May – June 2020
ISSN: 0193-4120 Page No. 5501-5511

# Hybrid Modeling with Inception based HMM for Face Recognition

**Lakshmi Patil, V.D. Mytri, Kiran Maka**

Dean, Sharnbasva University, Klb, Karnataka, India

Pro vice Chancellor, Sharnbasva University Klb, Karnataka, India

Dean, Dept. of MCA ,Sharnbasva University, Klb, Karnataka, India+

***Article Info***
***Volume 83***
***Page Number: 5501 - 5511***
***Publication Issue:***
***May - June 2020***

***Article History***
***Article Received:*** *19 November 2019*
***Revised:*** *27 January 2020*
***Accepted:*** *24 February 2020*
***Publication:*** *17 May 2020*

## Abstract

Principal, Appa Institute of Engineering and Technology, Klb, Karnataka, India Abstract— Since HMM models need observational feature vectors as input, the encoding of inception models are used in the HMM as observational vectors, which is very novel in the current face recognition methodologies. The accuracy has improved with this approach. SVD based linearly combined feature input models, CNN and CNN inception models, SVD and deep learning based hybrid models are trained and tested for performance on the ORL dataset. Two samples of data sets are drawn from the ORL dataset to create two distinct training and test sets. The performance of the models are measured on the both the datasets. The performance of all the models are compared with the base line model SVD based HMM. The accuracies improved when the proposed hybrid model based on deep learning and HMM was used, to 99.5% and 100% for ORL-Set 1 and ORL-Set 2. Finally, important conclusions of the research work are presented.

***Keywords:*** *Convolutional Neural Networks, Inception Models, CNN, Face Recognition, Hybrid Models in Face Recognition*

## I. INTRODUCTION

A neural network consists of input layer, hidden layers and output layers. If the number of hidden layers is more than 2, then, it is considered as deep neural network (DNN). In a deep learning model, features of the face are extracted automatically with an extraction layer, whereas in case of machine learning model, the feature are extracted manually with different techniques and then it is input to the fully connected neural network for extraction. In a fully connected neural network, every neuron in the hidden layers and output layer is connected with the every neuron in the previous layer. The connections of one neuron with a neuron in the previous layer have a weight associated with it. For example, if there is an image of 96 x 112 x3 (96 pixel in width, 112 pixels in height and 3 color channels), then number of weights in the hidden layer that immediately next to the input layer are 96*112*3 = 32,256 weights. If the size of input images are 300x400x3, then the number of weight become 360,000 which is unmanageable computationally without high end hardware. Also the number of weights adds up in the subsequent layers. Hence the total number of layers in the model is dependent on the size of the input images.

Convolutional Neutral Networks (CNN) [1-5] is designed to overcome this problem of more number of weights. It creates the three dimensional neurons. The neurons are arranged in a two dimensional manner and a third dimension added to extract the features from the images. In a CNN, every neuron in the intermediate layers is connected with a region in the previous layer. The output layer will have a have number of neurons equal to number of classes to be discriminated.

INPUT [96x112x3] has a set of input values of pixels of input image of width 96, height 112 and three color channels.

CONV layer computes the convolution of a region in the input layer with a filter and produces a single output that goes into a neuron in the next layer. The convolution [6-7] is a dot product between the weights of the filter and the region in the previous layer. One can have as many filters as desired for an accurate solution. Hence the CONV layer may have

the dimension of 96x112x20, where the there are 20 filters.

RELU layer will apply the activation function on the output of convolution and it outputs maximum of (0, x), which means, if the convolved value is less than zero, it adjusts the value to zero. Hence the final output of RELU layer will have a dimension same as CONV layer.

POOL layer is down sampling layer [8-13] in which, every region in the previous layer is converted to just one value by taking either maxim values (MAX POOL) or average value (AVG POOL). It reduces the size of output volume (48x56x20).

FC layer is a fully connected neural network and is input with the output of previous layer (POOL LAYER) after flattering the layer into a single column vector. The FC layer is the classification layer and all the previous layers are feature extraction layers. The output layer of FC may be a sigmoid function which outputs a probability of the binary class or it can be a softmax layer [14-15] which outputs probabilities for multi classes.

GoogLeNet Inception Model (GoogLeNet) has a new addition of 1x1 convolution and the FC layer is replaced with a average pooling layer at the end. The 1x1 Convolution is considered as dimensionality reduction module to reduce the number of computations. Since the dimensionality is reduced, in order to accommodate the loss of information, the number of filter along the depth can be increased. Suppose that there is volume of 14x14x480 to be convolved with 5x5 convolution as discussed in CNN model, and with a 1x1 convolution as introduced by GoogLeNet.

