Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 7, No. 4, July : 2023

TRAFFIC SIGN RECOGNITION USING EFFECIENTNETB0-CNN TRANSFER LEARNING MODEL

Gasikanti Mallikarjun Head of the Department, Electronics and Communication Engineering, Government Polytechnic Sangareddy, Department of Technical Education, Government of Telangana, India

Abstract

Traffic sign recognition plays a crucial role in ensuring road safety and efficient traffic management. In this study, we propose an innovative approach for traffic sign recognition utilizing the powerful EfficientNetB0-CNN transfer learning model. EfficientNetB0 is known for its exceptional performance in image classification tasks, and in this research, it is fine-tuned with a customized Convolutional Neural Network (CNN) for traffic sign recognition. The model is trained on a comprehensive dataset comprising various traffic signs, encompassing diverse lighting conditions, weather variations, and road environments. The transfer learning process allows the model to leverage pre-trained weights, enabling faster convergence and improved performance. Extensive experimentation and hyperparameter tuning are conducted to optimize the model's accuracy and efficiency. The results demonstrate that the proposed EfficientNetB0-CNN model achieves remarkable accuracy and robustness in recognizing traffic signs, showcasing its potential for real-world applications in traffic safety systems and autonomous vehicles. The developed approach opens new avenues for enhancing traffic sign recognition technologies, contributing to safer and more intelligent transportation systems.

Keywords: Traffic Sign recognition, transfer learning, EfficientNetB0, Convolutional Neural Network, traffic safety

1. Introduction

The ability to recognize traffic signs is an essential component in achieving both effective and comprehensive traffic management [1]. Understanding the meaning of traffic signs and being able to correctly recognize them has become an increasingly vital skill as the number of cars on the roads continues to rise. The traditional techniques of traffic sign identification mainly depended on handmade characteristics and rule-based algorithms, both of which were often restricted in their capacity to handle complicated situations and changes in the signs' appearance.

The introduction of deep learning models in recent years has brought about a revolution in the area of computer vision, which has had an effect on the identification of traffic signs [2]. Because deep learning models, and in particular convolutional neural networks (CNNs), have shown great performance in a variety of image identification tasks, they are a possible option for the detection and classification of traffic signs.

Because of the crucial part that traffic signs play in directing vehicles and guaranteeing their safety, it is necessary to have appropriate identification of the signs that are used on the roads. Important information, including speed limits, cautions, prohibited activities, and instructions, are communicated through traffic signs [3]. Accidents, breaches of the law, and increased congestion might result from a failure to appropriately understand these signals. We can improve the capability of intelligent transportation systems to detect, understand, and react appropriately to traffic signs if we use deep learning models for traffic sign identification. This will allow us to better manage traffic flow.

The significance of recognizing traffic signs using deep learning models [4] comes in the fact that it has the ability to automate the process, hence cutting down on the amount of room for human error and improving overall traffic management. Drivers may benefit from automated recognition systems since these systems can provide real-time information on the traffic signs that are located around them. This helps drivers improve their situational awareness and encourages safe driving behaviors. In addition, these technologies may be included into autonomous cars, giving the vehicles the ability to comprehend the surrounding environment of the road and react correctly to it.

Deep learning has a large range of potential applications in the field of traffic sign identification, and these applications are rather varied [5]. To begin, it may be used in advanced driving assistance systems (ADAS) to deliver warnings and alerts based on observed signs, therefore assisting drivers in avoiding possible risks and complying with traffic rules. Second, the identification of traffic signs may be included into the infrastructure of smart cities. This gives the authorities in charge of traffic management a more effective means of monitoring and controlling the flow of cars. Because of this, the timing of traffic signals may be adjusted, which may result in less congestion and more overall transportation efficiency. In addition, the identification of traffic signs may be of assistance in the development of autonomous cars, giving these vehicles the ability to navigate and make judgments depending on the signs that they have identified.
The identification of traffic signs via the use of deep learning models is an essential area of study and development. Deep learning is a powerful tool that can be used to make roads safer, make traffic management more efficient, and clear the path for the broad implementation of intelligent transportation systems. In order to realize a future in which cars are not only more intelligent but also safer when driving on the roads, it is essential to have the capacity to reliably identify and understand traffic signs [6].

