Vivank Sharma
Student, B.Tech,
Vellore Institute of Technology,
Vellore, India.

Dr.Valarmathi B
Associate Professor,
Vellore Institute of Technology,
Vellore, India.

Sumit Jahagirdar
Student, B.Tech,
Vellore Institute of Technology,
Vellore, India

Shobhit Srivastava
Student, B.Tech,
Vellore Institute of Technology,
Vellore, India.

Dr. K. Santhi
Associate Professor,
Vellore Institute of Technology,
Vellore, India.

Performance Analysis of the Classifiers for Optical Character Recognition

***Abstract*— A brain can easily able to recognize handwritten words which computers are still struggling to figure out what they are. This amazing power of the brain requires to be transferred to a computer for automatic digitalization of handwritten texts and manuscript. The work proposed in this paper tried to make a model which is able to recognize and predict different Hindi character which can be further used to identify and digitalize the whole page or text. Our model uses different approaches like a base-line neural network, convolutional neural network, and a simple logistic regression model.**

Keywords – Convolution Neural Network, Base-line, Neural Network, Optical Character Recognition, Devanagari

I. INTRODUCTION

M Machines are being updated on a daily basis by humans and slowly everyday computers are replacing day-to-day task performed by humans and mimic them. Whole world trying to make computers smarter day by day which gave rise to man different IT sectors and deep learning fields. When these became a popular huge amount of handwritten data are generated and needed from all over the world which led to a different language barrier and leads to the need for an automatic system to read that handwritten papers and then convert to necessary language which further help to extract knowledge from them. In this research paper, we tried to read the handwritten scanned images and perform various model analysis on them and finally compare them and identify which model is more accurate and can be used further for many more innovations.

II. RELATED WORK

In [1], they used the database which consists of 4,428 handwritten characters scanned at 300 dpi and after preprocessing data they extracted features like HOG and Projection profile histogram. The highlights removed are tried on different classifiers in particular Quadratic SVM, k-NN, weighted k-NN, Ensemble Subspace Discriminant and Bagged Trees. The fundamental idea of these classifiers is abridged beneath. SVM depends on the idea of regulated realizing which characterizes between objects of different classes utilizing choice hyperplanes.

Linear SVM orders dataset by a straight line between various item classes. In any case, in the vast majority of the order assignment, information is non-directly distinguishable for example no straight line exists to accurately arrange the objects of a dataset. In this way, progressively complex structures are expected to acquire an ideal detachment. Another method for arranging non-straight information is to delineate in higher dimensional space where it displays linearity. SVM give a proficient and simple method for changing to higher dimensional space by utilizing Kernels. On the off chance that a piece work fulfils the Mercers Theorem [2], at that point the internal result of two accepted info vectors(x1,x2) can be figured after they have been mapped in new high dimensional component space by some non-direct mapping[3]. Contingent on the part work utilized, SVM can learn polynomial classifiers, spiral premise work (RBF), or sigmoid capacities as shown in beneath conditions.

Another classifier, k-Nearest Neighbor (k-NN) can likewise be utilized for classification issue. To characterize another object it checks its K-Nearest Neighbors from the preparation information and likeness between two examples can be processed utilizing Euclidean, Minkowski or Mahalanobis Distance. To manage high-dimensional information, weighted k-NN was presented. It depends on loads change until the worthy dimension of characterization precision is come to.

Tree-structure classifier is an elective methodology for characterization. They structure a various leveled structure by dreary cuts of datasets. These classifiers have picked up need from recent years in view of their capacity to deal with