For example, if there is a volume of 10x10x200, then by convolving this volume with a filter of 5x5 then number of operations with 5x5 convolution = (10x10x20) x (5x5x200) = 100M. If the same volume is first convolved with 1x1 first and with 5x5 then,

- Number of operations with 1x1 convolution = (10x10x10) x (1x1x200) = 200,000.
- Number of operations with 5x5 convolution = (10x10x20) x (5x5x10) = 2M

Total number of operations is 2.2M which is mush lesser than 100M had the convolution was computed with just 5x5 filter. 1x1 Convolution can be considered as a dimensionality reduction with a non-linear way and PCA is a dimensionality reduction with a linear way.

CNN has been employed to solve a face recognition problem in [16-19]. A unique and novel approach of CNN can found in the ref [20]. There are many other CNN variants exist like light CNNs [21] and VGG face descriptors [22]. Google [23] has published its research in the area of face recognition as FaceNet and Facebook [24] also published its research as DeepNet. The GoogLeNet has been a milestone in the history of CNNs [25] which was published in the year 2014. The focus of GoogLeNet was to reduce the computational complexity.

In the next section, the deep learnig models that are used in this research work, namely, Neural Networks - Hidden Markov Models (CNN-HMM), Convolutional Neural Networks – Inception - Hidden Markov Models (CNN-INC-1-HMM), Convolutional Neural Networks – Inception – Mean 2 - Hidden Markov Models (CNN-INC-MEAN-2-HMM), Convolutional Neural Networks – Inception – Mean 3 - Hidden Markov Models (CNN-INC-MEAN-3-HMM), Convolutional Neural Networks – Inception – Mean 4 - Hidden Markov Models (CNN-INC-MEAN-4-HMM) and Convolutional Neural Networks – Inception – Mean 5 - Hidden Markov Models (CNN-INC-MEAN-5-HMM) are introduced. In Sec III, the simulation results are presented. In Sec IV conclusions are presented.

## II. DEEP LEARNING MODELS

In the research work of the authors [27] face recognition algorithms like Hidden Markov Models (HMM) and CNN-Inception based models were developed with new approaches. HMM models are basically sequence based models and model was developed by dividing the image into various states. Each state of the image is input to the model for training and testing purposes.

HMM models need the data to be presented in the form of states and states are represented with features. Each state has to be defined with a set of feature which can be used to input into HMM for training. An image can be used for training the HMM so that HMM can construct the state transition and emission probabilities. Deriving the state transition and emission probabilities from input images is known as training of HMM. Similarly, when some images need

to be tested, each image in the test set must be divided into the same number of states of training images. When test images are presented to the HMM, the transition and emission probabilities are used to determine the label of that input image. Here, the state is represented by a set of features derived from the image. Features must be the derived in the form of number so that computer can understand those numbers. The features mentioned in this context should not be confused with physical features of the face. As shown in Fig. 1, features can be derived from any of the following:

1. Pixel intensities of the image
2. Statistical parameters of the distribution of pixel intensities
3. Discrete Cosine Transformation Coefficients
4. KLT Coefficients
5. Singular Values etc

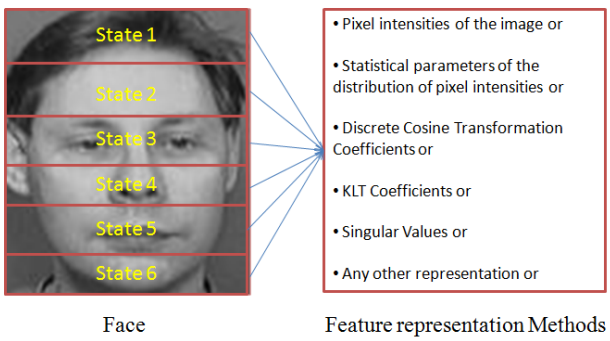


**Fig. 1: Feature representation methods for a typical facial image for HMM**

In [26, 27], a new method was proposed to represent the features of the states. The singular values of the pixel intensity matrix were combined linearly and the input to train the HMM. The proposed method has shown improvement in the successful face recognition rates for ORL-Set 1 and ORL-Set 2. In this paper, another method is proposed to represent the features. Features are extracted from the states of each image from the deep learning models. These features that were extracted from the deep learning models are input to HMM as features. The deep learning models [27] used in this research work to extract features are:

1. Convolutional Neural Networks (CNN)
2. Convolutional Neural Networks – Inception (CNN-INC-1)
3. Convolutional Neural Networks – Inception – Mean 2 (CNN-INC-MEAN-2)
4. Convolutional Neural Networks – Inception – Mean 3 (CNN-INC-MEAN-3)
5. Convolutional Neural Networks – Inception – Mean 4 (CNN-INC-MEAN-4)
6. Convolutional Neural Networks – Inception – Mean 5 (CNN-INC-MEAN-5).

