COMPARISON OF DIFFERENT CONVOLUTIONAL NEURAL NETWORK ARCHITECTURES FOR THE RECOGNITION OF NUMBERS IN SIGN LANGUAGE

Aman Jain, Simran Sharma
Jaypee Institute of Information Technology
Noida, Uttar Pradesh.

Abstract – **In this paper, convolutional neural networks have been used for the recognition of numbers in sign language. The efficiency of recognition, highly depends on the choice of the neural network architecture. This paper, focuses on determining the best architecture for recognition of numbers in sign language. This has been done by conducting training experiments on "Sign Language Digits" Dataset and comparing of results using accuracy metrics.**

Keywords – Convolutional neural networks; architecture; accuracy; compare.

I. INTRODUCTION

Neural Networks is widely used in various applications. Convolutional Neural Networks is a class of deep learning most commonly applied to analysis of images.

Every dataset has a different accuracy for different architectures and choosing the best architecture is the primary hurdle for the programmer. The purpose of this paper is to solve this ambiguity.

This paper, compares different architectures for the recognition of numbers in sign language on the "Sign Language Digits" Dataset. The results from each architecture is compared and the one with highest accuracy is chosen as the best architecture.

This project not just solves ambiguity related to architecture but also helps the differently abled people who can't hear. There will be no need for hiring any person

II. LITERATURE

This work is a result of the inspiration from [4] which describes the comparison of different neural network architectures for solving the problem of emotion recognition by facial expression.

Vivek Bheda, et al. [3] discussed about a method for using deep convolutional networks to classify images of both the the letters and digits in American Sign Language, in this paper

Karan Chauhan, et al. [5], have deployed a deep learning convolutional neural network based on keras and tensorflow using python for binary image classification. In this study, a large number of different images, which contains two types of animals, namely cat and dog are used for image classification. Four different structures of CNN are compared on CPU system, with four different combinations of classifiers and activation functions.

Shivashankara S, et. al. [6], have presented an optimal approach to accomplish the transliteration of 24 static sign language alphabets and numbers of American Sign Language into understandable English manuscript. This paper also presents the statistical result evaluation with the comparative graphical depiction of existing techniques and proposed technique.

Chen Wang, et. al. [7], has proposed a new approach to do the convolution in convolutional neural network to solve tough image classification on CIFAR10 and made some experiments to test the functionality of dropout layer.

Mingyuan Xin, et. al. [8] has proposed an innovative training criterion of depth neural network for maximum interval minimum classification error based on the analysis of the error backpropagation algorithm.

Md. Abul Kalam, et. al. [9], have proposed a 10 layers convolutional neural network model using residual learning to detect digits in sign language of any angle.

Pierre Sermanet, et. al. [10], have augmented the traditional Convolution Neural Network architecture by learning multistage features and by using Lp-pooling on the SVHN dataset. Furthermore, they have analysed the benefits of different pooling methods and multi-stage features in CNN.

Dan Cireşan, et.al. [11] combined various Deep Neural Networks trained on differently pre-processed data into a "Multi-Column DNN" (MCDNN) which boosted recognition performance, making the system insensitive also to variations in contrast and illumination.

Alex, et. al. [12] trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. The neural network has 60 million parameters, 650,000 neurons and consists of five convolutional layers.

Jie Huang, et. al. [13] proposed a novel 3D convolutional neural network (CNN) to approach the problems due to large variations of hand gestures. It extracts discriminative spatial-temporal features from raw video stream automatically without any prior knowledge, avoiding designing features.

Li Deng, et. al. [14] have performed a research addressing on the excellence of the convolution structure in neural networks for classification performance on the famous, MNIST dataset.

III. PROPOSED APPROACH

The dataset "Sign Language Digits" is first split into training and testing data using the inbuilt sklearn library. The training data is augmented using an inbuilt keras library which works on width shift, height shift, zoom, shear and rotation of then images, with the objective to populate the dataset.

The next phase is to construct the three models, one-layer CNN, two-layer CNN and four-layer CNN, respectively. Each model contains the layers – 2D convolution layer, 2D Maxpooling layer, Dropout Layer, Flatten layer and Dense layer.

