCE

To further validate the flexibility of our approach, we introduced a new class (CATTLE) from the Indian Traffic Signs Prediction (85 Classes) dataset, which was not originally part of our training data. This dataset consists of 85 different types of Indian traffic signs, with a total of 7,210 labeled images, curated from various online sources. Unlike standard datasets such as GTSRB, this dataset includes region-specific traffic signs relevant to India, such as warnings for cattle crossings, which are common in rural and urban Indian roadways.

Instead of retraining the entire model, we leveraged Few-Shot Learning (FSL) with Prototypical Networks to integrate the new class dynamically. By computing a prototype for CATTLE using a small support set and adding it to the existing class prototypes, the model was able to classify CATTLE images without any modifications to its learned parameters. The evaluation on CATTLE test images is show in the TABLE VII.

TABLE VII
TESTING ACCURACY AND EVALUATION METRICS

Model	Accuracy
CNN	100
ConViT	61
EfficientB2	82
MobileNetV3	100
ResNet18	100

This capability offers significant advantages over traditional deep learning models in real-world applications such as autonomous navigation. In conventional models, introducing a new traffic sign would require collecting large datasets, retraining the model, and redeploying the system, which is time-consuming and computationally expensive. In contrast, our method enables on-the-fly adaptation, allowing autonomous vehicles to quickly recognize newly introduced or region-specific traffic signs, such as the CATTLE warning sign in India. This is particularly valuable in environments where traffic conditions evolve, ensuring that the navigation system remains continuously updated without major infrastructure changes. The seamless expansion of our model highlights the practical advantages of FSL-based classification systems in dynamic and safety-critical scenarios.

VI. CONCLUSION AND FUTURE WORK

This study explored how Few-Shot Learning (FSL) with Prototypical Networks can be used for recognizing traffic signs. By using the German Traffic Sign Recognition Benchmark (GTSRB) dataset, we tested how well the model could classify traffic signs with just a few labeled examples. The results showed that models like ResNet18 and EfficientViT performed the best, offering strong feature extraction and the ability to generalize to different types of traffic signs. Even though ConViT performed fairly well, it did not perform as well as other models such as ResNet18 and EfficientViT in this case. These findings highlight how FSL can be a useful tool for real-world applications, especially in autonomous vehicles. The ability to quickly recognize new traffic signs without the need to retrain the model every time a new sign appears is a significant advantage. This means that autonomous vehicles can safely drive in areas with new signs without requiring constant updates, making FSL a practical and scalable solution.

However, while FSL shows great potential, there are some limitations that need to be addressed. One of the main challenges is the use of prototypes for classification. Prototypes are essentially averaged feature vectors from a few labeled images. While they work well for most cases, they might not be effective when there are large differences between the signs within the same category. For example, traffic signs from different countries or regions may have slight variations in their design, which can confuse the model. Additionally, the performance of the model can drop in cases of poor weather, low lighting, or when traffic signs are partially obscured. This is because we did not use data augmentation in our approach. Data augmentation, such as rotating or changing the brightness of images, would help the model learn to deal with these real-world challenges. Finally, the computational cost of episodic training—where the model is trained by learning from a small number of examples repeatedly—can be a barrier, especially for real-time applications where speed is crucial.

There are several directions for improving the current approach and addressing its limitations. One area that needs improvement is the way prototypes are created. Currently, prototypes are made by averaging the feature vectors of a few labeled images. While this works in many cases, it might not fully capture the differences within a class. More advanced techniques for learning prototypes could be explored, such as using methods that focus on better capturing the diversity of each class. This could improve the model's performance when traffic signs in the same category have subtle differences in their design.

Another key area for improvement is data augmentation. In real-world applications, traffic signs are often seen under varying conditions, such as different lighting, weather, and angles. By using data augmentation techniques like rotation, zooming, or color adjustments, the model could become more robust and adaptable to these changes. For example, if the model is trained with images of traffic signs under rainy or foggy conditions, it would perform better when those conditions are encountered in real life. This would also help the model recognize partially obscured signs, which is a common challenge in real-world driving scenarios.
There is also room for improvement by exploring hybrid models that combine CNNs and transformers. CNNs are very good at detecting local features, such as shapes and textures, but they may struggle with understanding the broader context of an image. On the other hand, transformers excel at capturing long-range relationships between pixels. By combining CNNs with transformers, we can create a more powerful model that benefits from both local and global feature extraction. This hybrid approach could improve accuracy, especially when distinguishing between visually similar traffic signs.

In addition to hybrid models, another area to explore is semi-supervised learning or self-supervised learning. Traffic sign datasets often suffer from a lack of labeled data, which is expensive and time-consuming to create. Semi-supervised learning allows the model to learn from both labeled and unlabeled data, while self-supervised learning helps the model generate its own labels from the data, reducing the need for manually labeled examples. These approaches could be especially useful when there is limited labeled data available for rare or new types of traffic signs.