It has been demonstrated in [27] that, CNN-INC-MEAN-3 yields the best performance for ORL-Set 1 and ORL-Set 2 data sets. In this paper, the features extracted from the CNN models and input to HMM models. The following models were simulated for the face recognition:

1. Convolutional Neural Networks - Hidden Markov Models (CNN-HMM)
2. Convolutional Neural Networks – Inception - Hidden Markov Models (CNN-INC-1-HMM)
3. Convolutional Neural Networks – Inception – Mean 2 - Hidden Markov Models (CNN-INC-MEAN-2-HMM)
4. Convolutional Neural Networks – Inception – Mean 3 - Hidden Markov Models (CNN-INC-MEAN-3-HMM)
5. Convolutional Neural Networks – Inception – Mean 4 - Hidden Markov Models (CNN-INC-MEAN-4-HMM)
6. Convolutional Neural Networks – Inception – Mean 5 - Hidden Markov Models (CNN-INC-MEAN-5-HMM)

Extraction of features from the images from the CNN models is an important step here before using it in the HMM models. In all the CNN models that are discussed in [27], whole image was presented to the input layer. The output of the CNN models differs depending upon the architecture of the CNN. For example, in the simple CNN model, the output was the probability of the image belonging to certain class. In the models that are developed for ORL-Set 1 [26-27] and ORL-Set 2 [26-27], the output was representing the probability of the image belonging to a certain person out of 40 persons.

In case of CNN-Inception model, the output was in the form of encodings for each image. When the methods like CNN-INC-MEAN-2, CNN-INC-MEAN-3, CNN-INC-MEAN-4 and CNN-INC-MEAN-5 were used, the encodings were in fact averaged over 2, 3, 4 and 5 facial expressions of the same person. In other words, features in the CNN-

Inception models were defined from encodings directly or indirectly.

Since HMM models were based on the states and each image is split into states, the CNN output must match with the input for HMM models. That means, the output of CNN-Inception models must be for each state of the image instead of being for the whole image. Then if there are 52 states in the facial image, then there will be 52 encodings. Each encoding is a vector of length 128.

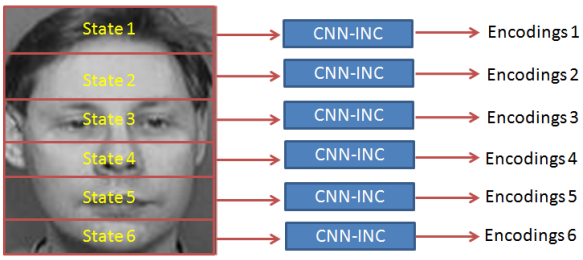


**Fig. 2:** Feature extraction for a typical facial image using CNN-INC models

Fig. 2 shows the feature extraction for 6 states when CNN-Inception models are used. The states shown in the figure does not have overlap of states. The overlap is not shown here for the sake of better display of the method. In the actual simulations, the states are overlapped.

Similarly, when simple CNN model is to be used with HMM, features are extracted for each state in the form of probability of that state belonging to a person. Images are split into a number that is equal to the number of states. The softmax output of CNN model for each state of the image is the feature to be input to HMM. In the case of ORL dataset, there are 40 persons and hence vector size of softmax output would be 40.

When CNN-INC-MEAN-2, CNN-INC-MEAN-3, CNN-INC-MEAN-4 and CNN-INC-MEAN-5 are to be used for feature extraction, then the encodings of same states of different facial expressions are averaged. For example, in CNN-INC-MEAN-3 model, for state 1, the encodings of state 1 of three facial expressions are averaged. The same process is repeated for all the states.