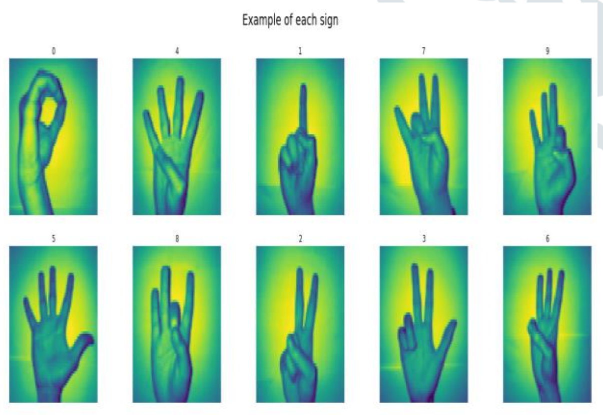
The data is trained on each model on the Google Collaboratory and is saved. The second part of the data, i.e., the testing data is tested on each model to predict the accuracy.

The results, i.e., the validation accuracy, training accuracy, validation loss and training loss are compared. Then, the comparison is represented using confusion matrix. The most accurate model is then tested with real life images, with prediction as output.

IV. DATASET

This project uses the dataset "Sign language digits dataset". It contains images of numbers in sign language.

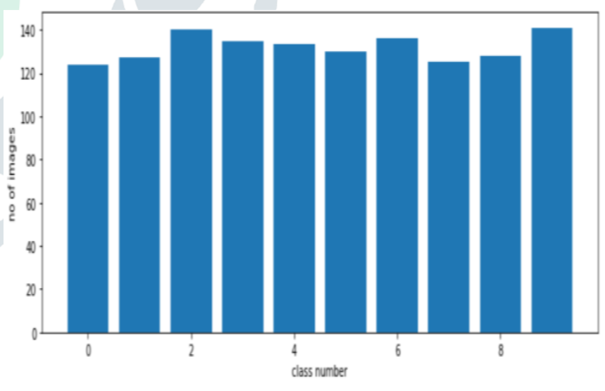
1. Ten images from test data:



2. Classes of images of data:

```
Counter({0: 124,
  1: 127,
  2: 140,
  3: 135,
  4: 133,
  5: 130,
  6: 136,
  7: 125,
  8: 128,
  9: 141})
```

3. Distribution of data in training dataset:



V. IMPLEMENTATION

A. *Image Pre-processing*

In image pre-processing, data augmentation is done on the training data using the keras library "ImageDegenerator". It generates batches of tensor image data with real time data augmentation. Width shift, height shift, zoom, shear and rotation are performed on the training data using this library.

B. *Training Models*

The data is trained in 10 epochs on "Google Collaboratory". The following layers are used in all three models:

- Conv2D
  It is the 2D Convolution Layer. It creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. It is used to extract features.

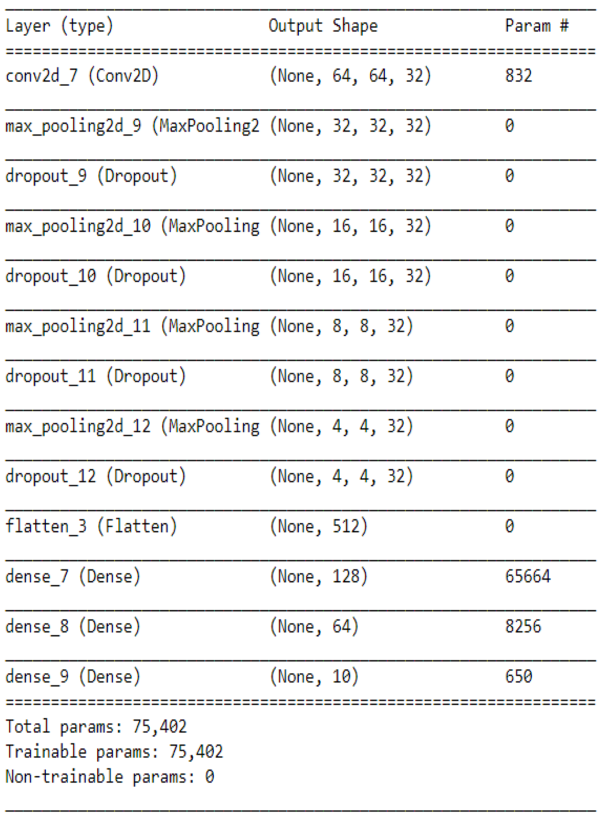
- MaxPooling2D
  Max pooling operation for spatial data. It identifies the most important feature out of the features extracted in the convolution layer.

- Dropout
  It is a regularisation technique for the CNN model. It is used for increasing the model's performance as it prevents overfitting.

- Flatten
  The convolution and pooling layers have multi-dimensional tensors as output. Flatten reduces them into a very long 1-D tensor, which is suitable for dense layer.