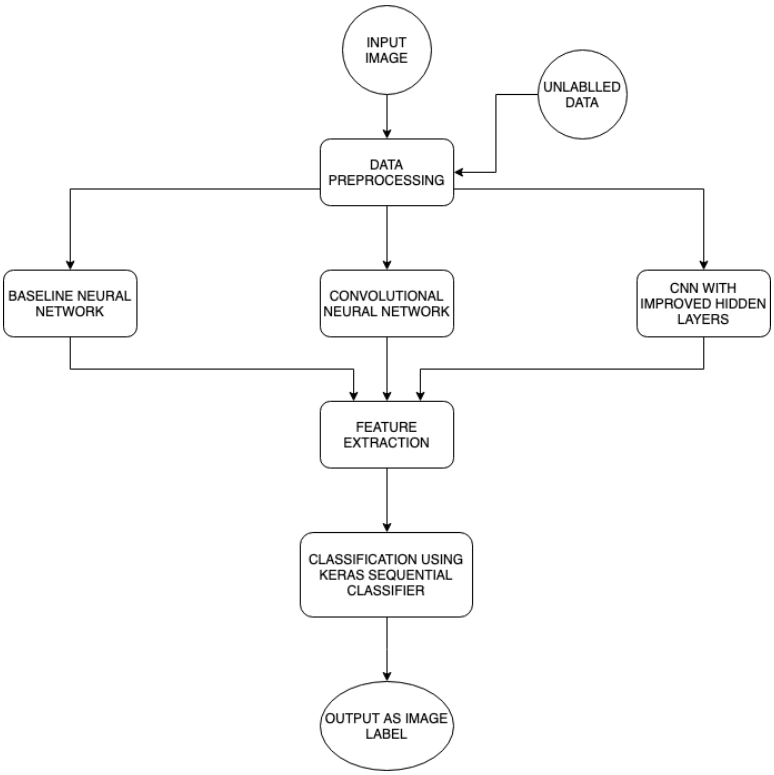
datasets with a substantial number of conditions and impervious to exceptions. Trees can be built dependent on highlights of characters and distinctive choice ways can be distinguished. Packing is a method which can be utilized with neural systems and trees to lessen related to forecasting. This brought about the exactness of 93-95 %.

III. PROPOSED WORK

We will perform an examination on our Devanagari dataset utilizing 2 distinctive neural system in particular Base-line neural system and Convolutional Neural Network (CNN) additionally we will see two instances of CNN demonstrate and break down which performs better.

Our work will begin with first information preprocessing then we will go it through our three neural systems. At that point, we will perform highlight extraction which again then will be utilized for ordering the picture type and remember it has appeared in Figure 1.

Fig. 1. FlowChart of the WorkFlow



Presently for training reason, we have utilized a dataset of Devanagari Script Characters. It includes 92000 pictures [32x32 pixels] comparing to 46 characters, consonants “Ka" to “Gya", and the digits 0 to 9. This dataset was initially made by Computer Vision Research Group, Nepal [4]. This dataset is utilized as a benchmark dataset for Devanagari Characters.

It likewise contains a CSV record which is made subsequent to preprocessing of pictures like Normalisation, Thinning, Noise Removal, and so on. (already preprocessed)

The CSV record is the measurement 92000 * 1025. There are 1024 information highlights of pixel esteems in grayscale (0 to 255). The segment "character" speaks to the Devanagari Character Name relating to each picture. The sample dataset of “Character Ka” is shown in Figure 2.

Fig. 2. Representation of “CHARACTER_KA”

	pixel_0000	pixel_0001	pixel_0002	pixel_0003	pixel_0004	pixel_0005	\
0	0	0	0	0	0	0	
1	0	0	0	0	0	0	
2	0	0	0	0	0	0	
3	0	0	0	0	0	0	
4	0	0	0	0	0	0	

	pixel_0006	pixel_0007	pixel_0008	pixel_0009	...	\
0	0	0	0	0	...	
1	0	0	0	0	...	
2	0	0	0	0	...	
3	0	0	0	0	...	
4	0	0	0	0	...	

	pixel_1015	pixel_1016	pixel_1017	pixel_1018	pixel_1019	pixel_1020	\
0	0	0	0	0	0	0	
1	0	0	0	0	0	0	
2	0	0	0	0	0	0	
3	0	0	0	0	0	0	
4	0	0	0	0	0	0	

