Using Multiple CNN Model Develop Bangla Number Plate Recognition System

1st Md Shakil
Department of Computer Science and Engineering
European University of Bangladesh)
Dhaka, Bangladesh
shakil.eub.cse@gmail.com

2nd Umayer Mohammad Affan
Department of Computer Science and Engineering
European University of Bangladesh)
Dhaka, Bangladesh
umaffan18@gmail.com

3rd Habibur Rahman
Department of Computer Science and Engineering
European University of Bangladesh
Dhaka, Bangladesh
hr03070@gmail.com

4th Afruja Sultana Muniya
Department of Computer Science and Engineering)
European University of Bangladesh
Dhaka, Bangladesh
muniyasultana289@gmail.com

***Abstract*—Automatic Number plate recognition (ANPR) has become an important field of research in computer vision due to its vast practical applications. The research focuses on developing an automatic Bangla number plate recognition system using Convolutional Neural Networks (CNNs). The proposed system aims to address the challenges faced in recognizing Bangla number plates. The system utilizes a dataset of 1440 different vehicle Bangla number plates, which is split into 70% for training, 20% for validation, and 10% for testing. This proposed model is built using eight pre-trained models, including VGG16, VGG19, DenseNet201, ResNet50, MobileNetv2, InceptionResNetv2, Inceptionv3, and Xception, and also two custom models are Max pooling and Average pooling with three Convolution Layer. The proposed models achieved the highest accuracy by applying the Custom Max Pooling operation with 99.53% on the training dataset. The MobileNetV2 Model achieved 99.38%, ResNet50 achieved 99.33% accuracy, VGG16 achieved 98.92%, VGG19 achieved 98.87%, Average Pooling achieved 98.71%, InceptionV3 achieved 98.01%, Xception achieved 97.89%, DenseNet201 achieved 97.74% and InceptionResNetV2 achieved 96.25% training accuracy.**

***Index Terms*—VGG16, VGG19, DenseNet201, ResNet50, MobileNetv2, InceptionResNetv2, Inceptionv3, Xception, Max Pooling, Average Pooling, Number Plate, Recognition, CNN**

I. Introduction

In 2022, there were more than 6700 traffic collisions and accidents, and 45% of the vehicles involved were unfit for the road [1]. In Bangladesh, the issue of vehicle problems is a major concern for both the government and citizens. The country's roads are often overcrowded, poorly maintained, and lack proper infrastructure, leading to frequent traffic jams and accidents. The problem of vehicle-related issues, including traffic congestion, road accidents, and illegal activities involving vehicles, is a significant concern. Number plate recognition is important for several reasons. One of the main reasons is that law enforcement agencies can use it to track and identify vehicles involved in criminal activities. It can help to enforce traffic laws and regulations. By capturing images of number plates, the technology can identify vehicles that are speeding, running red lights, or violating other traffic laws. This can lead to the issuance of tickets and fines, which can help to reduce traffic violations and improve road safety. So number plate recognition is an important tool for law enforcement and traffic management. Bengali license plate detection and recognition with discernment is more difficult for machines because of the ambiguity in the language [2].

Identify applicable funding agency here. If none, delete this.

II. Literature Review

This paper [3], essentially four computing phases make up the model. The first stage is pre-processing the obtained picture, which is essential to achieving better results in subsequent phases. Second, number plate extraction using the preprocessed picture to find the location of the number plate. Third, use the boundary box feature from the extracted plate to segment the characters. Recognize the segmented characters using template matching as the last step. In this paper [4], they suggest a two-stage detection pipeline coupled with the Vision API, which offers fast real-time inference speed in addition to dependably precise detection and identification performance. We added a filter called a haarcascade classifier on top of our main MobileNet SSDv2 detection model. The authors of this study [5] devised a technique that had a detection accuracy of 94%. The four modules that make up the suggested technique are preprocessing, processing of the license plates, identifying the characters on the plates, and determining whether or not the vehicle is registered. In this research [6], To recognize the number plate, a novel PSO method is implemented. MATLAB and numerous image processing examples are used to create the system. In this paper the car number plate image is obtained by the cameras. Here, we are using five phase for number plate detection and recognition. The researchers of this paper [7], introduce an Automatic Number Plate Recognition

System (ANPR) employing edge detection techniques, histogram manipulation, and morphological procedures for segmenting characters and locating plates. Character recognition and categorization employ artificial neural networks.