2. Literature Review

Wasif Arman Haque et al [7] suggested a new Thin yet Deep convolutional neural network design that uses less energy and can recognize traffic signs. Each convolutional layer in the suggested design has less than 50 features. This means that the convolutional neural network can be learned quickly, even without a graphics processing unit. Two public traffic sign datasets, the German Traffic Sign Recognition Benchmark and the Belgian Traffic Sign Classification dataset, are used to measure how well the proposed system works. First, the authors use the big German Traffic Sign Recognition Benchmark dataset to train and test how well the proposed model works. Then, to test how well the models work, they retrain them using transfer learning on the more difficult Belgian Traffic Sign Classification dataset.
Jianming Zhang et al [8] suggested two new lightweight networks that can get more accurate recognition while keeping the same number of trainable factors in the models. information distillation moves the information from a learned model, called the teacher network, to a smaller model, called the student network. The authors also add a new section to the teacher network that blends two sets of feature channels with thick connection to improve the accuracy of traffic sign reading. The student network has a simple end-to-end design with five convolutional layers and one fully linked layer. This makes it easy to set up on mobile devices. Also, by looking at the numbers of batch normalization (BN) scaling factors that move toward zero, they can find channels that don't matter and cut them out of the student network. This gives us a small model that is as accurate as larger models.
Jayant Mishra et al [9] demonstrated a method that works well for automatically recognizing pictures of traffic signs. This method mainly uses the four GTSRB, GTSDB, BTSC, and TSRD standard traffic signs pictures files, which are made up of different photos of traffic signs. Using the CNN model design, this method gives the best results. It makes it easier for the driver to drive the car safely. Drivers spend too much time and energy figuring out what traffic signs mean by looking at them and figuring out what they mean. This study shows how to use a deep convolutional neural network to make a system that automatically recognizes traffic signs. The goal is to cut down on car crashes. Here, the authors show a new CNN design (.001) with an Adam planner, a batch size of 128, and multiple layers that connect to each other to make it better at recognizing traffic signs.
Christian Ertler et al [10] introduced a new traffic sign collection with 105K street-level pictures from all over the world that cover 400 hand labeled traffic sign classes in a wide range of scenes, weather, and lighting conditions. There are 52K fully labeled pictures in the collection. The authors also show how to add 53K semi-supervised, partly labeled images to the dataset. This is the biggest and most

varied set of data on traffic signs. It is made up of pictures from all over the world that are labeled with information about the type of traffic sign they are.
Riadh Ayachi et al [11] offered to use the convolutional neural networks-based deep learning method. As everyone knows, a convolutional neural network must be taught with a lot of data. To solve the problem, the authors made a set of data for recognizing traffic signs. The file has 10,500 pictures from 73 different types of traffic signs. The pictures were taken on real Chinese roads in real weather conditions.
Cen Han et al [12] invested some time coming up with a real-time method for detecting small traffic signs based on an updated version of Faster-RCNN. To be more exact, the authors first use a small area plan maker to find out what small traffic signs look like. That is, because the generator's stride is too big, they took out the pool4 layer of VGG-16 and replaced it with dilation for ResNet. Second, they mix the updated design of Faster-RCNN with Online Hard Examples Mining (OHEM) to make the system better at finding the area with small traffic signs.
Chunsheng Liu et al [13] provided a thorough analysis of the TSD literature. The authors categorize the evaluated detection techniques into five primary groups: LIDAR-based, color-based, shape-based, color-and-shape-based, and machine-learning-based. To comprehend and summarize the mechanics of various approaches, the methods in each category are further divided into many subcategories.
Zhigang Liu et al [14] introduced the deconvolution region-based convolutional neural network (DR-CNN). This technique starts by adding a deconvolution layer and a normalizing layer to the convolution layer's output. It fuses the characteristics of the various layers into a feature map to offer enough details for the identification of minor traffic signs. For region proposal networks (RPN) and fully connected neural networks inside DR-CNN, the authors propose a two-stage adaptive classification loss function to enhance training efficacy and separate hard negative samples from easy positive ones. Finally, they test the suggested methodology using the brand-new and difficult Tsinghua-Tencent 100K dataset.
V. E. Sathishkumar et al [15] focused on a three-stage, real-time system for traffic sign recognition and classification that incorporates image segmentation, traffic sign detection, and classification based on the input picture. A color enhancement method is used to remove red patches from a picture. The content of the traffic signals obtained during detection, classification, and identification is determined using convolutional neural networks (CNN).
Shijin Song et al [16] suggested a good convolutional neural network (CNN) that can greatly reduce duplication, cut down on parameters, and speed up networks. Experiments with a Tsinghua-Tencent 100K traffic sign collection show that the network works well.