**CNN-HMM Algorithm:**

1. Each input image is divided into desired number of states.
2. Input the each state of facial image to the CNN model.
3. Perform a 2D convolution on the input image.
4. Perform batch normalization to normalize the convolved values.
5. Perform a max-pooling to select a maximum value of the window.
6. Repeat step 2 to step 4 thrice (a total of four times).
7. Arrange all the values in a vector form by flattening the data from fourth max-pooling layer.
8. Input the flattened data in the form vector to a dense layer.
9. Output from step 7 is processed in the next hidden dense layer.
10. Process the output from output from step 8 in a softmax layer.
11. The vector index in the softmax layer that has highest probability is the person of the input facial image.
12. Repeat this process for all the training images so that weights in the network are optimized.
13. Input the output from softmax layer as features to HMM for face recognition.
14. Run Baum-Welch algorithm to derive the optimum transition and observation matrices.
15. Input the state images of each facial image in test set one after the other to trained CNN model to find the vector of probabilities from softmax layer.
16. Input the output from softmax layer as features to the trained HMM for face recognition.
17. Repeat steps 15 and 16 for all the images.

**CNN-INC-MEAN-x-HMM Algorithm:**

1. Initialize the weights of network with GoogLeNet transfer learning model.
2. Each input image is divided into desired number of states.
3. Input the each state of facial image to the CNN-INC model.
4. Determine the encodings of the image with the inception network.
5. Repeat this process for all the training images and store the encodings in a train-face-state-

matrix. Each row in the matrix is the encodings of a certain state of a facial image.

6. Input the images in test set one after the other, determine the encodings and store it in a test-face-state-matrix. Each row in the matrix is the encodings of certain state of facial image.
7. Since there are five facial expression for each person in train set, create another train-person-state-matrix with average of encodings of N facial images of a person where N <= 5. Each row in the train-person-state-state matrix is the average encodings of certain state of each person.
8. Input the train-person-state-matrix as features to HMM for training.
9. Run Baum-Welch algorithm to derive the optimum transition and observation matrices.
10. Input the test-person-state-matrix as features to HMM for face recognition.

Repeat steps 10 for all the images.

## III. The Simulation Results

In this section, accuracy of the models that are introduced in this paper is presented. Comparison of the accuracy is made with respect to the models discussed in [26 ] and [27]. Performance is presented in the form of percentage accuracy in recognizing the correct person for each of the facial expression in the test set of ORL-Set 1.

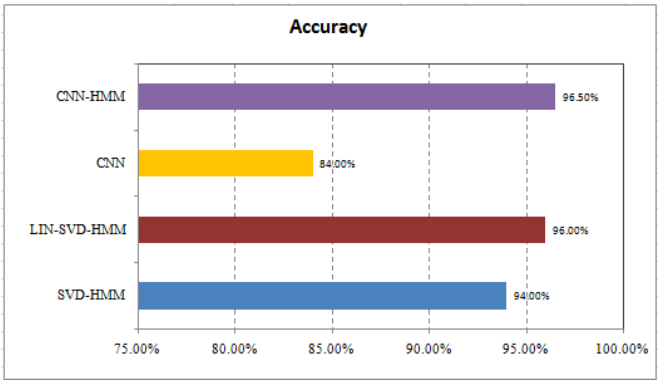


**Fig. 3:** Accuracy of Hybrid Model CNN-HMM for ORL-Set 1

Fig 3 shows the accuracy of the CNN-HMM for ORL-Set 1. The accuracy of CNN-HMM is compared with the SVD-HMM, LIN-SVD-HMM and CNN models that were discussed in [26] and [27]. It can be observed that CNN-HMM has yielded an accuracy of 96.5%, where as the CNN yielded only 84%. Hence this is a big jump in the accuracy. Also when compared with the HMM models of Chapter V, SVD-HMM and LIN-SVD-HMM, accuracy of CNN-HMM is better by 2% and 0.5% respectively. Hence, it can be understood that when features of the CNN model presented to HMM, the results improved.