III. Research Methodology

In this research, we proposed a system using Convolutional Neural Networks (CNNs) for number plate recognition. Convolutional Neural Network (CNN) is one of the most popular deep neural networks. The name comes from a mathematical linear operation between matrices called convolution. CNN have multiple layers; including convolutional layer, non-linearity layer, pooling layer and fully-connected layer [8]. Paramters are available in convolutional and fully connected layers, while pooling and nonlinear layers have no parameters. This proposed model is built using nine CNN models, while seven pre-trained models, including VGG16, VGG19, ResNet50, MobileNetv2, Inceptionv3, InceptionResNetv2, and Xception, and also two custom models are Max pooling and Average pooling with three Convolution layers. To accomplish this task, the system uses Bengali number plate images of 1440 different vehicles. Fig. 1 represents the overall work process of the proposed method performed in this study. At first, we collect data, then preprocess the data and then train our model for training every we applied total 100 epochs.

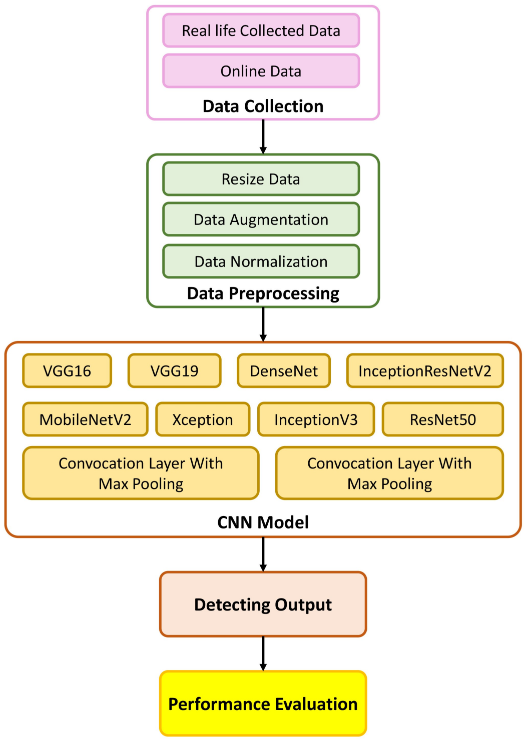


Fig. 1: Proposed System Diagram

A. Data Collection

Data collection is the method of gathering and measuring information on specific variables of interest, in a systematic and objective way. In this research, we collect data for the number plate recognition system. We collect data from real-life and online sources. Real-life data collection involves capturing images of number plates from various locations, such as parking lots, roads, highways etc. Online data collection involves scraping number plate images from publicly available sources, here we collect data from roboflow website. The dataset contains 1440 images. We split this dataset into 3 categories such as training set, validation set, and test set. Table I represents the split dataset information.

TABLE I: Dataset Outlook of this research

Dataset	No. of Images
Training Set	1007
Validation Set	288
Test Set	145
Total	**1440**

B. Data Preprocessing

The global objective of data preprocessing is to remove the unwanted variability or effects from the signal so that the useful information related to the property(-ies) of interest can be used for efficient modeling [9]. The specific objectives of the preprocessing techniques are dependent on the type of artefacts to be dealt with [9]. The goal of data preprocessing is to improve the quality of data, make it more organized and easier to analyze, and increase the accuracy of machine learning models. Image preprocessing is an essential step in research involving image analysis. In this proposed model authors resize the input images to the desired input shape of (224, 224, 3) using OpenCV's cv2.resize() function. This is done to ensure that all input images are of the same size and can be fed into the neural network.

C. Proposed CNN Model

In this proposed system we used Convolutional Neural Network(CNN) architecture. The CNN has an excellent performance in machine learning problems. Specially the applications that deal with image data, such as the largest image classification data set (Image Net), computer vision, and natural language processing (NLP), and the results achieved were very amazing [8]. There are many pre-train models available in CNN model. In this proposed system we used eight pre-train models and two custom models. The proposed system achieved the highest 99.58% accuracy by applying a custom Max Pooling model. Pre-train model such as VGG16, VGG19, DenseNet, InceptionResNetV2, MobileNetV2, Xception, Inception, and ResNet50. From this pre-train model ResNet50 achieved the highest accuracy, it achieved 99.33% accuracy. Two custom models are Max pooling and Average pooling with three convolution layers. Below describe the proposed system model.