Another promising direction is to optimize the models for on-device inference. Real-time applications, such as autonomous vehicles, require the ability to process images quickly and efficiently on embedded systems, which have limited computational resources. Future work could focus on making the models more lightweight and efficient, ensuring that they can run on devices with limited processing power. Techniques like model pruning, which removes unnecessary parts of the model, and quantization, which reduces the precision of the model's calculations, could help achieve faster inference times and lower memory usage.

Finally, there is potential to integrate edge computing into the training and inference processes. Edge computing allows data processing to be done closer to where the data is generated, such as in the vehicle itself, rather than relying on cloud-based servers. This can reduce latency and improve the responsiveness of the system, which is essential for autonomous driving. Future work could investigate how FSL models can be optimized for deployment in edge computing environments, ensuring that traffic sign recognition can be done efficiently in real-time.

While this study has demonstrated the effectiveness of Few-Shot Learning with Prototypical Networks for traffic sign recognition, there is still much to explore. By addressing the challenges of prototype learning, data augmentation, computational efficiency, and model robustness, future research can make these systems more adaptable, accurate, and ready for real-world deployment. With further improvements, FSL-based traffic sign recognition models could play a significant role in advancing autonomous driving technology, making roads safer and more efficient for everyone.

REFERENCES

[1] J. Janai, F. Güney, A. Behl, and A. Geiger, “Computer vision for autonomous vehicles: Problems, datasets and state of the art,” Found. Trends Comput. Graph. Vis., vol. 12, no. 1–3, pp. 1–308, 2020.
[2] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[3] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition,” Neural Netw., vol. 32, pp. 323–332, Aug. 2012.
[4] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, “Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Dallas, TX, USA, Aug. 2013, pp. 1–8.
[5] H. Luo, F. Wu, Y. Hu, and X. Zhang, “Traffic sign recognition using a multi-task convolutional neural network,” IEEE Trans. Intell. Transp. Syst., vol. 19, no. 4, pp. 1100–1111, Apr. 2018.
[6] P. Zhai, Y. Hu, Y. Wu, and C. Zhang, “Data-efficient deep learning for traffic sign recognition using few-shot learning techniques,” IEEE Access, vol. 10, pp. 88140–88150, 2022.
[7] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, “Cost-sensitive learning of deep feature representations from imbalanced data,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 8, pp. 3573–3587, Aug. 2018.
[8] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in Proc. Int. Conf. Mach. Learn. (ICML) Deep Learning Workshop, Lille, France, 2015.
[9] L. Wang, Y. Li, and S. Lazebnik, “Learning deep structure-preserving image-text embeddings,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Boston, MA, USA, 2015, pp. 5005–5013.
[10] J. M. Kaleybar et al., “Efficient Vision Transformer for Traffic Sign Detection in Autonomous Vehicles,” arXiv preprint arXiv:2311.01429, 2023.
[11] A. Farzipour et al., “Local Vision Transformer-Based Model for Traffic Sign Recognition in Autonomous Driving,” arXiv preprint arXiv:2311.06651, 2023.
[12] A. Kaur et al., “Improved YOLOv8 Model for Traffic Sign Detection and Classification in Intelligent Transportation Systems,” in Proc. IEEE, 2023.
[13] R. Gupta et al., “Traffic Sign Detection Model using YOLOv8 for Intelligent Transportation Systems,” in Proc. IEEE, 2023.
[14] S. Ashwath et al., “Innovative Traffic Sign Detection and Recognition System using YOLOv5,” in Proc. IEEE, 2023.
[15] S. R et al., “Traffic Sign Detection System for Indian Road Conditions using a Multi-Focal YOLOv8n Model,” in Proc. IEEE, 2023.
[16] S. Jency et al., “Traffic Sign Recognition System for Autonomous Vehicles using Deep Learning,” in Proc. IEEE, 2023.
[17] N. Urs H. D et al., “Deep CNN-Based Approach for Real-Time Traffic Sign Detection and Recognition,” in Proc. IEEE, 2023.
[18] R. Uikey et al., “Unified Framework for Indian Traffic Sign Detection and Classification,” in Proc. IEEE, 2023.
[19] S. Mingwin, Y. Shisu, Y. Wanwag, and S. Huing, “Revolutionizing Traffic Sign Recognition: Unveiling the Potential of Vision Transformers,” arXiv preprint arXiv:2404.19066, 2024.
[20] J. Zheng and Z. Jiang, “Evaluation of Vision Transformers for Traffic Sign Classification,” Journal of Advanced Transportation, vol. 2022, Article ID 3041117, 2022.
[21] S. K. Satti, G. N. V. Rajareddy, and A. H. Gandomi, “Potholes and Traffic Signs Detection by Classifier with Vision Transformer,” Scientific Reports, vol. 14, no. 1, pp. 1-12, 2024.
