International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 07 | July 2023 www.irjet.net p-ISSN: 2395-0072

An Enhancement of Braille Character Perception Using Deep Learning and Artificial Intelligence Techniques

Ms. A.Varshini [1], Dr.S. Gopinathan[2]

[1] M.Phil Research Scholar, Department of Computer Science, University of Madras, Tamil Nadu, India.
[2] Professor, Department of Computer Science, University of Madras, Tamil Nadu, India.

---***---

Abstract - *India is the largest popular country in the world. Due to shortage of vitamins, by birth 1% of the children are born with blindness. Blind people start their career with Braille. Braille makes reading and writing possible for visually impaired people. Braille-to-text translation serves as a bridge between the two worlds. The communication process has been simplified by advances in technology. This research article describes how to convert Braille to English using deep learning. In this work, 26 English Braille Images are used as a dataset after the segmentation process. The proposed method converts braille visuals to English text using convolutional neural network CNN models such as LeNet, VGG-16, DenseNet121, ResNet50, and Inceptionv3. The implemented system Inceptionv3 achieves a high level of prediction accuracy of 92%. Experiments show that the proposed Braille character recognition gives an accurate result.*

***Key Words*: Braille, Deep Learning, Convolutional Neural Network, LeNet, VGG-16, DenseNet121, ResNet50, and Inceptionv3.**

1. INTRODUCTION

Over 2.2 billion people worldwide are blind or visually impaired, according to the World Health Organization (WHO) [1]. The people having impairments of vision utilize Braille, a system of pointed dots that may be read by finger contact, as it is difficult for them to read and write text [2]. A Braille cell or block is made up of six dots that are organized in two columns and three rows. Each dot within the cell can be either raised or not raised, representing different combinations of dots that correspond to specific characters. Counting starts at the top and goes down to the bottom.

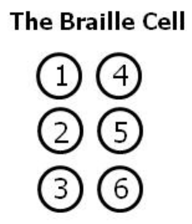


Fig. 1. Braille Cell

By combining different combinations of raised dots within the cell, all the letters of the alphabet, numbers, punctuation marks, and even special symbols can be represented in Braille.

Understanding the patterns of raised dots and their corresponding characters is fundamental in recognizing and interpreting Braille text accurately.

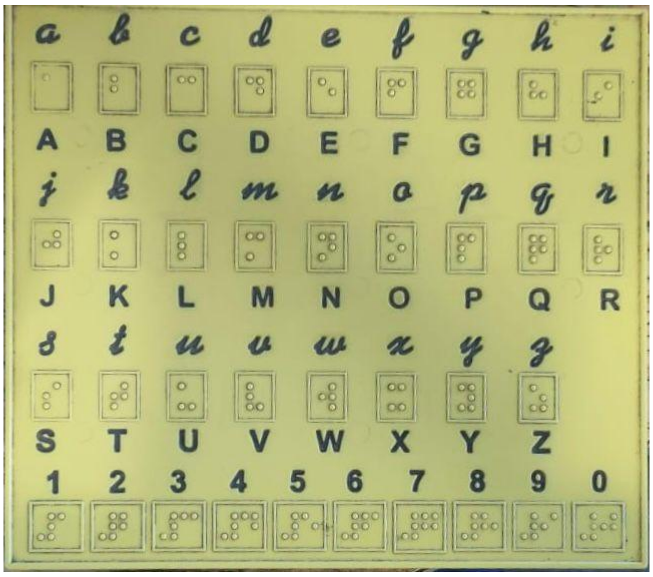


Fig. 2. Braille Alphabet Board

1.1 Challenges

In Braille images, it is challenging to preprocess the images to remove all the noise that is introduced. After all the braille cells have been recognized, they need to be translated into the appropriate English alphabet. When a single alphabet is represented by recognized braille cells, there is a chance that they might have significant variations.