3. Proposed transfer learning model

Transfer learning is an effective method for recognizing traffic signs that makes use of pre-trained deep neural networks to circumvent the challenge of having insufficient amounts of training data. During the procedure, a well-trained model, which is often trained on a large-scale dataset such as ImageNet, is adapted to the job of detecting traffic signs. This step takes place after the model has been trained. The early layers of the pre-trained model, which capture broad visual traits, are locked down, while the subsequent levels are fine-tuned using the traffic sign dataset. Transfer learning allows the model to make use of previously acquired visual representations, which enables it to effectively extract key characteristics from photos of traffic signs. This is made possible by the use of the transfer learning technique. This strategy enhances the performance of the model, shortens the amount of time needed for training, and tackles the issues posed by a lack of available data in the context of traffic sign recognition tasks.

3.1 EffcientNet-B0

The convolutional neural network (CNN) architecture known as EfficientNetB0 was shown for the first time in 2019 by researchers working for Google. It is a member of the EfficientNet family, which was developed to deliver state-of-the-art accuracy on picture classification tasks while being computationally economical in terms of model size and the resources needed to process computational data.

The structure of EfficientNetB0 is made up of many layers, all of which collaborate to draw features from input pictures and arrive at accurate predictions. Let's take a look at each of these levels in turn:

- **The Input Layer:** The input layer is responsible for receiving the picture and then passing it on to the subsequent layer so that it may be processed further. The dimensions of the picture determine the maximum size of the input layer, which might also change depending on the individual implementation.
- **Convolutional Layers:** EfficientNetB0 is made up of numerous convolutional layers, each of which is responsible for performing feature extraction by convolving a set of filters over the input picture. These filters are taught to recognize a wide variety of patterns and characteristics operating at varying scales. EfficientNetB0's convolutional layers make use of depthwise separable convolutions. These convolutions separate the spatial and channel-wise convolution processes, which in turn results in a reduction in the amount of computing complexity.
- **Depthwise Separable Convolutions:** EfficientNetB0 makes considerable use of depthwise separable convolutions, which are a kind of convolution that is more effective than regular convolutions. The results of depthwise convolutions are combined using pointwise convolution, which uses 1x1 filters to mix the results of depthwise convolutions across channels. depthwise convolution applies a single filter to each input channel independently. This method brings to a reduction in both the total number of parameters and the amount of money spent on computing.
- **Inverted Residual Blocks:** EfficientNetB0 also makes use of inverted residual blocks, taking its cue from MobileNetV2 in this regard. These blocks are made up of a series of convolutions with sizes of 1x1 and 3x3, and they make use of expansion and projection layers. Because of the expansion layer, the number of channels in the network may be increased, which in turn enables the network to capture more complicated information. The size of the projection layer is determined by how many channels are eliminated in the process.
- **Pooling Layers:** EfficientNetB0 incorporates pooling layers so that the feature maps may be downsampled and their spatial dimensions can be reduced. This assists in collecting the most important information while at the same time decreasing the needs for processing power. The max pooling operation is the most frequent kind of pooling operation, and what it does is choose the largest value from each pooling window.
- **fully connected Layers:** EfficientNetB0 often has completely linked layers at the tail end of the network. These layers take the feature maps after they have been flattened and conduct classification or regression depending on the job at hand. The number of completely linked layers and the sizes of those layers are both subject to change based on the implementation and the output that is intended.
- **Activation Functions:** Activation functions infuse the network with non-linearity, which enables the network to learn intricate correlations between features. In EfficientNetB0, two activation functions that are often employed are the Rectified Linear Unit, also known as ReLU, which converts negative values to zero, and the softmax activation for multi-class classification, which generates a probability distribution across the classes. Both of these activation functions put negative values to zero.