1) VGG16 Architecture: The VGG16 architecture is a pre-trained convolutional neural network (CNN) that is commonly used as a base model for transfer learning. The VGG16 architecture consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. All the convolutional layers use a 3x3 filter size and a stride of 1, while the pooling layers use a 2x2 filter size and a stride of 2. The network takes an input image of size 224x224x3 (i.e., width x height x RGB channels). The convolutional layers are followed by two fully connected layers of size 4096, and the output layer is a softmax layer of size 1000, which is the number of classes in the ImageNet dataset. In this proposed system, the VGG16 model is imported from the Keras Applications module and its base layers are frozen. This means that the pre-trained weights are retained and not updated during training. The last convolutional block of the VGG16 model is then unfrozen to allow fine-tuning of the weights. A custom head is added to the base model, which consists of several dense layers. In the base model custom head first line add Flatten layer, second line adds a dense layer with 128 neurons and ReLU activation. The third line adds a dense layer with 64 neurons and ReLU activation and last line adds a dense layer with 4 neurons and a linear activation function. Finally, the model is compiled with a mean squared error (MSE) loss function and the Adam optimizer with a learning rate of 0.0001. Fig. 2 represents the work process of VGG19 model.

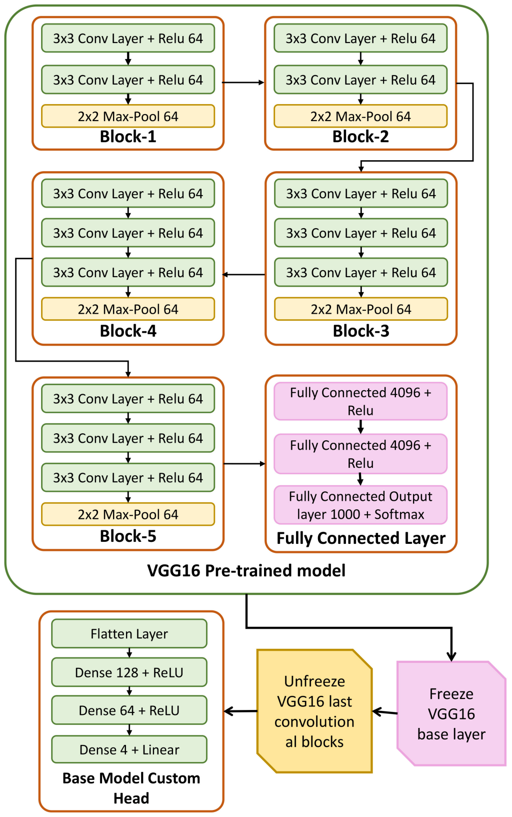


Fig. 2: VGG16 Pre-trained Model Architecture

2) VGG19 Architecture: All pre-trained models are implemented as vgg16. The VGG19 architecture consists of 19 layers, including 16 convolutional layers, 3 fully connected layers, and an input layer. 16 convolutional layers organized into 5 blocks, each consisting of multiple convolutional layers followed by a max-pooling layer. First two blocks, each block consists of two 2D convolutional layers and a max-pooling layer. Also last three blocks, each block This block consists of four 2D convolutional layers and a max-pooling layer. After these five blocks, there are three fully connected layers with 4096 neurons each, followed by a final fully connected layer with 1000 neurons, corresponding to the number of ImageNet classes.

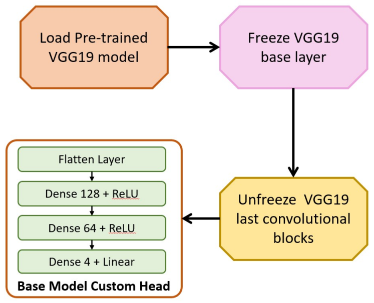


Fig. 3: VGG19 Pre-trained Model Architecture

3) DenseNet201 Architecture: DenseNet201 is a deep neural network that consists of 201 layers, and it is similar in structure to the original DenseNet architecture, but with more layers and more filters in each convolutional layer. The Dense Convolutional Network (DenseNet) architecture connects each layer directly to the other in a feed-forward fashion [10]. DenseNet contains a very narrow layer (12 filters per layer) with a small set of feature maps included in the network's collective information [11]. Three transition layers and four dense blocks make up the DenseNet201 design. A convolutional layer with 64 filters of size 7x7 and stride 2 is present in the first dense block. The next four dense blocks each include 256, 128, 64, and 32 filters. A convolutional layer with 128 filters of size 1x1 and stride 1 makes up the first transition layer. This layer is followed by an average pooling layer with filters of size 2x2 and stride 2. The last layer, which is completely linked, uses the global descriptor created by the global average pooling layer to produce the output. Fig. 4 represents the work process of DenseNet201 model.