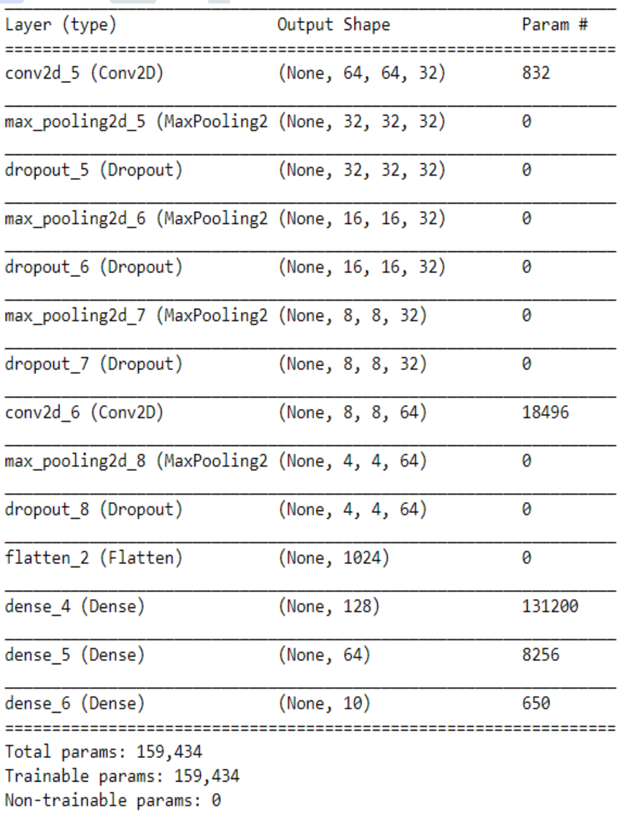
- Dense
  It is a densely connected layer of neurons. It implements the activation function. In the Four-layer model, three dense layers are implemented. The activation function "relu" is used in first two and "softmax" in the last layer.

1. One CNN Layer



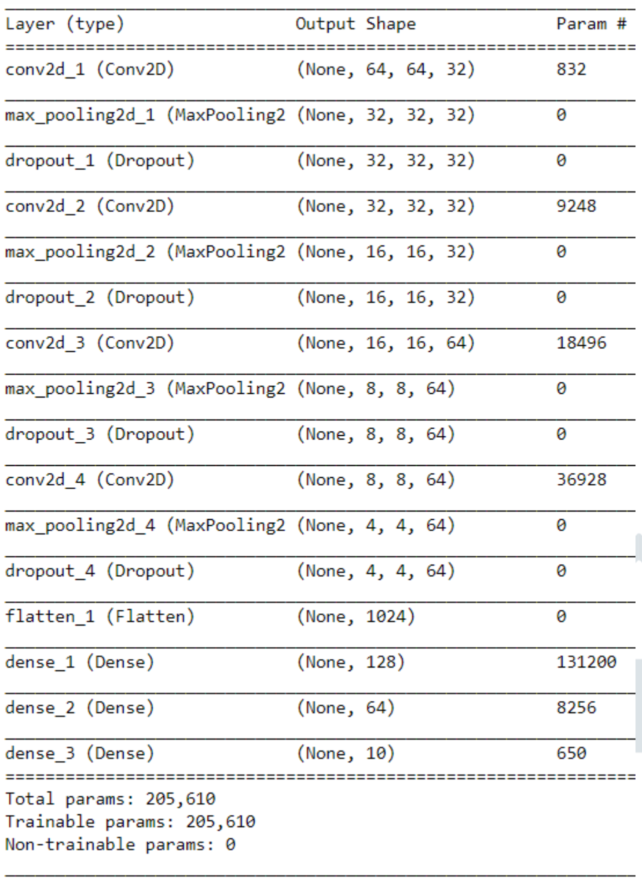
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 64, 64, 32)        832
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
dropout_9 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
dropout_10 (Dropout)         (None, 16, 16, 32)        0
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 8, 8, 32)          0
_________________________________________________________________
dropout_11 (Dropout)         (None, 8, 8, 32)          0
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 4, 4, 32)          0
_________________________________________________________________
dropout_12 (Dropout)         (None, 4, 4, 32)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_8 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_9 (Dense)              (None, 10)                650
=================================================================
Total params: 75,402
Trainable params: 75,402
Non-trainable params: 0
_________________________________________________________________
```

2. Two CNN Layers



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 64, 64, 32)        832
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
dropout_7 (Dropout)          (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 64)          18496
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               131200
_________________________________________________________________
dense_5 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_6 (Dense)              (None, 10)                650
=================================================================
Total params: 159,434
Trainable params: 159,434
Non-trainable params: 0
```