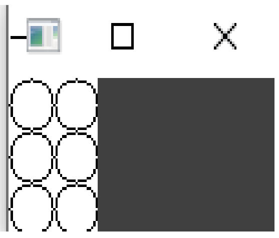


Fig. 3. Empty Circle Braille Dot

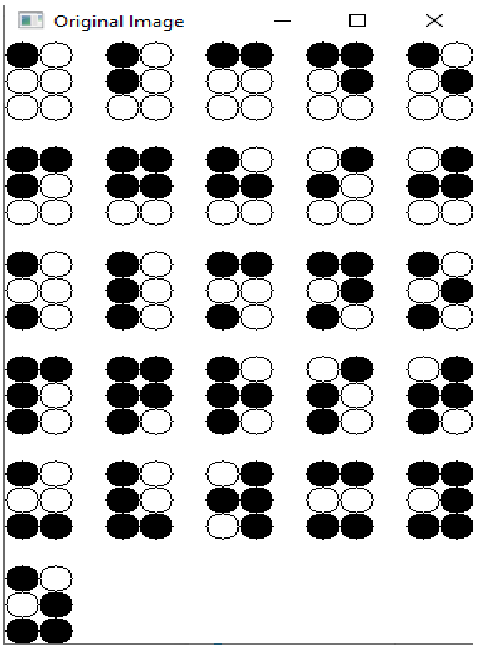


Fig. 4. Braille Dot Alphabet Created

In figure 5, the recognized braille cells are likely to contain large variations representing the same alphabet.

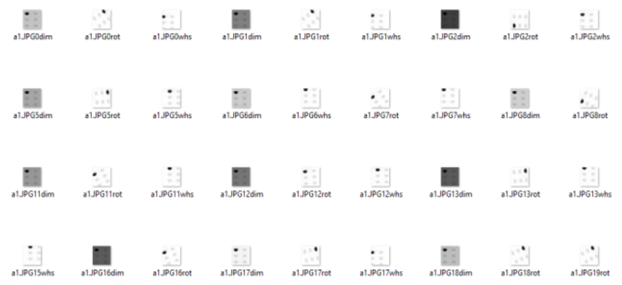


Fig 5. Sample of Braille Alphabet 'A'

1.2 Aim Research:

To improve the accuracy of Braille text recognition, it is essential to create a dataset of Braille characters and compare the collected data to identify matching patterns. By carefully analyzing the dataset and comparing it to the input Braille text, that enhance a higher level of precision can be achieved in the recognition process. This approach enables accurate interpretation and translation of Braille characters into readable text.

1.3. Proposed Model:

The suggested deep CNN model is carefully presented, and several approaches are addressed. Convolutional neural networks (CNNs) are an artificial neural network (ANN) type that are frequently used for pattern recognition applications. Deep CNN primarily consists of alternative, convolution, max-pooling, and fully connected layers that learn high level information from low level representations [4].

In this study, we constructed a deep CNN model with two key layers: feature extraction and classification. For feature extraction, the proposed deep CNN networks selected are LeNet, VGG-16, DenseNet121, ResNet50, and Inceptionv3.

Dataset:

The standard dataset "Braille Character Dataset" is downloaded to the local directory "C:BrailleDATASET" from the Kaggle website in order to be used for training and testing the CNN model. According to the English alphabet from "A" to "Z," the Braille character dataset was separated into 26 groups. Each category has its own folder. The model takes into account 1560 photos altogether, divided by 26 alphabet folders and 3 unique folders. So, each Braille cell may be recognized in 60 different ways using our trained model. Fig. 6 displays the Sample dataset for category (Directory) 'A', which consists of 60 sets.

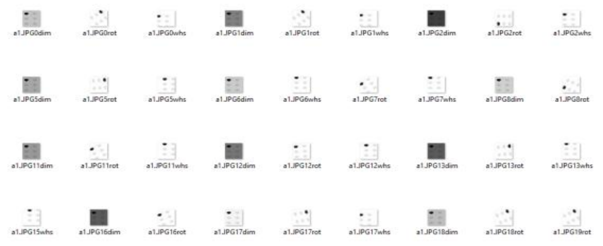


Fig 6. The alphabet class "A" folder contains 60 examples of graphics.

1.4. Proposed model of Algorithm

Python-3 was the programming language used to create the model, while the earliest version of Jupyter notebook utilized was 6.4.11. Additionally utilized are NumPy, glob, random, Matplotlib, TensorFlow, and Keras, among other tools and libraries. Below is the method for the Braille classifier model.