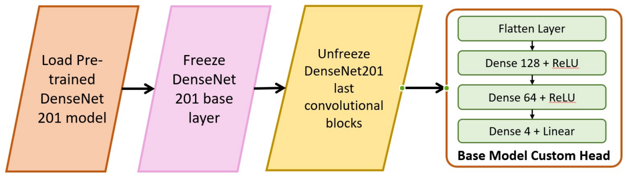

Load Pre-trained DenseNet 201 model
Freeze DenseNet 201 base layer
Unfreeze DenseNet201 last convolutional blocks
Flatten Layer
Dense 128 + ReLU
Dense 64 + ReLU
Dense 4 + Linear
Base Model Custom Head

Fig. 4: DenseNet201 Pre-trained Model Architecture

4) InceptionResNetV2 Architecture: InceptionResNetV2 is a pre-trained model. The InceptionResNetV2 architecture consists of repeated blocks of Inception modules and residual connections, which are referred to as InceptionResNet blocks. There are 9 blocks in the InceptionResNetV2 architecture, each with a different number of Inception modules and residual connections. The first block is called the Stem block. The stem consists of a series of convolutional and pooling layers that preprocess the input image. From the remaining 8 blocks, three InceptionResNet blocks called the InceptionResNet-A, InceptionResNet-B, and InceptionResNet-C. This block consists of a set of parallel branches, each containing a different type of convolutional layer. These branches are then concatenated to produce the output of the block. Two Reduction blocks are called- Reduction-A and Reduction-B. This block reduces the spatial dimension of the feature map and increases the number of filters. Then applied one Average Pooling layer, one Droput layer, and one Softmax layer.

5) MobileNetV2 Architecture: MobileNetV2 is designed to be efficient and lightweight for mobile and embedded vision applications. MobileNetV2 is consisting of 19 layers of bottleneck [12]. OpenCV, which uses ResNet-10 in the base model [12]. For number plate recognition, MobileNetV2 can be used as a feature extractor to identify and localize number plates in an image. MobileNetV2 model is used as a base model by setting include-top to False and loading the pre-trained ImageNet weights [13]. A custom head is then added to this architecture, consisting of three dense layers with ReLU activation and a linear output layer.

6) Xception Architecture: Xception is a deep convolutional neural network architecture proposed by François Chollet in 2016. This model uses the concept of Depthwise Separable Convolutions [14]. Depthwise Separable Convolutions are modifications to usual convolutions which expected to be more efficient in terms of computation time [14]. Xception also uses residual connections to improve training stability and performance. The network consists of 36 convolutional layers organized into 14 modules, with skip connections between certain layers to help information flow more easily through the network.

7) InceptionV3: Inceptionv3 is a convolutional neural network (CNN) architecture that was developed by Google for image classification tasks. It is a deep learning architecture that uses multiple layers of convolutional and pooling layers to extract relevant features from images. For number plate recognition, Inceptionv3 can be used as a feature extractor to extract relevant features from license plate images. The extracted features can then be used to train a machine learning algorithm or a deep neural network to recognize license plates. It is a commonly used image recognition model [15]. The model is the culmination of many ideas developed by multiple researchers over the years [16].

8) ResNet50 Architecture: ResNet50 is a deep neural network architecture that was developed by Microsoft. On the deep residual learning framework, ResNet is developed [17]. Even with exceedingly deep neural networks, the vanishing gradient problem is resolved. ResNet50 are divided into several different stages. Each stage consists of multiple residual blocks, which allow for the training of very deep neural networks. ResNet-50, with 50 layers is one of the variants of ResNet [17]It has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer [18].

9) Custom Max Pooling Model: For number plate recognition, we used custom Max Pooling model with three convolutional layers. The first convolutional layer has 32 filters, a kernel size of (3,3), and an activation function of 'relu'. Max pooling helps to gather significant information and reduces the size of the images [13]. The data is transferred to the second convolution layer after the first convolution is complete. The second convolutional layer has 64 filters, a kernel size of (3,3), and an activation function of 'relu'. The third convolutional layer has 128 filters, a kernel size of (3,3), and an activation function of 'relu'. After the third convolutional layer, the data is flattened, and the fully connected layers are added. The first fully connected layer has 256 units and an activation function of 'relu'. The final layer has 4 units and an activation function of 'linear'. Our CNN model initiates with Keras.models.sequential() [13].

10) Custom Average Pooling Model: The average pooling operation is used in the same way as the max pooling model. This operation also used three convolutional layer. In hidden layer also used Relu activation function and Linear function used in fully connected layer. he model is compiled with the 'mse' loss function and the Adam optimizer with a learning rate of 0.0001.

D. *Performance Evaluation*

Performance evaluation is a process of measuring the performance of a model. In this proposed system, the performance of a model is evaluated on the validation, training, and test sets using the evaluate() method of the model object. The evaluate() method calculates the loss and accuracy of the model on the given data and returns them as output.