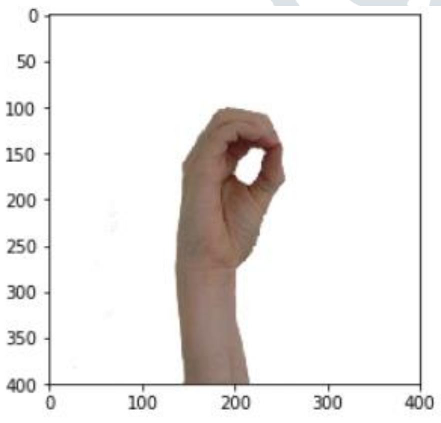
3. Four CNN Layers



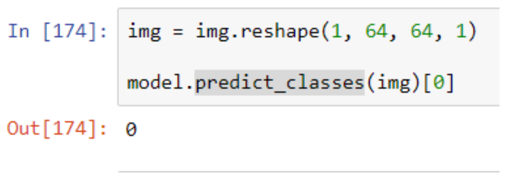
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 32)        832
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               131200
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 205,610
Trainable params: 205,610
Non-trainable params: 0
_________________________________________________________________
```

C. Testing

The dataset is split into training data and testing data using a model selection library of sklearn, "train_test_split". These models are not just tested on this testing data from dataset but also on real-life images.



The above image first undergoes pre-processing – grayscale, histogram equalization and normalisation and then is tested with the four-layer model, as it came out to be most efficient.

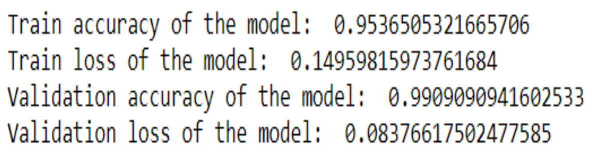


```
In [174]: img = img.reshape(1, 64, 64, 1)

          model.predict_classes(img)[0]

Out[174]: 0
```

VI. RESULTS

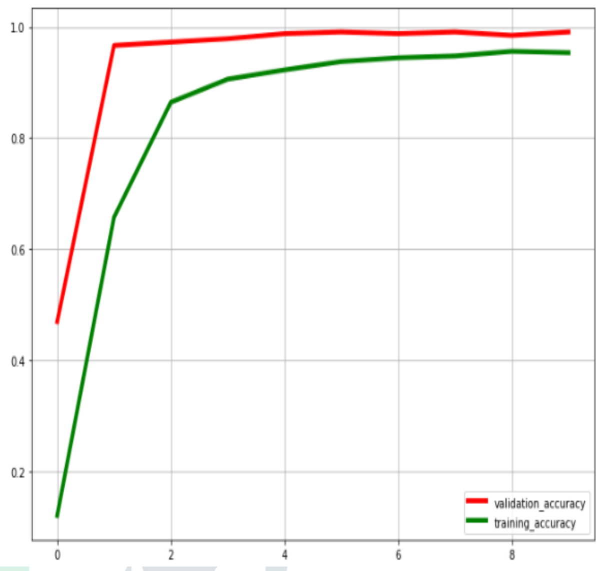
A. Four CNN layers

❖ This model has the maximum training accuracy as well as maximum validation accuracy:

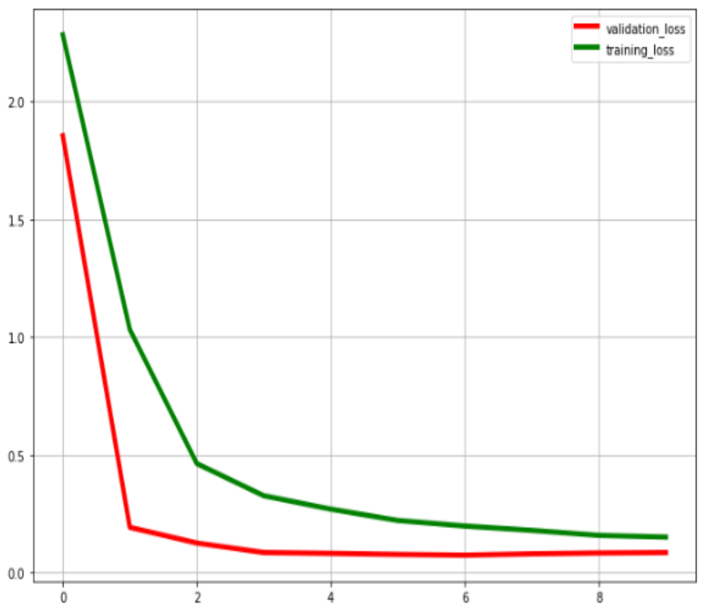