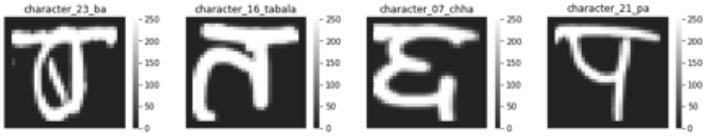
	pixel_1021	pixel_1022	pixel_1023	character
0	0	0	0	character_01_ka
1	0	0	0	character_01_ka
2	0	0	0	character_01_ka
3	0	0	0	character_01_ka
4	0	0	0	character_01_ka

[5 rows x 1025 columns]

Now lets visualise four random images.

Figure 3 demonstrates the Heat Map of four random pictures resized and plotted the columns (pixels) consecutively.

Fig. 3. Heat Map of 4 random characters



We have to split our dataset into two parts. The 80% of the dataset is used for training and 20% of the dataset is used for testing.

We need to likewise encode our names concerning handling strings can't be prepared in a neural system. To do that we first believer the string into Labels.

So all the "character_XX_YY" marks would be mapped to names extending from 0 to 45 (Because we have 46 classes)

E.g Suppose for Image of Devanagari Character ‘क’ the label will be “character_01_ka” where “01” represent the

number of images and “ka” represents its label.

From that point forward, we set up this to be utilized by Keras by utilizing its to categorical [5] work.

Presently we have to reshape the information. Since every point is in the form of a single row. In this model, we're going to work with pictures.

So the neural system needs pictures to separate coarse and fine highlights with the goal that it can prepare and characterize them [6].

Here we compared a proposed method with the baseline model and CNN. The following models are used for training and testing purposes and it is given below. We have to split our dataset into two parts. The 80% of the dataset is used for training and 20% of the dataset is used for testing.

We need to likewise encode our names concerning handling strings can't be prepared in a neural system. To do that we first believer the string into Labels.

So all the "character_XX_YY" marks would be mapped to names extending from 0 to 45 (Because we have 46 classes)

E.g Suppose for Image of Devanagari Character ‘क’ the label will be “character_01_ka” where “01” represent the number of images and “ka” represents its label.

From that point forward, we set up this to be utilized by Keras by utilizing its to categorical [5] work.

Presently we have to reshape the information. Since every point is in the form of a single row. In this model, we're going to work with pictures.

So the neural system needs pictures to separate coarse and fine highlights with the goal that it can prepare and characterize them [6].

Here we compared a proposed method with the baseline model and CNN. The following models are used for training and testing purposes and it is given below.

A. Base Line Model

A Base-line is a strategy that utilizes heuristics, basic outline measurements, arbitrariness, or machine figuring out how to make expectations for a dataset. You can utilize these forecasts to gauge the pattern's execution (e.g., precision). This measurement will at that point move toward becoming what you look at some other machine learning calculation against. [7].

So to manufacture Base-line model we have utilized Keras Sequential model, which basically includes us including a layer in a steady progression consecutively.

The main layer, a.k.a the info layer requires a touch of consideration as far as the state of the information it will be looking at.

So only for the primary layer, we will indicate the info shape, i.e., the state of the information picture - lines, sections, and the number of channels. For initializing model, we will utilize ReLu Activation Function usage described beneath.

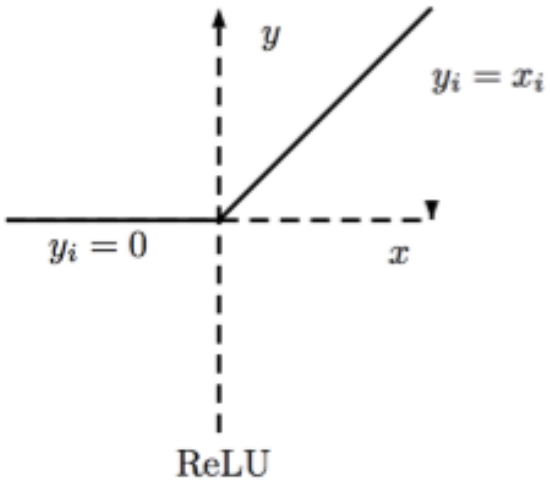
Activation functions are extremely critical for an Artificial Neural Network to learn and comprehend something truly entangled and Non-straight complex utilitarian mappings between the information sources and reaction variable. They acquaint non-direct properties with our Network. Their primary reason for existing is to change over an information signal of a hub in an A-NN to a yield signal.