1. Save the Braille dataset to a local directory using Kaggle.

2. CNN model architecture for import transfer learning

3. Image preparation.

4. Feature for training, testing and validation and classification layers.

5. Add Fully connected layers with softmax and relu activation function

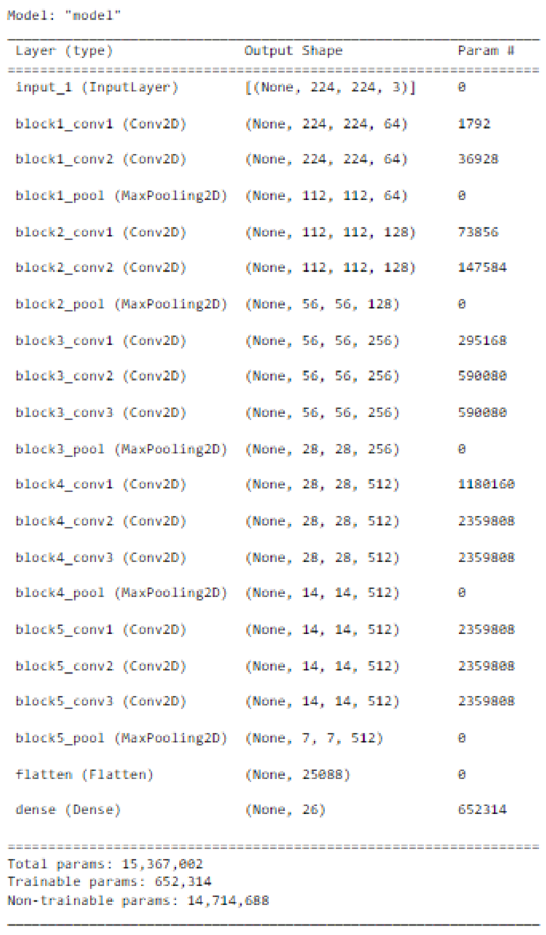
6. Compile and fit the model

7. Assess the model.

Step 1: Create the training, validation, and testing directories in the first stage. Each directory now has 26 directories, each representing 26 classes and 26 alphabets. There are 60 identical photos in each directory. Figure 6 displays the 60 photos in directory 'A' that are part of class 'A'.

Step-2: Equation 1 is used to import the five CNN architectures needed to categorize the multiclass picture. (input_shape="IMAGE_SIZE,""imagenet"weights, "include_top" False)

The VGG-16 model comprises a total of 16 convolutional and fully connected layers, according to a summary of the model's layers. is depicted in fig 7.



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 26)                652314

=================================================================
Total params: 15,367,002
Trainable params: 652,314
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure-7: VGG16 Model overview

Step-3: Before being fed into the model, each picture is transformed to a negative image using the cv2 package.

Figure 8 displays both the original image and a sample output negative image.