```
Train accuracy of the model:  0.9536505321665706
Train loss of the model:  0.14959815973761684
Validation accuracy of the model:  0.9909090941602533
Validation loss of the model:  0.08376617502477585
```

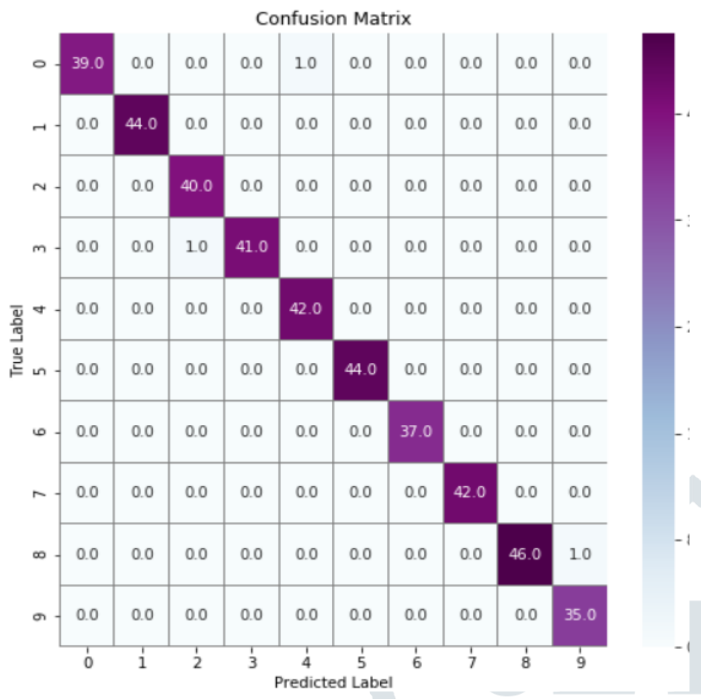
• Representation of valliation accuracy and training loss



• Representation of validation loss and training loss.



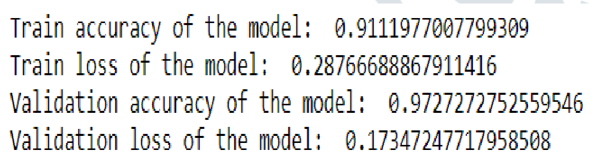
❖ Confusion matrix



B. Two CNN layers

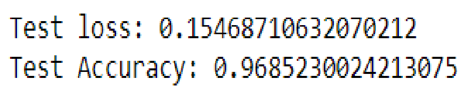
❖ Accuracy and loss:

For train:



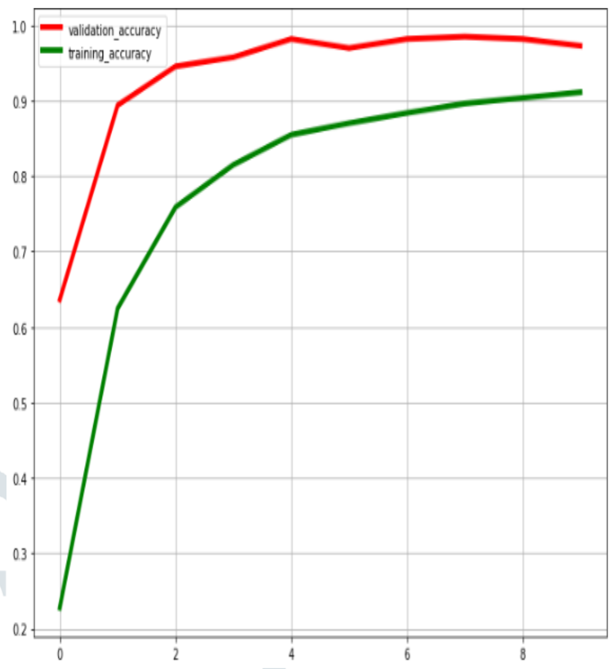
```
Train accuracy of the model:  0.9111977007799309
Train loss of the model:  0.28766688867911416
Validation accuracy of the model:  0.9727272752559546
Validation loss of the model:  0.17347247717958508
```

For test:

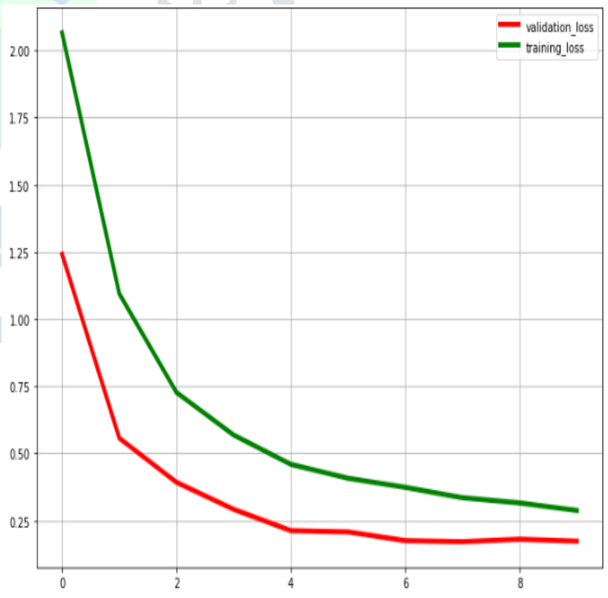