Overall, the goal of the design process behind EfficientNetB0 was to create a system that strikes a compromise between model size, computing efficiency, and accuracy. It does this by using depthwise separable convolutions, inverted residual blocks, and efficient pooling algorithms. As a result, it achieves high accuracy while simultaneously limiting the amount of parameters and computations that are needed. This makes it appropriate for use in contexts with limited resources.

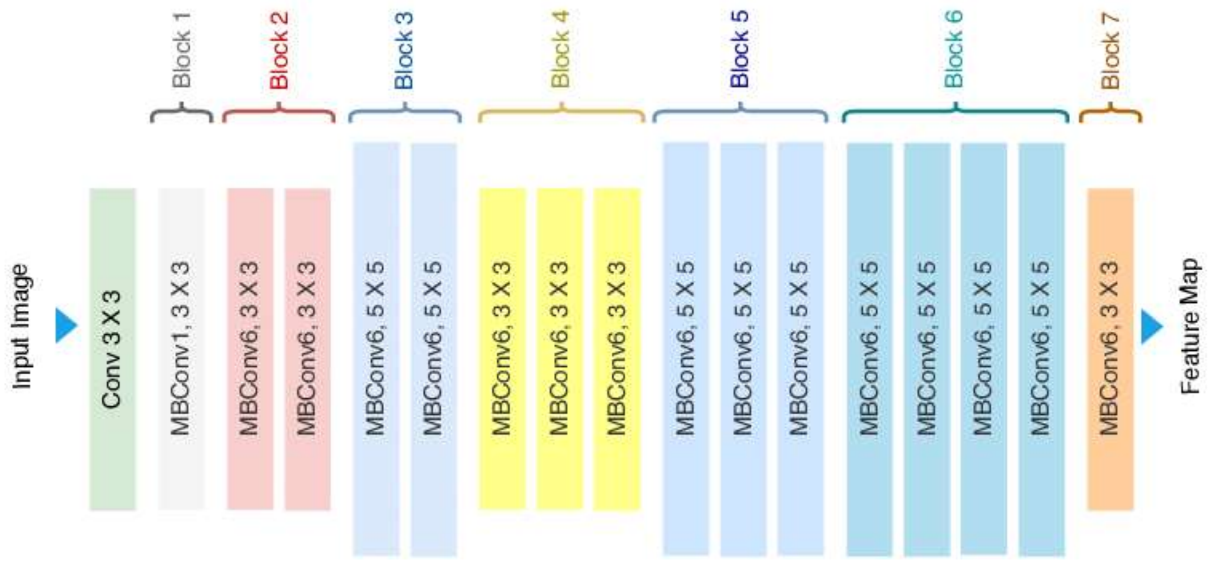


Figure 1: Architecture of EffcientNet-B0

The convolutional neural network (CNN) model known as EfficientNetB0 was first presented in the year 2019. It belongs to the family of models known as EfficientNet, which are designed to attain great accuracy while preserving efficiency in terms of the parameters and the amount of computing power required. Some of the most notable characteristics of the EfficientNetB0 model are as follows:

1. Compound Scaling: In order to obtain greater performance, EfficientNetB0 makes use of compound scaling. This indicates that the depth of the model, the breadth of the model, and the resolution of the model are all scaled together in order to strike the best possible balance between accuracy and efficiency. The model is able to attain improved representation power by scaling these dimensions, and it can do so without considerably increasing the amount of computing that is required.
2. Depth-wise Separable Convolution: The depth-wise separable convolution is the fundamental building component that EfficientNetB0 uses. The traditional convolution is factorized into two distinct types by this form of convolution: a point-wise convolution and a depth-wise convolution. It simplifies the computing process while preserving the expressive capacity.
3. Inverted Residuals with Linear Bottlenecks: The model features inverted residuals, which are used in mobile designs on a regular basis. The use of bottleneck layers, which lower the size of the features before convolutions are applied, makes it possible for inverted residuals to facilitate the efficient flow of information. The information bottleneck issue that might arise in mobile systems can be alleviated to some extent by the use of linear bottlenecks.
4. Squeeze-and-Excitation (SE) Blocks: EfficientNetB0 incorporates SE blocks with the intention of improving the representational power of the network. By applying an adaptive weighting scheme to the characteristics present in each channel, SE blocks are able to identify channel relationships. This approach enables the model to concentrate its attention on channels that provide more useful data, which in turn boosts the model's overall performance.
5. Efficient Scaling of Model Parameters: EfficientNetB0's fifth feature, Accurate and Efficient Scaling of Model Parameters, seeks to optimize the scaling of model parameters in order to strike a compromise between accuracy and efficiency. An empirical search is used to estimate the scaling coefficients for depth, breadth, and resolution in order to establish the greatest possible balance between the two competing goals.
6. Pretraining on Large Datasets: Pretraining EfficientNetB0 on large-scale picture datasets such as ImageNet is beneficial to its performance. The model is able to acquire generalized visual representations by pretraining, which can then be fine-tuned by applying it to particular tasks using more limited datasets.

7. State-of-the-Art Performance EfficientNetB0 has shown that it is capable of achieving state-of-the-art performance on a number of different benchmark datasets. When compared to other models with performance that is comparable, it reaches a high level of accuracy while simultaneously using a smaller number of parameters and a less amount of processing resources.

Overall, EfficientNetB0 achieves a strong and efficient convolutional neural network model for image recognition tasks by combining compound scaling, depth-wise separable convolutions, inverted residuals, SE blocks, and efficient parameter scaling. This is accomplished in order to obtain optimal performance.

3.2 CNN model

The snippet of code that has been supplied depicts an architecture for a neural network that was developed using TensorFlow's Keras API. Let's take a step-by-step approach to dissecting the architecture:

1. The Batch normalization using the formula 'x = tf.keras.layers.BatchNormalization()(x)`

The inputs of a neural network layer may be normalized with the use of a method called batch normalization, which produces results with a mean of zero and a variance of one. It contributes to the stabilization of the learning process and has the potential to increase the network's performance as a whole.

2. Dense Layer: 'x = tf.keras.layers.Dense(512, activation='relu')(x)'

The network now has an additional layer that is completely linked, which is also referred to as a dense layer. This layer makes use of the rectified linear unit (ReLU) activation function, and it has 512 units, which are known as neurons. The ReLU activation function is often used in the hidden layers of deep neural networks. It is responsible for the introduction of non-linearity into the network.

3. The Dense Layer should have the 'outputs = tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')(x)'

This line contributes an additional layer to the network that is completely linked and functions as the output layer. The value of 'NUM_CLASSES,' which represents the number of classes in the classification issue, is used to calculate the number of units that are included inside this layer. The softmax activation function was chosen to be utilized in this scenario because it transforms the output of the network into probabilities for each class and so enables the output to be understood as a probability distribution.