IV. Experimental Result Analysis

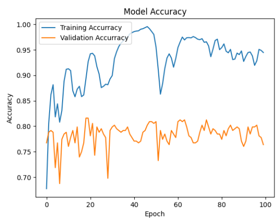
After training to understand the model proficiency, we test our model using a test dataset. In this research, we used total of 1440 bangla vehicle number plate images. In this proposed system, we used different types of CNN pre-trained and custom models. The highest accuracy of this proposed system is 99.58% in the training data set. For every proposed model we run up to 100 epochs. From 100 epochs we present some epochs randomly of training accuracy and training loss as a table.

A. *Custom Max Pool Operation Result Analysis*

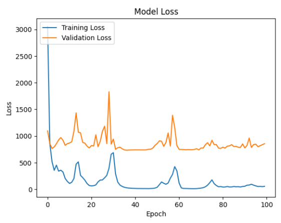
Applying Max Pooling operation with three convolutional layers we get the highest accuracy 99.53% on that time training loss is 12.62. After running 47th epoch accuracy doesn't improve. Table II represents the training accuracy of Max Pooling Operation. Fig.5 shows the graphical representation of the training accuracy and validation accuracy and also shows training loss and validation loss by applying Max Pooling operation.

TABLE II: Epochs Outcomes Applying Max Pooling Operation

Epoch	Training Loss	Training Accuracy
01	3052	67.75%
02	858.07	80.54%
11	110.10	91.27%
22	68.16	94.30%
37	29.07	97.28%
41	16.98	98.56%
47	12.62	99.53%



(a) Model Accuracy Graph



(b) Model Loss Graph

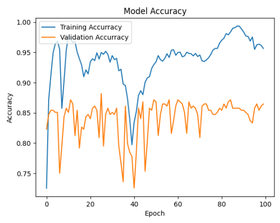
Fig. 5: By using the Custom Max Pooling Operation, a graphical representation accuracy of training and validation, along with training loss and validation loss.

B. *ResNet50 Model Result Analysis*

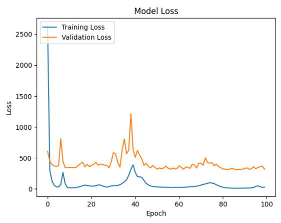
Applying CNN ResNet50 model achieved 99.33% training accuracy at that time training loss is 10.22. After 88 epochs this model achieved the highest accuracy. The training accuracy and loss of the ResNet5o Model are shown in Table III for each epoch. Fig.6 shows the graphical representation of the training accuracy and validation accuracy and also shows training loss and validation loss by applying ResNet50 Model.

TABLE III: Epochs Outcomes Applying ResNet50 Model

Epoch	Training Loss	Training Accuracy
01	2630	72.25%
02	297.39	87.17%
03	126.03	91.27%
06	31.33	96.87%
11	15.44	97.74%
83	15.11	98.10%
88	10.22	99.33%



(a) ResNet50 Model Accuracy Graph



(b) ResNet50 Model Loss Graph

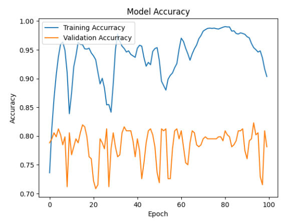
Fig. 6: By using the ResNet50 Model, a graphical representation accuracy of training and validation, along with training loss and validation loss.

C. *MobileNetV2 Model Result Analysis*

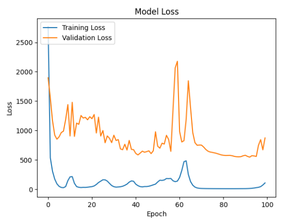
MobileNetV2 model achieved 99.03% training accuracy on that time training loss is 8.42. After 81 epochs this model achieved the highest accuracy. After run the 81th epoch accuracy doesn't improve from 99.03% The training accuracy and loss of the ResNet5o Model are shown in Table **??** for each epoch. Fig.7 shows the graphical representation of the training accuracy and validation accuracy and also shows training loss and validation loss by applying MobileNetV2 Model.

TABLE IV: Epochs Outcomes Applying MobileNetV2 Model

Epoch	Training Loss	Training Accuracy
01	2768.39	73.601%
02	534.57	81.26%
06	31.33	96.87%
32	35.00	97.74%
71	12.23	98.25%
74	9.80	98.76%
81	8.42	99.03%



(a) MobileNetV2 Model Accuracy Graph



(b) MobileNetV2 Model Loss Graph

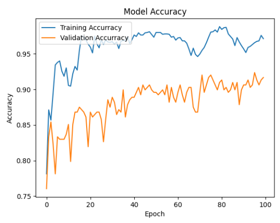
Fig. 7: By using the MobileNetV2 Model, a graphical representation accuracy of training and validation, along with training loss and validation loss.