```
Test loss: 0.15468710632070212
Test Accuracy: 0.9685230024213075
```

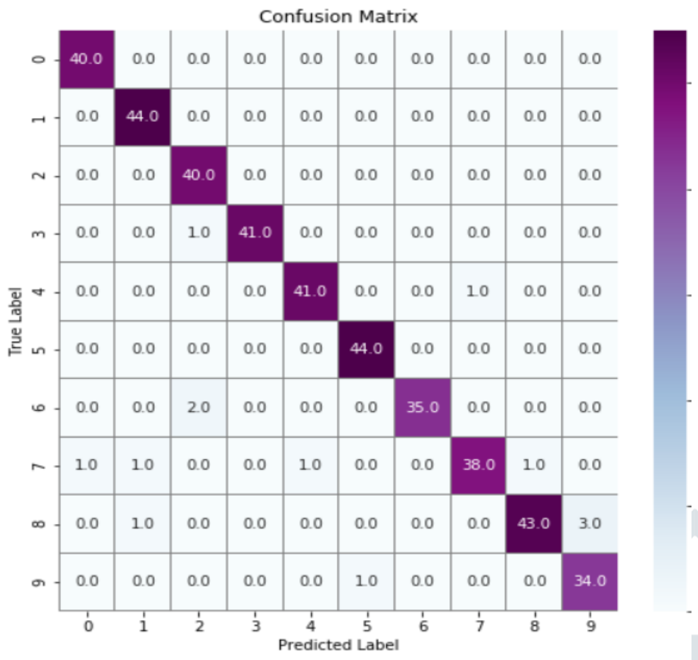
- Representation of validation accuracy and training loss:



- Representation of validation loss and training accuracy:



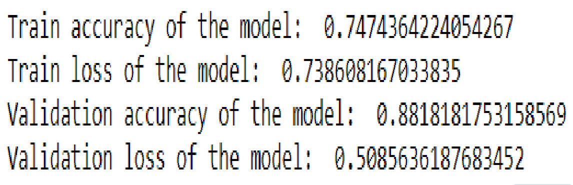
❖ Confusion Matrix:



C. One CNN layer

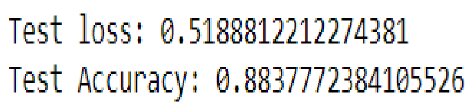
❖ Accuracy and loss:

For train:



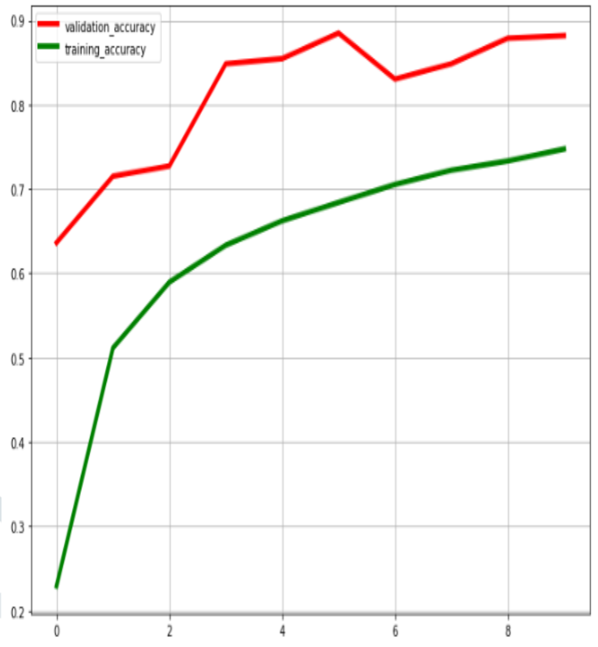
```
Train accuracy of the model:  0.7474364224054267
Train loss of the model:  0.738608167033835
Validation accuracy of the model:  0.8818181753158569
Validation loss of the model:  0.5085636187683452
```

For test:

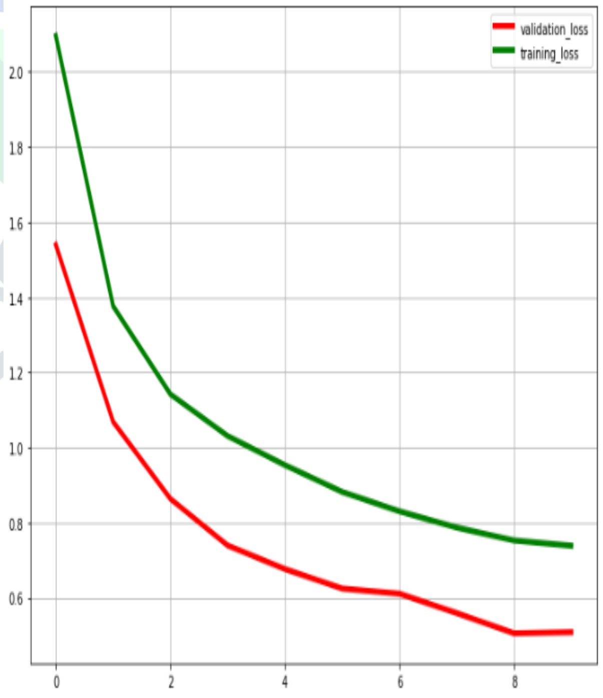