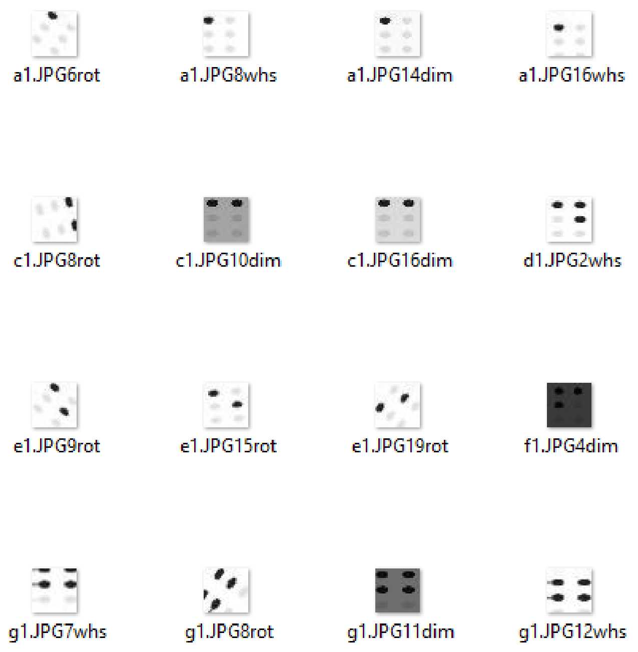


Figure :8 (a) Original Image

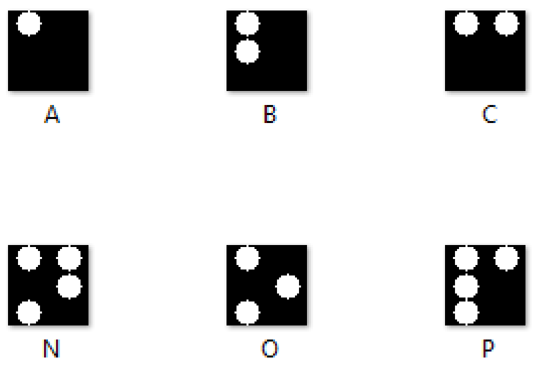
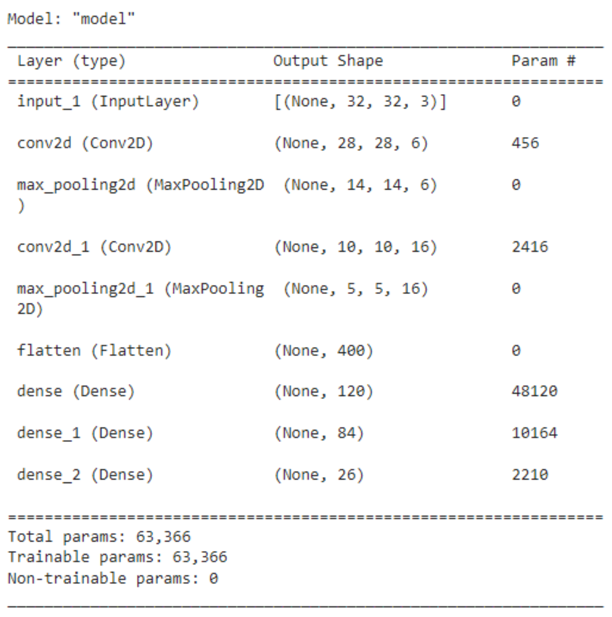


Figure: 8 (b) Negative image

It is discovered that the original picture size is 28x28. 100x100 is the new image's new foundation size for our model. The photos are shown in a figure-8 before and after scaling.

Step-4: In step 4, the features from the training, test, and validation dataset are extracted using the predict_generator () function and converted to a NumPy array as indicated in equation 2. Dense(len(folders), activation='softmax')(x) = prediction - (2)

Step 5: The final fully linked layer is given a convolution foundation at this point. Initialization of the sequential model comes first. Flatten layer, Dense1 layer, Dropout layer, Dense2 layer, and another Dropout layer are all fully linked layers that are inserted as the convolution foundation. In these layers, the ReLu activation function was used. The SoftMax activation Dense layer, which has 26 classes, is inserted as the final output layer. Figure 9 displays the summary of the Convolution model.



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 3)]       0

 conv2d (Conv2D)             (None, 28, 28, 6)         456

 max_pooling2d (MaxPooling2D  (None, 14, 14, 6)        0
 )

 conv2d_1 (Conv2D)           (None, 10, 10, 16)        2416

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 16)         0
 2D)