D. *VGG19 Model Result Analysis*

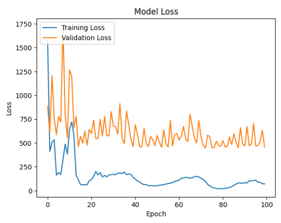
VGG19 model achieved 98.81% training accuracy on that time training loss is 19.57. After running the 80th epoch this model achieved the highest accuracy. After running the 80th epoch accuracy doesn't improve from 98.81% The training accuracy and loss of the VGG19 Model are shown in Table-V for each epoch. Fig.8 shows the graphical representation of the training accuracy and validation accuracy and also shows training loss and validation loss by applying VGG19 Model.

TABLE V: Epochs Outcomes Applying VGG19 Model

Epoch	Training Loss	Training Accuracy
01	1569.26	78.13%
02	410.5824	87.11%
07	170.09	93.99%
45	50.20	97.99%
48	52.02	98.10%
78	22.43	98.35%
80	19.57	98.81%



(a) VGG19 Model Accuracy Graph



(b) VGG19 Model Loss Graph

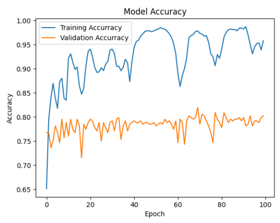
Fig. 8: By using the VGG19 Model, a graphical representation accuracy of training and validation, along with training loss and validation loss.

E. *Custom Average Pool Operation Result Analysis*

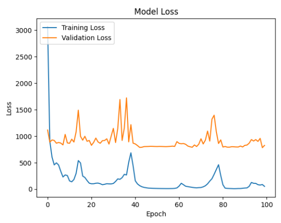
Applying the Average Pooling operation with three convolutional layers we get 98.71% accuracy on that time training loss is 23.18. After running the 92th epoch model achived this accuracy Table-VI represents the training accuracy of Avarage Pooling Operation. Fig.9 shows the graphical representation of the training accuracy and validation accuracy and also shows training loss and validation loss by applying Avarage Pooling operation.

TABLE VI: Epochs Outcomes Applying Average Pooling Operation

Epoch	Training Loss	Training Accuracy
01	3068.28	65.10%
02	902.78	79.46%
04	457.39	86.96%
11	154.28	92.09%
45	32.47	97.22%
53	10.73	98.51%
92	23.18	98.71%



(a) Model Accuracy Graph



(b) Model Loss Graph

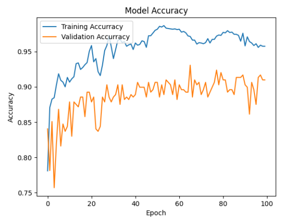
Fig. 9: By using the Custom Average Pooling Operation, a graphical representation accuracy of training and validation, along with training loss and validation loss.

F. *VGG16 Model Result Analysis*

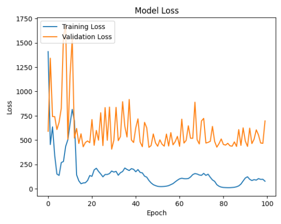
VGG16 model achieved 98.66% training accuracy on that time training loss is 24.45. After running the 54th epoch this model achieved the highest accuracy. After running the 54th epoch accuracy doesn't improve from 98.66% The training accuracy and loss of the VGG16 Model are shown in Table-VII for each epoch. Fig.10 shows the graphical representation of the training accuracy and validation accuracy and also shows training loss and validation loss by applying VGG16 Model.

TABLE VII: Epochs Outcomes Applying VGG16 Model

Epoch	Training Loss	Training Accuracy
01	1409.71	78.08%
02	454.36	87.06%
15	81.82	93.37%
29	156.25	97.17%
49	42.39	97.60%
51	25.36	98.15%
54	24.45	98.66%



(a) VGG16 Model Accuracy Graph

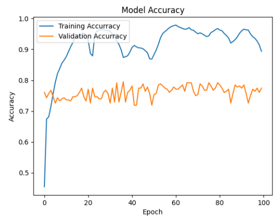


(b) VGG16 Model Loss Graph

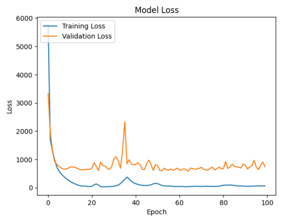
Fig. 10: By using the VGG16 Model, a graphical representation accuracy of training and validation, along with training loss and validation loss.