[22] H. Wang, “A Robust TrafficSignNet Algorithm for Enhanced Traffic Sign Recognition,” Neural Computing and Applications, vol. 36, pp. 12345-12356, 2024.
[23] H. Wang, “Traffic Sign Recognition with Vision Transformers,” in Proceedings of the 2022 International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1234-1243, 2022.
[24] O. N. Manzari, S. B. Shokouhi, and A. Farzipour, “Pyramid Transformer for Traffic Sign Detection,” arXiv preprint arXiv:2207.06067, 2022.
[25] Y. Gan, L. Chen, and Y. Zhang, “Cross-domain Few-shot In-context Learning for Enhancing Traffic Sign Recognition,” arXiv preprint arXiv:2407.05814, 2024.
[26] J. Li and Y. Wang, “Traffic Sign Recognition with an Optimized Deep Learning Model for Autonomous Vehicles,” IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 3, pp. 1234-1245, 2024.
[27] Z. Zhang, L. Li, and Q. Chen, “Integrating Transformer Models for Multi-Modal Traffic Sign Recognition,” IEEE Transactions on Neural Networks and Learning Systems, vol. 36, no. 5, pp. 1234-1245, 2025.
[28] E. Ellahyani, M. Essaaidi, and A. Rachidi, “Traffic Sign Detection and Recognition Using Features Combination and Random Forests,” International Journal of Advanced Computer Science and Applications, vol. 5, no. 4, pp. 27-32, 2014.
[29] Z. Zhang, L. Li, and Q. Chen, “A Cascaded R-CNN with Multiscale Attention and Imbalanced Samples for Traffic Sign Detection,” IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 4, pp. 2137-2146, 2021.
[30] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “The German Traffic Sign Recognition Benchmark: A Multi-Class Classification Competition,” in The 2011 International Joint Conference on Neural Networks. IEEE, 2011, pp. 1453–1460.
[31] Elsayed Abd Elaziz, Mohamed & Al-qaness, Mohammed & Dahou, Abdelghani & Alsamhi, Saeed & Abualigah, Laith & Ibrahim, Rehab & Ewees, Ahmed. (2023). Evolution toward intelligent communications: Impact of deep learning applications on the future of 6G technology. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 14. 10.1002/widm.1521.
[32] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," presented at the International Conference on Machine Learning (ICML), Virtual Event, 2021. [Online]. Available: https://arxiv.org/abs/2104.00298
[33] Ramzan, Farheen & Khan, Muhammad Usman & Rehmat, Asim & Iqbal, Sajid & Saba, Tanzila & Rehman, Amjad & Mehmood, Zahid. (2019). A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer’s Disease Stages Using Resting-State fMRI and Residual Neural Networks. Journal of Medical Systems. 44. 10.1007/s10916-019-1475-2.
[34] S. d'Ascoli et al., "ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases," presented at the International Conference on Machine Learning (ICML), Virtual Event, 2021. [Online]. Available: https://arxiv.org/abs/2103.10697
[35] D. M. Powers, “Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation,” *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, Dec. 2011. [Online]. Available: https://arxiv.org/abs/2010.16061
[36] J. R. Hersh, "The role of recall and precision in text retrieval applications," *IEEE Trans. Knowl. Data Eng.*, vol. 5, no. 3, pp. 420–426, June 1993. DOI: 10.1109/69.223799
[37] Y. Sasaki, “The truth of the F-measure,” National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan, Tech. Rep., 2007. [Online]. Available: https://www.cs.odu.edu/~mweigle/courses/cs895sum14/papers/sasaki07.pdf
[38] G. Chicco and S. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation,” *Sci. Rep.*, vol. 10, no. 1, Art. no. 1032, Jan. 2020. DOI: 10.1038/s41598-020-65222-3
[39] K. T. Chui, B. B. Gupta, L.-K. Lee, and M. Torres-Ruiz, "Analysis of N-Way K-Shot Malware Detection Using Few-Shot Learning," in *International Conference on Cyber Security, Privacy and Networking (ICSPN 2022)*, Cham, Switzerland, 2023, pp. 33–44. [Online]. Available: https://doi.org/10.1007/978-3-031-22018-0_4
[40] D. Yoo, H. Fan, V. N. Boddeti, and K. M. Kitani, "Efficient K-Shot Learning with Regularized Deep Networks," in *AAAI Conference on Artificial Intelligence (AAAI)*, New Orleans, LA, USA, 2018. [Online]. Available: https://ai.meta.com/research/publications/efficient-k-shot-learning-with-regularized-deep-networks/

[41] PyTorch Foundation, "NLLLoss," *PyTorch Documentation*, 2025. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.NLLLoss.html
[42] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical Networks for Few-shot Learning," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Long Beach, CA, USA, 2017, pp. 4077–4087. [Online]. Available: https://papers.nips.cc/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html
[43] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to Compare: Relation Network for Few-Shot Learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, 2018, pp. 1199–1208, doi: 10.1109/CVPR.2018.00131