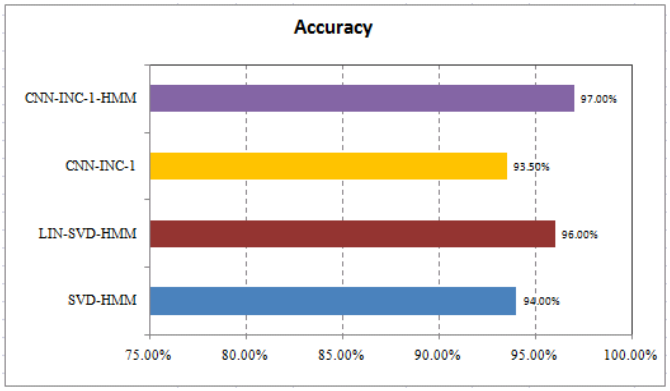


**Fig. 4:** Accuracy of Hybrid Model CNN-INC-1-HMM for ORL-Set 1

Fig. 4 shows the accuracy of the hybrid model CNN-INC-1-HMM for ORL-Set 1. The lift obtained by the CNN-INC-1-HMM over the CNN-INC-1 is 3.5%. So the HMM has contributed to the recognition rate of 3.5% over the CNN-INC-1 since the features of CNN-INC-1 was presented to HMM instead of using it directly for face recognition. When compared with the other HMM models like SVD-HMM and LIN-SVD-HMM, CNN-INC-1-HMM has given a lift of 3% and 2% respectively. The 3% and 2% are the contribution from the features of CNN-INC-1.

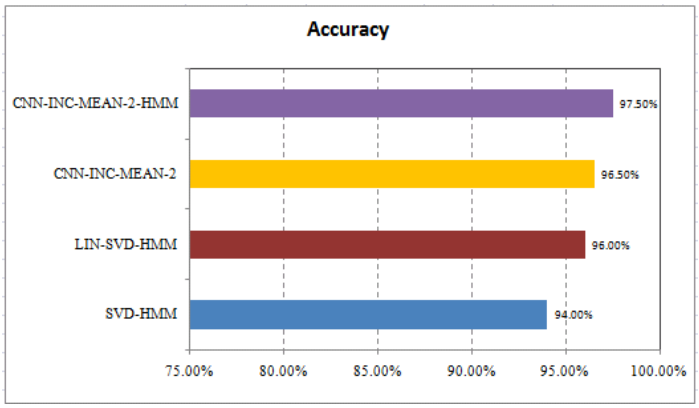

**Fig. 5:** Accuracy of Hybrid Model CNN-INC-MEAN-2-HMM for ORL-Set 1

Fig. 5 shows the percentage accuracy of the hybrid model. The hybrid model used here is CNN-INC-MEAN-2-HMM. The hybrid model has resulted in a lift of 1%, 1.5% and 3.5% over the models CNN-INC-MEAN-2, LIN-SVD-HMM and SVD-HMM. These results along with the above results prove the fact that features extracted from the CNN-Inception models gives lift over the individual models like CNN-INC-MEAN-2, LIN-SVD-HMM and SVD-HMM.

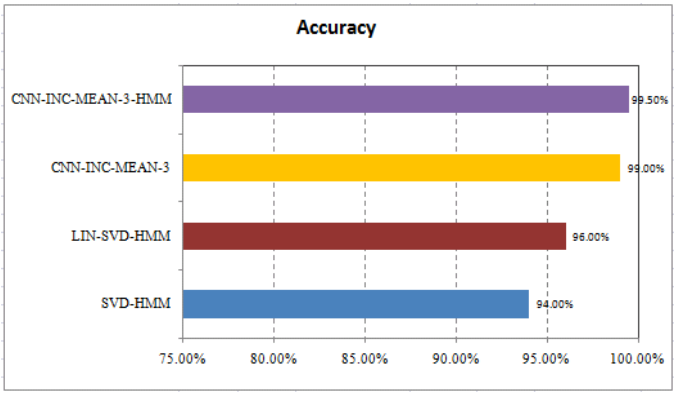


**Fig. 6:** Accuracy of Hybrid Model CNN-INC-MEAN-3-HMM for ORL-Set 1

As it is established in [27], the accuracy of CNN-INC-MEAN-3 model was best along with CNN-INC-MEAN-4 among all the models compared for ORL-Set 1. The accuracy stands at 99% for ORL-Set 1. When the features of the CNN-INC-MEAN-3 model were input to a HMM model, the accuracy of model CNN-INC-MEAN-3-HMM jumped to 99.5%, which is 0.5% lift, as shown in Fig. 7.6. When compared with CNN-INC-MEAN-2-HMM, the lift is 2%. Compared to other HMM models like SVD-HMM and LIN-SVD-HMM, lift achieved by CNN-INC-MEAN-3-HMM is 5.5% and 3.5% respectively.