G. *Xception Model Result Analysis*

Xception model achieved 97.89% training accuracy on that time training loss is 41.14. After running the 61th epoch this model achieved the highest accuracy. Fig.11 shows the graphical representation of the training accuracy and validation accuracy and also shows training loss and validation loss by applying Xception Model.



(a) Xception Model Accuracy Graph

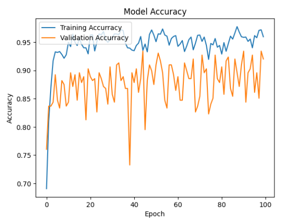


(b) Xception Model Loss Graph

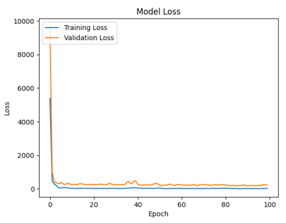
Fig. 11: By using the Xception Model, a graphical representation accuracy of training and validation, along with training loss and validation loss.

H. *InceptionV3 Model Result Analysis*

InceptionV3 model achieved 97.41% training accuracy on that time training loss is 11.22. After running the 88th epoch this model achieved the highest accuracy. Fig.12 shows the graphical representation of the training accuracy and validation accuracy and also shows training loss and validation loss by applying InceptionV3 Model.



(a) InceptionV3 Model Accuracy Graph

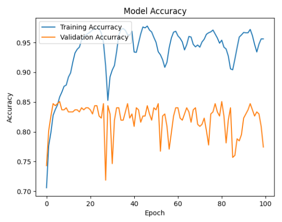


(b) InceptionV3 Model Loss Graph

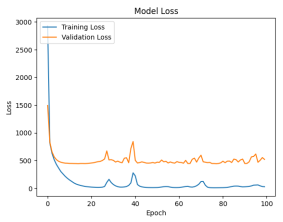
Fig. 12: By using the InceptionV3 Model, a graphical representation accuracy of training and validation, along with training loss and validation loss.

I. DenseNet201 Model Result Analysis

DenseNet201 model achieved 97.74% training accuracy on that time training loss is 14.36. After running the 47th epoch this model achieved the highest accuracy. Fig.13 shows the graphical representation of the training accuracy and validation accuracy and also shows training loss and validation loss by applying DenseNet201 Model.



(a) DenseNet201 Model Accuracy Graph

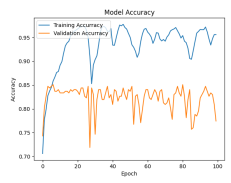


(b) DenseNet201 Model Loss Graph

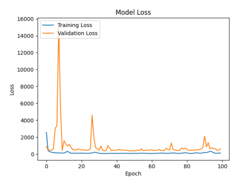
Fig. 13: By using the DenseNet201 Model, a graphical representation accuracy of training and validation, along with training loss and validation loss.

J. InceptionResNetV2 Model Result Analysis

InceptionResNetV2 model achieved 96.25% training accuracy on that time training loss is 46.13. After running the 48th epoch this model achieved the highest accuracy. Fig.14 shows the graphical representation of the training accuracy and validation accuracy and also shows training loss and validation loss by applying InceptionResNetV2 Model.



(a) Model Accuracy Graph



(b) Model Loss Graph

Fig. 14: By using the InceptionResNetV2 Model, a graphical representation accuracy of training and validation, along with training loss and validation loss.

After developing our model we check the performance of this proposed model on the training data set.Fig. 15 to show the number plate recognition form image.

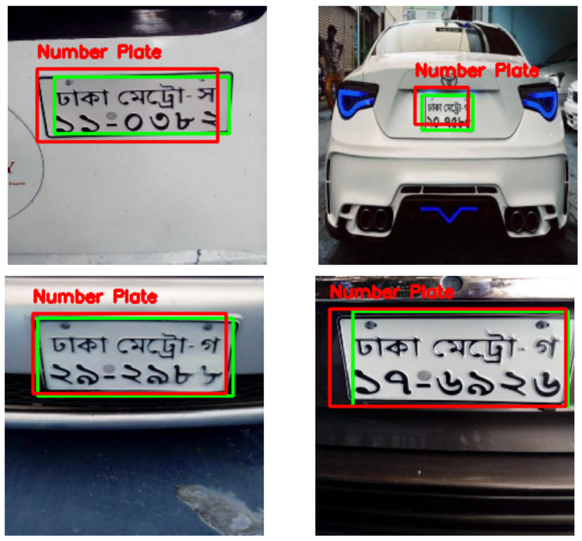


Fig. 15: Number Plate Detection Form Image.

Table-VIII Comparison across every model that has been used.