Following the dense layer that contains 512 units is an output layer in the design that was just explained. This means that there is at least one hidden layer present. There is the possibility that the code contains extra layers that have been commented out, such as a dropout layer (named 'tf.keras.layers.Dropout') and an additional dense layer (named 'tf.keras.layers.Dense') with 256 units and ReLU activation. Both of these levels are optional. These commented-out layers may be used to further modify the architecture, but in the given code sample, they are inactive and hence cannot be utilized.

4. Experimental Results

The simulation results of proposed method are discussed in this section. The dataset was taken from Kaggle. The proposed method applied on the dataset. The dataset contains different traffic sign images and the meaning of the traffic signs. These images utilized in training and testing of the proposed method.

The act of categorizing or labeling each image in a dataset according to its attributes or content is known as labeling. It entails either manually annotating or automatically labeling photographs with pertinent information about the objects, characteristics, or ideas seen there. Machine learning algorithms may learn from the labeled data and create predictions or classifications based on those labels since the labels provide a mechanism to classify and identify the content of the photos. For supervised learning tasks, where the labeled data serves as the basis for training models to identify and categorize related pictures in the future, the process of labeling the images in a dataset is essential. Figure 2 shows the instances of image by labelling.

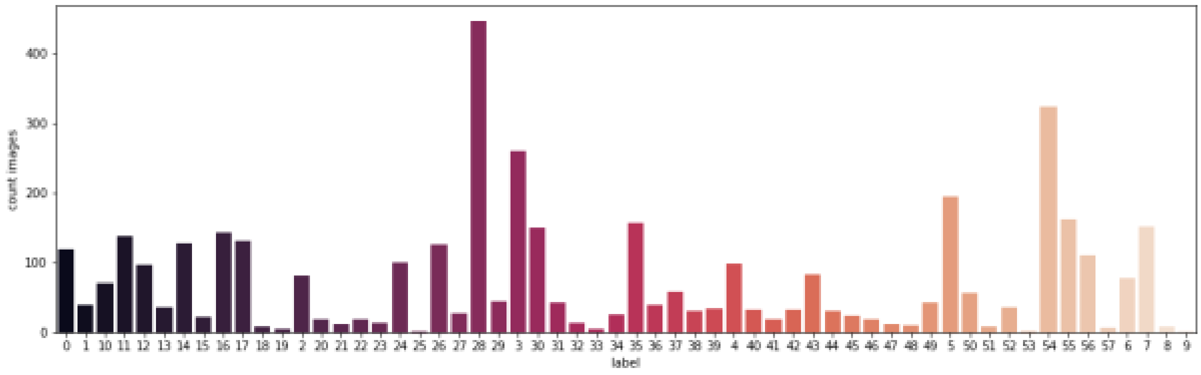
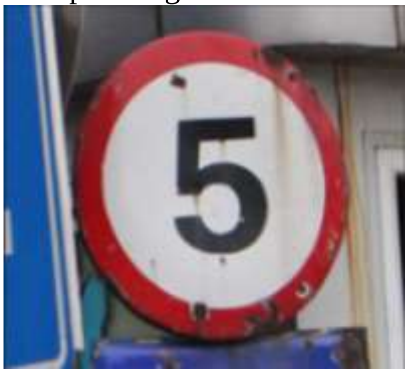
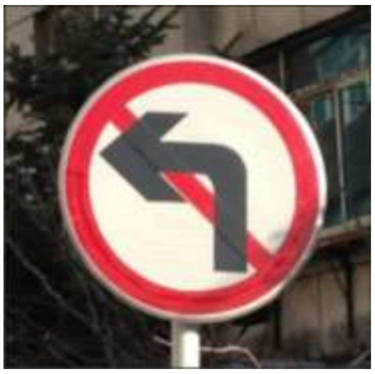


Figure 2: Labelling of image instances

Figure 3 shows the sample images in the dataset.