ReLu (Rectified Linear units) has turned out to be extremely well known in recent years. It was as of late demonstrated that it had multiple times enhancement in combination from “Tanh function”.

It's simply R(i) = max(0,i) (i.e.) if i < 0 , R(i) = 0 and if i >= 0 , R(i) = i. Thus as observing the mathematical type of this capacity, we can see that it is exceptionally basic and efficient. A ton of times in Machine learning and software engineering we see that most straightforward and predictable procedures and strategies are just favored and are ideal. Henceforth it maintains a strategic distance from and corrects disappearing angle issue.[8]

Practically all profound learning Models use ReLu these days. The graph appears in Figure 4 where the x-axis represents ‘ i ’ (input variable) and y represent ‘R(i) Relu Function of ‘ i’.

Fig. 4. RELU Function Graph



As its constraint is that it should just be utilized inside hidden layers of a Neural Network Model.

Subsequently, for yield layers, we should utilize a “softmax” Function for a Classification issue to figure the probabilities for the classes, and for a relapse issue, it ought to just utilize a direct capacity. [9]

Hence the Baseline model will look like as in Figure-5

Fig. 5. Base Line Model

Presently we have built up a standard model for correlation of the precision of alternate models.

```
def baseline_model():
    model = Sequential()
    model.add(Dense(num_pixels, input_dim=num_pixels, kernel_initializer='normal', activation='
relu'))
    model.add(Dense(num_classes, kernel_initializer='normal', activation='softmax'))
    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

B. *Convolutional Neural Network*

CNN is a deep learning calculation which can take in an information picture, appoint significance (learnable loads and inclinations) to different viewpoints/protests in the picture and have the capacity to separate one from the other. The pre-preparing asked in a CNN is much lower when contrasted with other order calculations. While in primitive techniques channels are hand-designed, with enough preparing, CNN can get familiar with these channels/attributes. [10]

Presently when creating CNN designs is to add two convolution plus activation layers consecutive before we continue to the pooling layer for down-sampling. This is done as such that the Kernel estimate utilized at each layer can be little.

At the point, when various convolutional layers are added consecutive, the general impact of the numerous little portions will be like the impact created by a bigger piece, such as having two 3x3 parts rather than a 7x7 bit.

Presently we include hidden layers. When we include two more ConvAct plus ConvAct plus Pool layer grouping. What's more, a straightforward Dropout layer to ensure the system does not rely upon the preparation information excessively.

So we will continue to send this information to the Fully Connected ANN. To do this, our information must be a 1D vector and not a 2D picture. So we level it. What's more, continue to Artificial Neural system ANN and include two progressively hidden layer.

Our concern is order. So succeeding the last layer, we utilize a softmax initiation capacity to order our names. Presently we simply need to characterize the analyser and misfortune capacities to limit and order the CNN.

The Convolutional Neural Network model is shown in FIGURE-6.

Fig. 6. Model of CNN

C. *CNN with an Improvised Hidden Layer*

Now, in this proposed model we changed our ANN layer by adding two hidden dense layers one with 128 units and other with 64 units for giving input in our CNN model and

```
def cnn_model():
    model = Sequential()
    model.add(Conv2D(32, (4, 4), input_shape=(img_height, img_width, img_depth),
                     activation='relu', name="conv_1"))
    model.add(MaxPooling2D(pool_size=(2, 2), name="pool_1"))
    model.add(Conv2D(64, (3, 3), activation='relu', name="conv_2"))
    model.add(MaxPooling2D(pool_size=(2, 2), name="pool_2"))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', name="dense_1"))
    model.add(Dense(50, activation='relu', name="dense_2"))
    model.add(Dense(num_classes, activation='softmax', name="modeloutput"))