 flatten (Flatten)           (None, 400)               0

 dense (Dense)               (None, 120)               48120

 dense_1 (Dense)             (None, 84)                10164

 dense_2 (Dense)             (None, 26)                2210

=================================================================
Total params: 63,366
Trainable params: 63,366
Non-trainable params: 0
_________________________________________________________________
```

Fig -9: Braille Convolution model Summary

Step6: Step 6 involves compiling the model with 10 epochs and 32 batch sizes. It is important to point out that the system's accuracy increases with each training dataset run.

2. Methodology:

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper.

2.1 Preprocessing

The two primary steps of this Braille character recognition system are described below. Using image preprocessing techniques, picture alignment and enhancement are carried out in the first stage. The suggested convolution neural network is then used to recognize characters in the second stage. The Braille character recognition flow diagram.

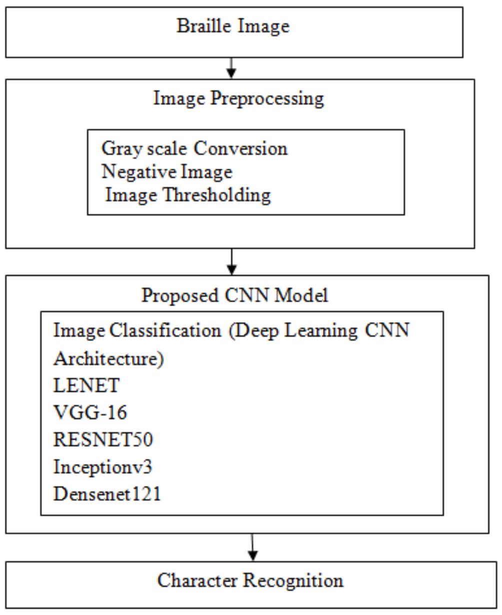


Figure: 10 Braille character recognition procedure flowchart.

2.1 Proposed Image Classification Method

LENET : One of the first pre-trained models was LeNet-5[7]. Recognizing the handwritten and machine printed characters. LeNet model can achieve a test accuracy of 78.0% . Shown in Figure 13.

Classification Layers :

Convolutional Layers: The first Conv2D layer with 6 filters and a kernel size of (5, 5) applies convolutional operations to the input images, extracting various features through the use of learned filters.

The first MaxPooling2D layer with a pool size of (2, 2) performs down-sampling, reducing the spatial dimensions of the feature maps and retaining the most prominent features. The second Conv2D layer with 16 filters and a kernel size of (5, 5) further extracts higher-level features from the previously obtained feature maps.

The second MaxPooling2D layer with a pool size of (2, 2) performs additional down-sampling, further reducing the spatial dimensions and capturing the most important features.

Fully Connected Layers:

Fully Connected Layers: Flatten layer to flatten the Output. Dense layer with 120 units and ReLU activation. An 84-unit dense layer activated by ReLU. Dense layer with num_ class units (softmax activation).

VGG-16:VGG stands for Visual Geometry Group [6].
VGG-16 was used for creating the model and to extract features from the images [10]. VGG-16 model can achieve a test accuracy of 89.9% in ImageNet. Shown in Fig 14.

Key features of the VGG-16 architecture:
Input Layer: The input to the VGG-16 network is an RGB image of size 224x224 pixels.

VGG-16 Base Model:
The VGG-16 base model is loaded using the VGG16 class from Keras with pre-trained weights from the ImageNet dataset. The input_shape parameter specifies the input image size (224x224 pixels) and the number of color channels (3 for RGB). The include_top parameter is set to False to exclude the fully connected layers at the top of the VGG-16 architecture.

Freezing Layers: The code freezes the weights of all layers in the VGG-16 model to prevent them from being updated during training. The for loop iterates through all layers in the VGG-16 model and sets layer.trainable = False. Output Layer: A Dense layer with a softmax activation function is added as the output layer.

INCEPTIONv3: Inception is also much lower than VGG or its higher performing successors. These methods could be applied to optimize the Inception architecture as well, widening the efficiency gap again [5].The InceptionV3 base model is loaded with pre-trained weights from the ImageNet dataset. Inception module that reduces the grid-size while expands the filter banks[13]. The base model consists of multiple layers, including convolutional layers, pooling layers, and inception modules. Each Inception module consists of parallel branches of convolutional layers of varying filter sizes (1x1, 3x3, 5x5) and a max pooling layer. To reduce computational complexity, InceptionV3 employs factorization techniques such as using 1x1 convolutions to reduce the number of input channels before applying larger filters.Inceptionv3 model can achieve a test accuracy of 92.8%.show in Fig 15

This factorization helps in reducing the number of parameters and improves computational efficiency.

This allows us to use the convolutional layers and inception modules as a feature extractor by taking the output from these layers and feeding it into custom classification layers.

DENSENET-121:
DenseNet 121 stands for Densely Connected Convolutional Network-121 [9]. DenseNet121 model can achieve a test accuracy of 91.1% shown in Fig 16

DenseNet-121 promotes feature reuse by allowing information to flow directly from earlier layers to later layers. This reduces the vanishing gradient problem and encourages better gradient flow during training. Multipath-DenseNet is capable of more accurate predictions with fewer parameters[12].
Gradient Propagation: The dense connections facilitate gradient propagation by providing shorter paths for gradients to flow through the network during backpropagation.

The gradients computed during backpropagation become extremely small as they propagate from the higher layers to the lower layers of the network. As a result, the weights in the lower layers receive minimal updates, which leads to slow or no learning in those layers.

The DenseNet-121 architecture consists of multiple dense blocks, transition layers, and a final classification layer.