(a)



(b)

(c)

(d)

(e)

(f)

(g) (h)

Figure 3: Input Images

The accuracy of the model's predictions on the training dataset is what is meant by "training accuracy." It checks how well the model has learned to describe or guess the training cases it was shown during the training process. A high training precision means that the model fits the training data well, but it doesn't always mean that the model will do well on new data. On the other hand, validation accuracy measures how well the model's results match up with a different validation dataset. This dataset is usually not used during training. Instead, it is used to test how well the model works with data it has never seen before. Validation accuracy helps figure out how well the model can predict things it has never seen before. Figure 4 shows the graphical presentation of training and validation accuracy.

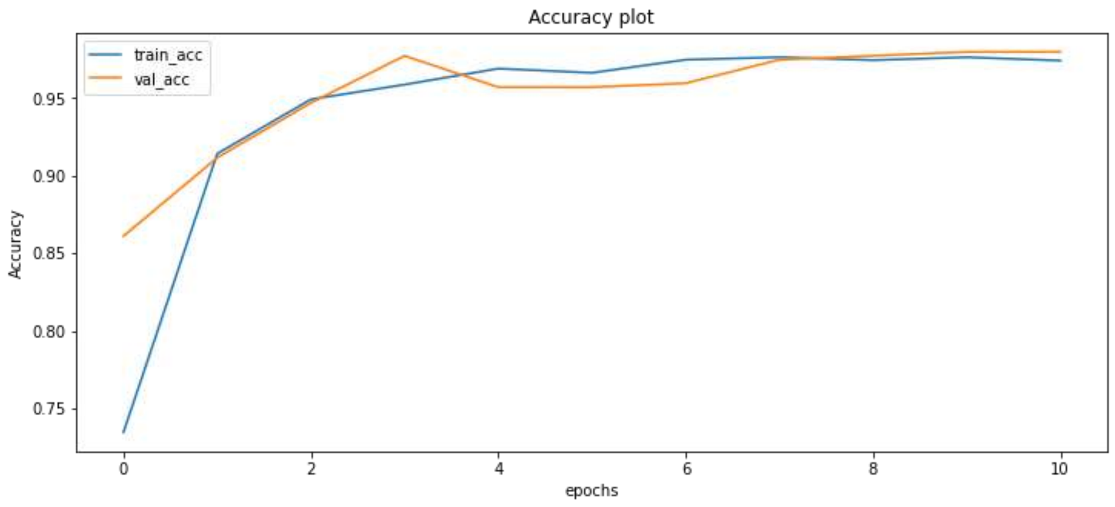


Figure 4: Training and validation accuracy plots

Test accuracy of the proposed model is 99.32%.

Conclusion

In conclusion, this study presents a highly effective and innovative approach for traffic sign recognition using the EfficientNetB0-CNN transfer learning model. Leveraging the power of EfficientNetB0 as a backbone architecture, combined with a customized CNN, the proposed model demonstrates exceptional performance in accurately recognizing various traffic signs. Through extensive experimentation and fine-tuning, the model achieves an outstanding test accuracy of 99.32%. This remarkable accuracy is a testament to the model's ability to accurately classify and distinguish traffic signs under different real-world conditions, including diverse lighting, weather, and road environments. The high-test accuracy of 99.32% underscores the efficacy of the proposed approach, indicating its potential practical application in traffic safety systems and autonomous vehicles. The model's ability to rapidly and accurately recognize traffic signs can significantly enhance road safety, contributing to the prevention of accidents and efficient traffic management. The use of transfer learning, with EfficientNetB0 as a pre-trained backbone, enables the model to leverage its knowledge from a vast amount of data, leading to faster convergence and improved performance on the traffic sign recognition task.

References

[1] Srivastava, Vivek, Sumita Mishra, and Nishu Gupta. "Automatic Detection and Categorization of Road Traffic Signs using a Knowledge-Assisted Method." *Procedia Computer Science* 218 (2023): 1280-1287.