    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

activation function of ip_activation which stands for input activation which is none other than ReLu Activation Function as described above.[11]

Modification of hidden layers is shown in FIGURE-7

Fig. 7. Modification of Hidden Layers

```
# Now add the Dense layers
h_dense_0 = Dense(units=128, activation=ip_activation, kernel_initializer='uniform')
cnn.add(h_dense_0)
# Let's add one more before proceeding to the output layer
h_dense_1 = Dense(units=64, activation=ip_activation, kernel_initializer='uniform')
cnn.add(h_dense_1)
```

IV. EXPERIMENTAL RESULT

Keywords Used	
Keyword	**Explanation**
Cycle	**Epoch**
Waste	**Loss**
Precission	**Accuracy**
val_waste	**val_loss**
precission on value	**accuracy on value**

A. *Baseline Model*

For our first model base-line model and the output for training data as given in TABLE-1

TABLE 1 (BASELINE MODEL)

Cycle	1/10							
15s -	waste:	1.2548	precision:	0.6697	val_waste:	0.7903	precision value:	0.7863
Cycle	2/10							
14s -	waste:	0.5911	precision:	0.8434	val_waste:	0.5224	precision value:	0.862
Cycle	3/10							
14s -	waste:	0.3696	precision:	0.9055	val_waste:	0.3931	precision value:	0.8961
Cycle	4/10							
14s -	waste:	0.2563	precision:	0.9356	val_waste:	0.3414	precision value:	0.9081
Cycle	5/10							
15s -	waste:	0.1837	precision:	0.9551	val_waste:	0.2924	precision value:	0.9192
Cycle	6/10							
15s -	waste:	0.1345	precision:	0.97	val_waste:	0.2728	precision value:	0.9251
Cycle	7/10							
15s -	waste:	0.1011	precision:	0.9789	val_waste:	0.2582	precision value:	0.9289
Cycle	8/10							
15s -	waste:	0.0742	precision:	0.9866	val_waste:	0.2478	precision value:	0.9278
Cycle	9/10							
14s -	waste:	0.0543	precision:	0.9919	val_waste:	0.2335	precision value:	0.9357
Cycle	10/10							
14s -	waste:	0.0409	precision:	0.9951	val_waste:	0.238	precision value:	0.9312

On validating our data on test data we get following precision of 93.31 %.

NOW graph of how the model learned over its Cycles is given in Figure 8 and 9 of the model -Accuracy and Model-LOSS respectively.

Fig. 8. *ACCURACY VS EPOCH GRAPH OF MODEL ACCURACY*

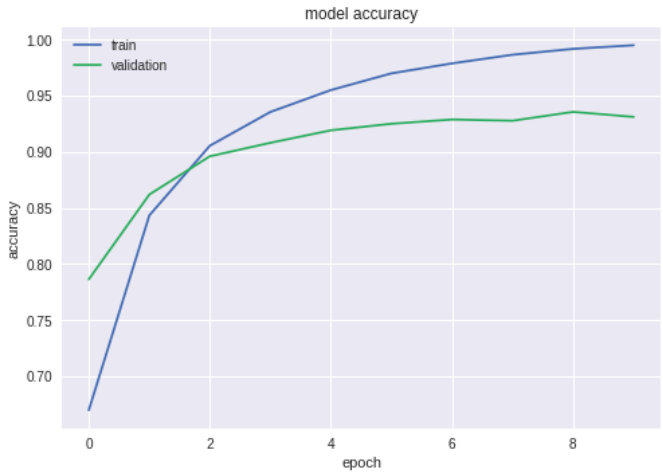
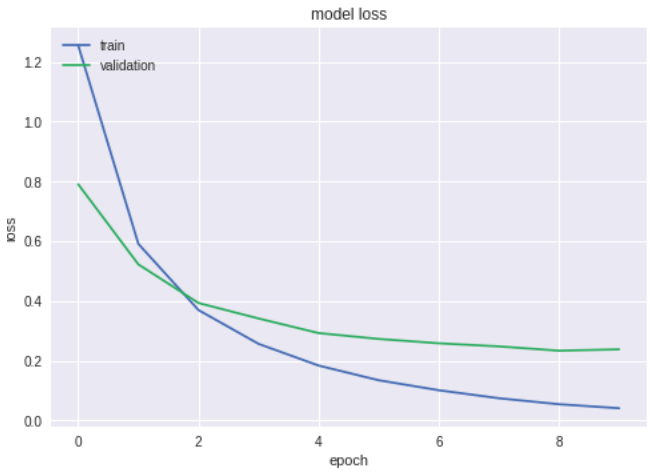