Each layer within a dense block typically consists of a batch normalization layer, a ReLU activation, and a 3x3 convolutional layer. Layers of transition include batch normalization, 1x1 convolution, and average pooling. The DenseNet121 model with pre-trained ImageNet weights, excluding the top (classification) layer. The model is stored in the variable DenseNet.

The fully connected layers are defined as: x = Flatten() (DenseNet. Output).

The Flatten layer is used to flatten the output from the previous layer into a 1-dimensional vector. This flattening operation is necessary to connect the fully connected layers.
RESNET50: Resnet stands for Residual Network-50 [8].ResNet-50 is a deep network architecture consisting of 50 layers. ResNet eases the optimization by providing faster convergence at the early stage [11]. It includes multiple residual blocks with different numbers of filters in each block. The classification layer used is a Dense layer with a softmax activation function. the final classification based on the extracted features from the ResNet50 base model. Prediction = Dense(len(folders), activation='softmax')(x)

The len(folders) represents the number of classes, and the activation='softmax' specifies the softmax activation function. The softmax function calculates the probabilities for each class, allowing the model to output the class probabilities for multi-class classification.

The class with the highest probability is considered the final predicted class label for a given input image. ResNet50 model can achieve a test accuracy of 25%

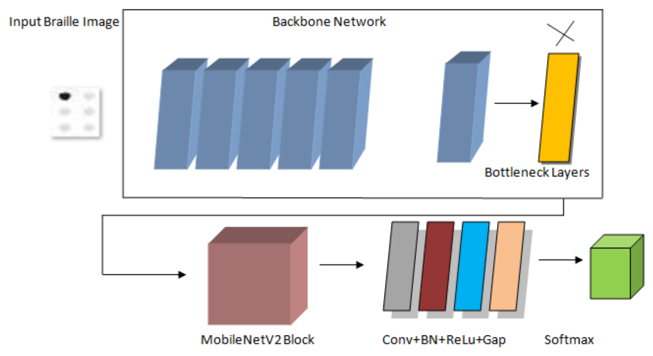


Figure: 11-The network architecture of proposed deep CNN model bottleneck layers in backbone network are replaced with MobileNetV2 Block.

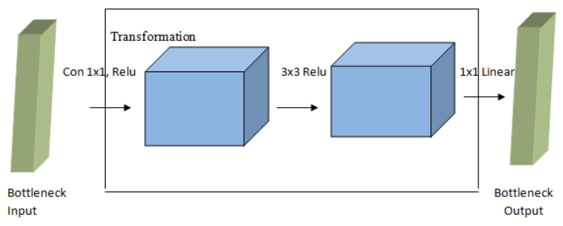
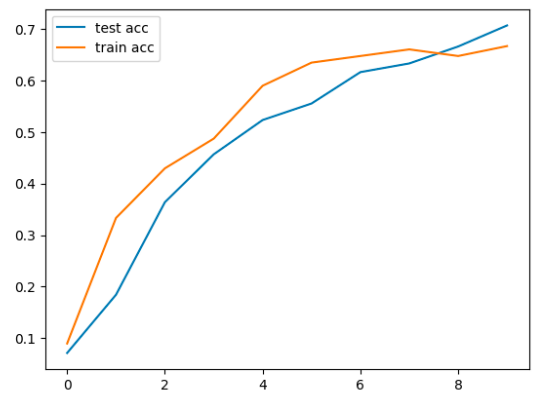


Figure: 12- Block Diagram for MobileNetV2

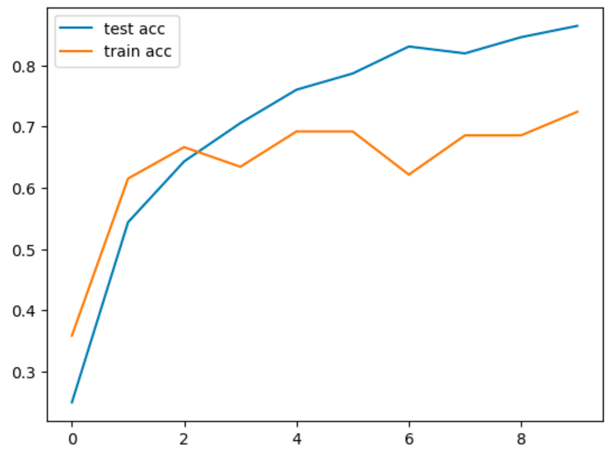
3. Experimental Results:

The dataset is downloaded from the Kaggle website and saved in the local directory. The English alphabets from "A" to "Z" were searched for using 26 folders from the collection of all alphabets. Each folder has a series of 60 photos that are variants of the same alphabet. 1560 images total—26 x 60—were taken into consideration for the model. The mentioned model was ran through 10 epochs. The figures below indicate the model's output accuracy.



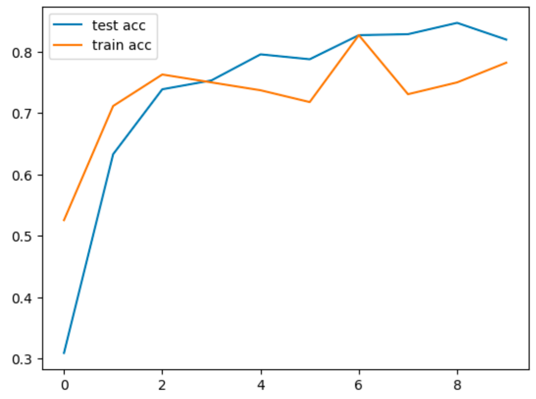
```
Test Accuracy: 0.7804487347602844
Train Accuracy: 0.6666666865348816
```

Figure 13 : Test Accuracy LeNet CNN Model



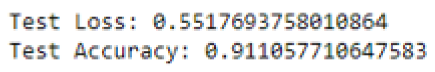
```
Test Accuracy: 0.8998397588729858
Train Accuracy: 0.7243589758872986
```

Figure 14: VGG-16 CNN Model Test Accuracy



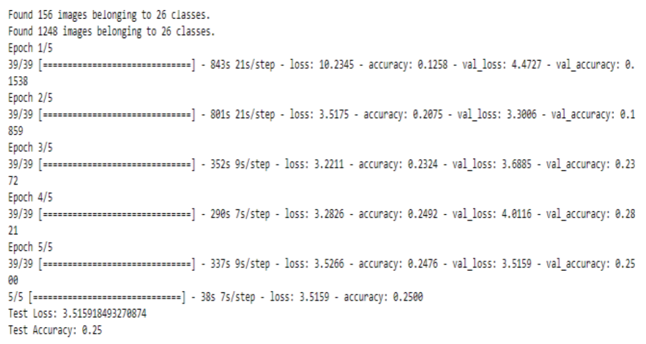
```
Test Accuracy: 0.9286859035491943
Train Accuracy: 0.7820512652397156
```

Figure 15: Inceptionv3 CNN Model Test Accuracy