```
Test loss: 0.5188812212274381
Test Accuracy: 0.8837772384105526
```

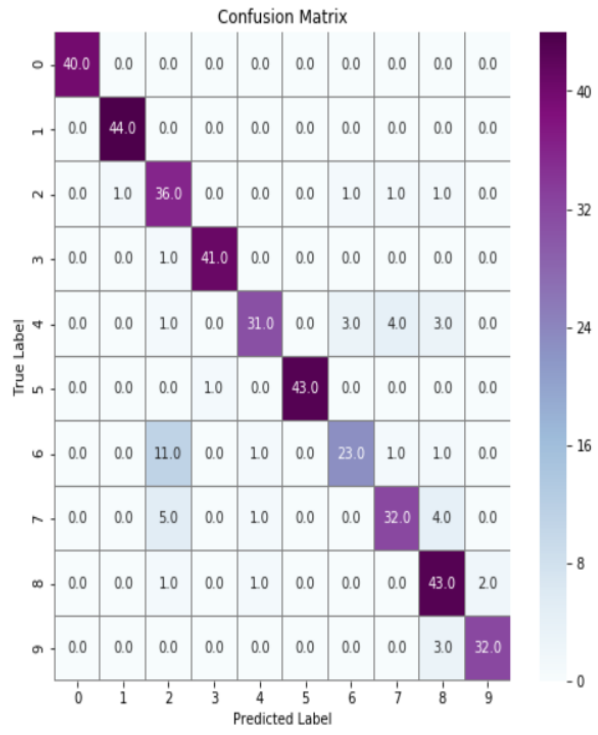
- Representation of validation accuracy and training loss

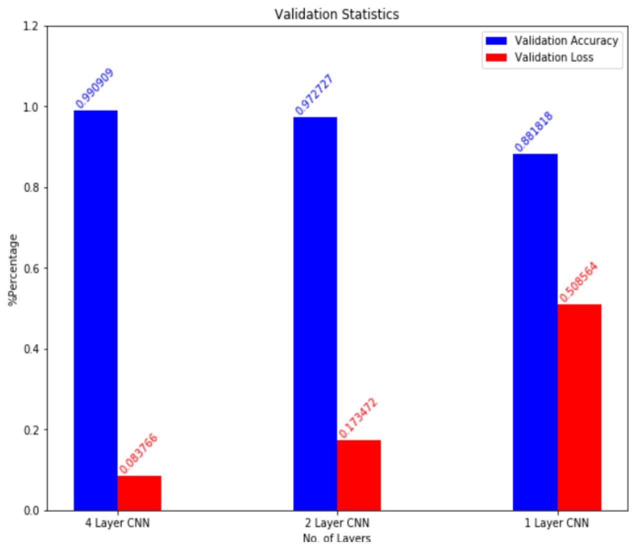


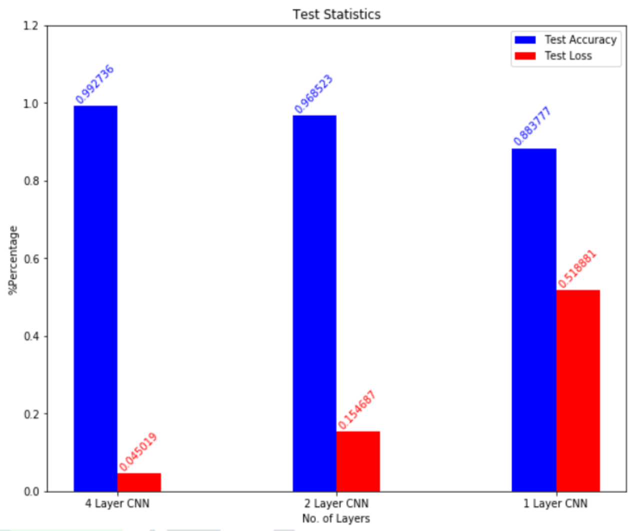
- Representation of validation loss and training accuracy



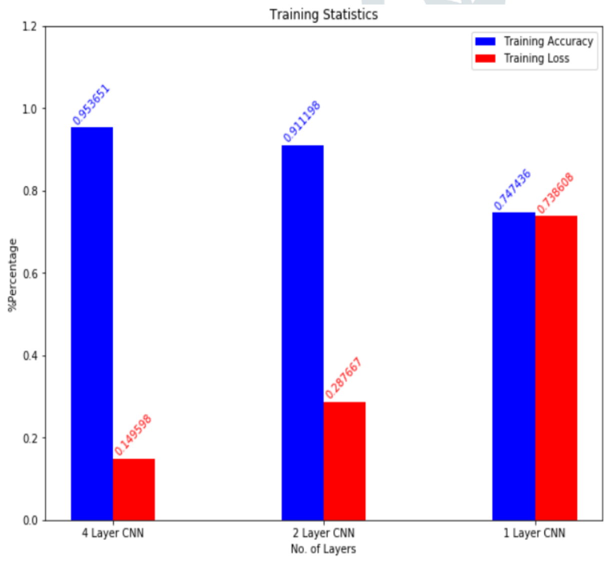
❖ Confusion matrix:







D. *Comparing the scores of the three models*



VII. CONCLUSION

In this paper, the accuracy of three convolutional neural network models are compared on the dataset "Sign-Language-Detection-Dataset". This study will make it much easier for others, to choose the best model for this dataset.

The conclusion of this study is that the model with four CNN layers is the best fit for this dataset. Anyone who works with the same dataset, doesn't have to put extra efforts in testing every model and can directly start with four layers.

VIII. REFERENCES

[1] https://keras.io/preprocessing/image/
[2] https://keras.io/layers/convolutional/
[3] Vivek Bheda and N. Dianna Radpour, "Using Deep Convolutional Networks for Gesture Recognition in American Sign Language".
[4] A.O. Vorontsov and A.N. Averkin "COMPARISON OF DIFFERENT CONVOLUTION NEURAL NETWORK ARCHITECTURES FOR THE SOLUTION OF THE PROBLEM OF EMOTION RECOGNITION BY FACIAL EXPRESSION", *Proceedings of the VIII International Conference "Distributed Computing and Grid-technologies in Science and Education" (GRID 2018), Dubna, Moscow region, Russia, September 10 - 14, 2018.*