Fig. 9. *ACCURACY VS EPOCH GRAPH OF MODEL LOSS*



B. Convolutional Neural Network

We got output for training data as given in TABLE-2-

TABLE-2 (CNN)

Cycle	1/10							
101s -	waste:	1.2215	precision:	0.6692	val_waste:	0.4054	precision value:	0.8832
Cycle	2/10							
101s -	waste:	0.326	precision:	0.9033	val_waste:	0.2637	precision value:	0.9218
Cycle	3/10							
96s -	waste:	0.2109 -	precision:	0.9361 -	val_waste:	0.1806 -	precision value:	0.9458
Cycle	4/10							
98s -	waste:	0.1524 -	precision:	0.9538 -	val_waste:	0.1670 -	precision value:	0.951
Cycle	5/10							
89s -	waste:	0.1226 -	precision:	0.9614 -	val_waste:	0.1366 -	precision value:	0.9596
Cycle	6/10							
88s -	waste:	0.0974 -	precision:	0.9685 -	val_waste:	0.1304 -	precision value:	0.9624
Cycle	7/10							
88s -	waste:	0.0778 -	precision:	0.9751 -	val_waste:	0.1233 -	precision value:	0.9647
Cycle	8/10							
88s -	waste:	0.0647 -	precision:	0.9794 -	val_waste:	0.1181 -	precision value:	0.9665
Cycle	9/10							
86s -	waste:	0.0577 -	precision:	0.9814 -	val_waste:	0.1148 -	precision value:	0.9672
Cycle	10/10							
87s -	waste:	0.0514 -	precision:	0.9831 -	val_waste:	0.1154 -	precision value:	0.9686

On validating our data on test data we get following precision of 96.85 %.

Now graph of how the model learned over its Cycles is given in figure 10 and 11 of the model - Precision and Accuracy and Model-Loss respectively.

Fig. 10. ACCURACY VS EPOCH GRAPH OF MODEL ACCURACY

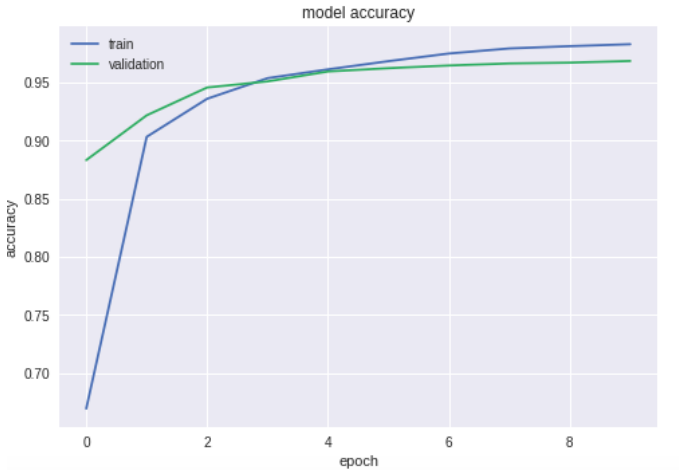
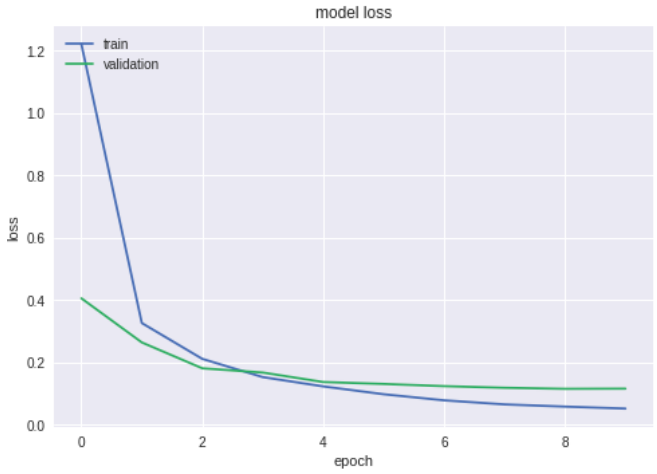