```
Test Loss: 0.5517693758010864
Test Accuracy: 0.911057710647583
```

Figure 16 : DenseNet121 CNN Model Test Accuracy



```
Found 156 images belonging to 26 classes.
Found 1248 images belonging to 26 classes.
Epoch 1/5
39/39 [==============================] - 843s 21s/step - loss: 10.2345 - accuracy: 0.1258 - val_loss: 4.4727 - val_accuracy: 0.
1538
Epoch 2/5
39/39 [==============================] - 801s 21s/step - loss: 3.5175 - accuracy: 0.2075 - val_loss: 3.3006 - val_accuracy: 0.1
859
Epoch 3/5
39/39 [==============================] - 352s 9s/step - loss: 3.2211 - accuracy: 0.2324 - val_loss: 3.6885 - val_accuracy: 0.23
72
Epoch 4/5
39/39 [==============================] - 290s 7s/step - loss: 3.2826 - accuracy: 0.2492 - val_loss: 4.0116 - val_accuracy: 0.28
21
Epoch 5/5
39/39 [==============================] - 337s 9s/step - loss: 3.5266 - accuracy: 0.2476 - val_loss: 3.5159 - val_accuracy: 0.25
00
5/5 [==============================] - 38s 7s/step - loss: 3.5159 - accuracy: 0.2500
Test Loss: 3.515918493270874
Test Accuracy: 0.25
```

Figure:17- ResNet50 CNN Model Test Accuracy

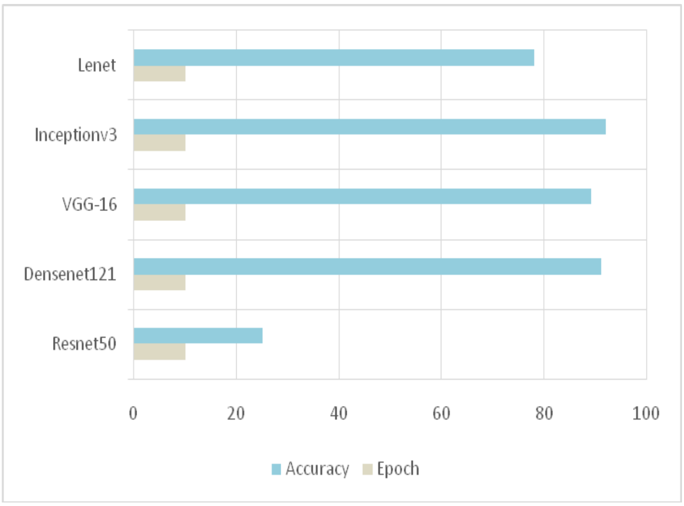


Figure:18- Comparison of epoch and accuracy

4. CONCLUSIONS

This study examines the braille identification accuracy for five CNNs with various depths and architectures. The input images are preprocessed and segmented before the train and test phase. The dataset has increased the accuracy also gets increased. The proposed system delivers that Densenet121 and Inceptionv3 show the highest accuracy of 91.1% and 92.8%.

REFERENCES

[1] T. Kausar, S. Manzoor, A. Kausar, Y. Lu, M. Wasif and M. A. Ashraf, "Deep Learning Strategy for Braille Character Recognition," in IEEE Access, vol. 9, pp. 169357-169371, 2021, doi: 10.1109/ACCESS.2021.3138240.

[2] A. Mousa, H. Hiary, R. Alomari and L. Alnemer, "Smart Braille system recognizer", Int. J. Comput. Sci. Issues, vol. 10, no. 6, pp. 52, 2013.

[3] Murthy, Vishwanath Venkatesh; Hanumanthappa, M.; Ruiz Mendoza, Juan Carlos, "VGG16 CNN based Braille Cell classifier model for Translation of Braille to Text" Special Education . 2022, Vol. 1 Issue 43, p4388-4396. 9p.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer sVision," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 2818-2826, doi: 10.1109/CVPR.2016.308.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Tech. Rep., 2015.

[7] Changjian Li and Weiqi Yan, "Braille Recognition Using Deep Learning", ICCCV '21: Proceedings of the 4th International Conference on Control and Computer Vision August 2021 https://doi.org/10.1145/3484274.3484280.

[8] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[9] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

[10] Shazzadul Islam and Sabit Ibn Ali Khan, "Bird Species Classification from an Image Using VGG-16 Network", ICCCM '19: Proceedings of the 7th International Conference on Computer and Communications Management July 2019 https://doi.org/10.1145/3348445.3348480.

[11] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[12] Hasan, N., Bao, Y., Shawon, A. et al. DenseNet Convolutional Neural Networks Application for Predicting COVID-19 Using CT Image. SN COMPUT. SCI. 2, 389 (2021). https://doi.org/10.1007/s42979-021-00782-7.

[13] Christian Szegedy Google Inc and Vincent Vanhoucke, "Rethinking the Inception Architecture for Computer Vision", https://doi.org/10.48550/arXiv.1512.00567

BIOGRAPHIES

A.Varshini, is a M.Phil. Research scholar at the Department of Computer Science, University of Madras. She received M.Sc., a degree from Bhaktavatsalam Memorial College for Women, Tamil Nadu, India. Her research area includes Digital Image Processing and Braille Character Perception.

Dr. S. Gopinathan is the Professor in the Department of Computer Science at the University of Madras. He has more than 26 years of experience in teaching.
He has published more than 70 articles in SCOPUS and SCI INDEXED journals.