TABLE VIII: Comparison within all the applied models.

Model	Epoch	Training Loss	Training Accuracy
Custom Max Pool	47	12.62	99.53%
ResNet50	88	10.22	99.33%
MobileNetV2	81	8.42	99.03%
Custom Average Pool	92	23.18	98.71%
VGG16	54	24.45	98.66%
Xception	61	41.14	97.89%
InceptionV3	88	11.22	97.41%
DenseNet201	47	14.36	97.74%
InceptionResNetV2	48	46.13	97.25%

V. CONCLUSION AND FUTURE WORK

In this research Bangla number plate recognition system using Convolutional Neural Networks (CNNs) has shown impressive results in recognizing Bangla number plates. The best-performing model was the custom Max Pooling operation also CNN pre-trained model ResNet50. Also better performance on other models. Future research should focus on improving the accuracy of unseen data. Future research could focus on developing more robust recognition systems that can handle these challenges. In future research, we also focus on classifying the vehicle using numbers.

REFERENCES

[1] R. Ahmed and K. H. Mahmud, “Potentiality of high-resolution topographic survey using unmanned aerial vehicle in bangladesh,” *Remote Sensing Applications: Society and Environment*, vol. 26, p. 100729, 2022.

[2] A. Sufian, A. Ghosh, A. Naskar, F. Sultana, J. Sil, and M. H. Rahman, “Bdnet: bengali handwritten numeral digit recognition based on densely connected convolutional neural networks,” *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 2610–2620, 2022.

[3] M. J. Hossain, M. H. Uzzaman, and A. Saif, "Bangla digital number plate recognition using template matching for higher accuracy and less time complexity," *International Journal of Computer Applications*, vol. 975, p. 8887, 2018.

[4] A. Ashrafee, A. M. Khan, M. S. Irbaz, A. Nasim, and M. Abdullah, "Real-time bangla license plate recognition system for low resource video-based applications," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 479–488, 2022.

[5] M. A. Islam, G. M. Chowdhury, and M. N. M. Haque, "Bangla license plate detection, recognition and authentication with morphological process and template matching," in *2021 2nd International Conference for Emerging Technology (INCET)*, pp. 1–6, IEEE, 2021.

[6] M. N. C. Tadas, "Review on detection and recoginition of car number plate using particle swarm optimization algorithm,"

[7] A. Badr, M. M. Abdelwahab, A. M. Thabet, and A. M. Abdelsadek, "Automatic number plate recognition system," *Annals of the University of Craiova-Mathematics and Computer Science Series*, vol. 38, no. 1, pp. 62–71, 2011.

[8] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, pp. 1–6, Ieee, 2017.

[9] P. Mishra, A. Biancolillo, J. M. Roger, F. Marini, and D. N. Rutledge, "New data preprocessing trends based on ensemble of multiple preprocessing techniques," *TrAC Trends in Analytical Chemistry*, vol. 132, p. 116045, 2020.

[10] F. D. Adhinata, D. P. Rakhmadani, M. Wibowo, and A. Jayadi, "A deep learning using densenet201 to detect masked or non-masked face," *JUITA: Jurnal Informatika*, vol. 9, no. 1, pp. 115–121, 2021.

[11] R. Roslidar, K. Saddami, F. Arnia, M. Syukri, and K. Munadi, "A study of fine-tuning cnn models based on thermal imaging for breast cancer classification," in *2019 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, pp. 77–81, IEEE, 2019.

[12] M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud, and J.-H. Kim, "An automated system to limit covid-19 using facial mask detection in smart city network," in *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pp. 1–5, IEEE, 2020.

[13] S. Chakraborty, F. J. M. Shamrat, M. M. Billah, M. Al Jubair, M. Alauddin, and R. Ranjan, "Implementation of deep learning methods to identify rotten fruits," in *2021 5th international conference on trends in electronics and informatics (ICOEI)*, pp. 1207–1212, IEEE, 2021.

[14] S. N. Endah, I. N. Shiddiq, *et al.*, "Xception architecture transfer learning for garbage classification," in *2020 4th International Conference on Informatics and Computational Sciences (ICICoS)*, pp. 1–4, IEEE, 2020.

[15] A. Demir, F. Yilmaz, and O. Kose, "Early detection of skin cancer using deep learning architectures: resnet-101 and inception-v3," in *2019 medical technologies congress (TIPTEKNO)*, pp. 1–4, IEEE, 2019.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[18] B. Mandal, A. Okeukwu, and Y. Theis, "Masked face recognition using resnet-50," *arXiv preprint arXiv:2104.08997*, 2021.