Fig. 11. ACCURACY VS EPOCH GRAPH OF MODEL LOSS



C. *CNN with improvised hidden layers*

We got output for training data as given in TABLE-3-

TABLE 3 (CNN WITH IMPROVED HIDDEN LAYER)

Cycle	1/20										
73600/	73600	54s	731us/step	waste:	0.6275	precission:	0.816	val_waste:	0.1865	precission value:	0.9424
Cycle	2/20										
73600/	73600	49s	667us/step	waste:	0.1665	precission:	0.9486	val_waste:	0.1207	precission value:	0.9632
Cycle	3/20										
73600/	73600	49s	666us/step	waste:	0.111	precission:	0.9659	val_waste:	0.0989	precission value:	0.9721
Cycle	4/20										
73600/	73600	49s	665us/step	waste:	0.0836	precission:	0.9742	val_waste:	0.0976	precission value:	0.9729
Cycle	5/20										
73600/	73600	49s	668us/step	waste:	0.0658	precission:	0.9793	val_waste:	0.0837	precission value:	0.9757
Cycle	6/20										
73600/	73600	49s	666us/step	waste:	0.055	precission:	0.9828	val_waste:	0.0763	precission value:	0.9796
Cycle	7/20										

Cycle	7/20										
7360 0/	7360 0	49s	665us/step	waste:	0.0476	precission:	0.9852	val_waste:	0.0746	precission value:	0.9807
Cycle	8/20										
7360 0/	7360 0	49s	665us/step	waste:	0.0416	precission:	0.9868	val_waste:	0.0713	precission value:	0.982
Cycle	9/20										
7360 0/	7360 0	26s	665us	waste:	0.0404	precission:	0.9872	val_waste:	0.0701	precission value:	0.9821
Cycle	10/10										
7360 0/	7360 0	43s	582us/step	waste:	0.0180	precission:	0.9942	val_waste:	0.0706	precission value:	0.9836

On validating our data on test data we get following precision of 98.36 %.

NOW graph of how the model learned over its given in figure 12 and 13 -of the model -Accuracy and Model-Loss respectively.