[5] Karan Chauhan, Shrwan Ram, "Image Classification with Deep Learning and Comparison between Different Convolutional Neural Network Structures using Tensorflow and Keras", *International Journal of Advance Engineering and Research Development Volume 5, Issue 02, February -2018.*

[6] Shivashankara S, Srinath S, "American Sign Language Recognition System: An Optimal Approach", *I.J. Image, Graphics and Signal Processing, 2018, 8, 18-30 Published Online August 2018 in MECS.*

[7] Chen Wang, Yang Xi, "Convolutional Neural Network for Image Classification".

[8] Mingyuan Xin and Yong Wang, "Research on image classification model based on deep convolution neural network", *EURASIP Journal on Image and Video Processing (2019).*

[9] Md. Abul Kalam, Md. Nazrul Islam Mondal, Boshir Ahmed, "Rotation independent Digit Recognition in Sign Language", *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE).*

[10] Pierre Sermanet, Soumith Chintala, Yann LeCun, "Convolutional Neural Networks Applied to House Numbers Digit Classification".

[11] Dan Cireşan, Ueli Meier, Jonathan Masci, Jürgen Schmidhuber, "Multi-column deep neural network for traffic sign classification", *Elsevier 2012 Special Issue.*

[12] Alex Krizhevsky, Ilya Sutskever, Geoffrey E.Hinton "ImageNet Classification with Deep Convolutional Neural Networks".

[13] Jie Huang, Wengang Zhou, Houqiang Li, and Weiping Li, "SIGN LANGUAGE RECOGNITION USING 3D CONVOLUTIONAL NEURAL NETWORKS".

[14] Li Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research", *IEEE SIGNAL PROCESSING MAGAZINE, November 2012.*