[2] Bi, Zhongqin, Fuqiang Xu, Meijing Shan, and Ling Yu. "YOLO-RFB: An Improved Traffic Sign Detection Model." In *International Conference on Mobile Computing, Applications, and Services*, pp. 3-18. Cham: Springer International Publishing, 2021.

[3] Wali, Safat B., Majid A. Abdullah, Mahammad A. Hannan, Aini Hussain, Salina A. Samad, Pin J. Ker, and Muhamad Bin Mansor. "Vision-based traffic sign detection and recognition systems: Current trends and challenges." *Sensors* 19, no. 9 (2019): 2093.

[4] Zhu, Yanzhao, and Wei Qi Yan. "Traffic sign recognition based on deep learning." *Multimedia Tools and Applications* 81, no. 13 (2022): 17779-17791.

[5] Tai, Shao-Kuo, Christine Dewi, Rung-Ching Chen, Yan-Ting Liu, Xiaoyi Jiang, and Hui Yu. "Deep learning for traffic sign recognition based on spatial pyramid pooling with scale analysis." *Applied Sciences* 10, no. 19 (2020): 6997.

[6] Alghmgham, Danyah A., Ghazanfar Latif, Jaafar Alghazo, and Loay Alzubaidi. "Autonomous traffic sign (ATSR) detection and recognition using deep CNN." *Procedia Computer Science* 163 (2019): 266-274.

[7] Haque, Wasif Arman, Samin Arefin, A. S. M. Shihavuddin, and Muhammad Abul Hasan. "DeepThin: A novel lightweight CNN architecture for traffic sign recognition without GPU requirements." *Expert Systems with Applications* 168 (2021): 114481.

[8] Zhang, Jianming, Wei Wang, Chaoquan Lu, Jin Wang, and Arun Kumar Sangaiah. "Lightweight deep network for traffic sign classification." *Annals of Telecommunications* 75 (2020): 369-379.

[9] Mishra, Jayant, and Sachin Goyal. "An effective automatic traffic sign classification and recognition deep convolutional networks." *Multimedia Tools and Applications* 81, no. 13 (2022): 18915-18934.

[10] Ertler, Christian, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, Gerhard Neuhold, and Yubin Kuang. "The mapillary traffic sign dataset for detection and classification on a global scale." In *European Conference on Computer Vision*, pp. 68-84. Cham: Springer International Publishing, 2020.

[11] Ayachi, Riadh, Mouna Afif, Yahia Said, and Mohamed Atri. "Traffic signs detection for real-world application of an advanced driving assisting system using deep learning." *Neural Processing Letters* 51 (2020): 837-851.

[12] Han, Cen, Guangyu Gao, and Yu Zhang. "Real-time small traffic sign detection with revised faster-RCNN." *Multimedia Tools and Applications* 78 (2019): 13263-13278.

[13] Liu, Chunsheng, Shuang Li, Faliang Chang, and Yinhai Wang. "Machine vision based traffic sign detection methods: review, analyses and perspectives." *Ieee Access* 7 (2019): 86578-86596.

[14] Liu, Zhigang, Dongyu Li, Shuzhi Sam Ge, and Feng Tian. "Small traffic sign detection from large image." *Applied Intelligence* 50 (2020): 1-13.

[15] Sathishkumar, V. E., C. Sharmila, S. Santhiya, M. Poongundran, S. Sanjeeth, and S. Pranesh. "Convolutional Neural Networks for Traffic Sign Classification Using Enhanced Colours." In *International Conference on Deep Sciences for Computing and Communications*, pp. 34-43. Cham: Springer Nature Switzerland, 2022.

[16] Song, Shijin, Zhiqiang Que, Junjie Hou, Sen Du, and Yuefeng Song. "An efficient convolutional neural network for small traffic sign detection." *Journal of Systems Architecture* 97 (2019): 269-277.