Fig. 12. ACCURACY VS EPOCH GRAPH OF MODEL ACCURACY

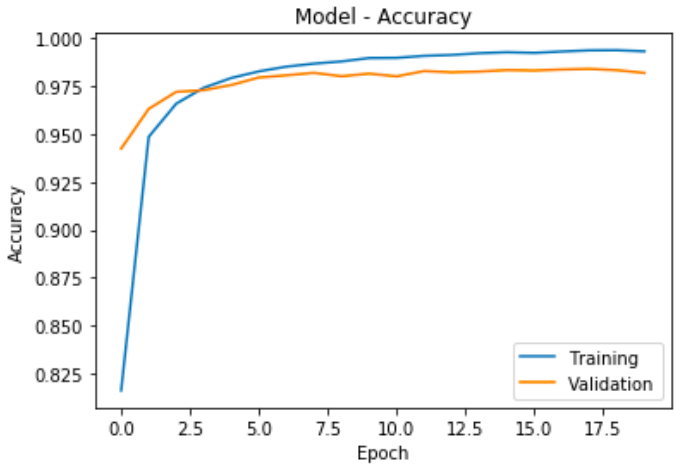
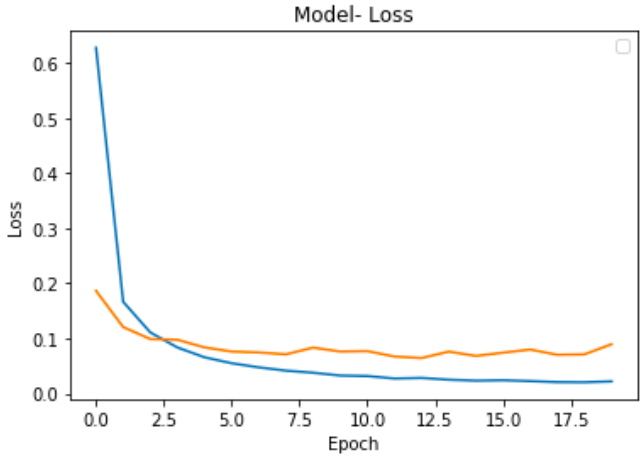


Fig. 13. ACCURACY VS EPOCH GRAPH OF MODEL LOSS



V. CONCLUSION

MODEL	Accuracy
Baseline	93.31
Convolution Neural Network	96.65
CNN with improved hidden layers	98.36

From the above Table, we can conclude that CNN gives better accuracy than the Baseline model and this accuracy can be increased by improving the hidden layers of the CNN and changing neural networks.

For future work we are planning to extend the model to optically recognize not only characters but also for paragraphs as this can be used by splitting the paragraph into words and then again into the character which can be further used for translation further in real world and context based image searching [11]

VI. SOME COMMON MISTAKES

Some images which are round and have left out strokes are hard to predict and hence some images can be mislabelled and because of that our model accuracy can effect in real world.

VII. REFERENCES

1. Yadav, M., & Purwar, R. (2017). Hindi handwritten character recognition using multiple classifiers. 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence. doi: 10.1109/confluence.2017.7943140
2. Vapnik,"Support-vector Networks", Journal of Machine Learning, vol. 20, no.3, pp. 273-297, 1995.
3. Thorsten Joachims,"Learning to Classify Text Using Support Vector Machines", Springer, vol.668, 2002.
4. Computer Vision Research Group, Nepal. [website archive] (https://web.archive.org/web/20160105230017/http://cvresearchnepal.com/wordpress/dhcd/)
5. Keras Utils https://web.archive.org/web/20160105230017/http://cvresearchnepal.com/wordpress/dhcd/)
6. Optical Character Recognition for Sanskrit Using Convolution Neural Networks Meduri Avadesh, 2018
7. DEEP NEURAL NETWORK BASELINE FOR DCASE CHALLENGE 2016 Qiuqiang Kong, Iwnoa Sobieraj, Wenwu Wang, Mark Plumbley Centre for Vision, Speech and Signal Processing, University of Surrey, UK {q.kong, iwona.sobieraj, w.wang, m.plumbley}@surrey.ac.uk
8. H. Ide and T. Kurita, "Improvement of learning for CNN with ReLU activation by sparse regularization," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, 2017, pp. 2684-2691.
9. B. Xin, T. Wang and T. Tang, "A deep learning and softmax regression fault diagnosis method for multi-level converter," 2017 IEEE 11th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED), Tinos, 2017, pp. 292-297
10. S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, 2017, pp. 1-6.
11. X. Yin, W. Chen, X. Wu and H. Yue, "Fine-tuning and visualization of convolutional neural networks," 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA), Siem Reap, 2017, pp. 1310-1315.
12. W. Ahmad and C. M. S. Faisal, "Context based image search," 2011 IEEE 14th International Multitopic Conference, Karachi, 2011, pp. 67-70.