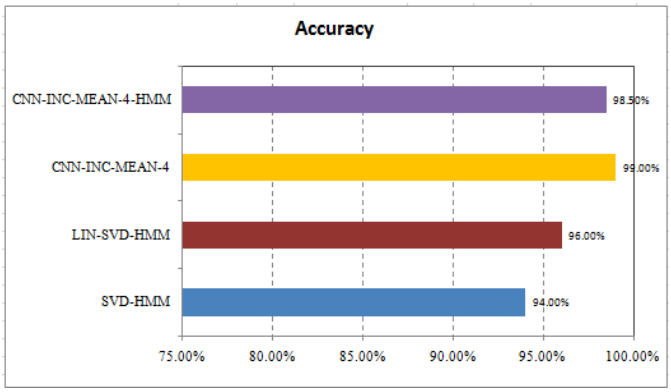


**Fig. 7:** Accuracy of Hybrid Model CNN-INC-MEAN-4-HMM for ORL-Set 1

As shown in Fig. 7, the accuracy of CNN-INC-MEAN-4 model was best along with CNN-INC-MEAN-3 among all the models compared for ORL-Set 1. The accuracy stands at 99% for ORL-Set 1. However, model CNN-INC-MEAN-4-HMM has yielded accuracy of 98.5%, which is -0.5% lift. That means when compared with CNN-INC-MEAN-3-HMM, there is negative lift. Compared to other HMM models like SVD-HMM and LIN-SVD-HMM, lift achieved by CNN-INC-MEAN-4-HMM is 4.5% and 2.5% respectively.

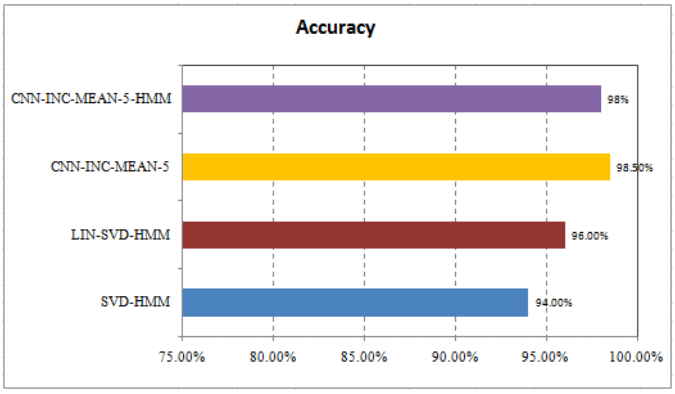


**Fig. 8:** Accuracy of Hybrid Model CNN-INC-MEAN-5-HMM for ORL-Set 1

As shown in Fig. 8, the accuracy of CNN-INC-MEAN-5-HMM model is lower than the CNN-INC-MEAN-3-HMM and CNN-INC-MEAN-4-HMM by 1.5% and 0.5% respectively. It is already discussed in last chapter that CNN-INC-MEAN-5 model performance was inferior to CNN-INC-MEAN-3 and CNN-INC-MEAN-4. Hence the features of CNN-

INC-MEAN-5 model, when input to HMM, it underperforms. Also CNN-INC-MEAN-5-HMM model performance was less by 0.5% than CNN-INC-MEAN-5 model.

As a next test, ORL-Set 2 dataset is used for determining the performance of the hybrid model. Fig. 9 shows the accuracy of the models SVD-HMM, LIN-SVD-HMM, CNN and CNN-HMM models. Accuracy of SVD-HMM, LIN-SVD-HMM, CNN and CNN-HMM models are 93.5%, 94.5%, 88.5% and 95.5% respectively. It can be noticed that the lift obtained by using the features of inception models in HMM models is 7% over the CNN model. But when compared with the SVD-HMM and LIN-HMM models are 2% and 1% respectively.

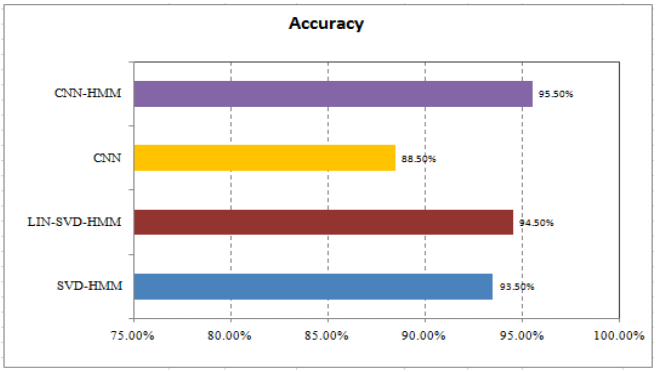


**Fig. 9:** Accuracy of Hybrid Model CNN-HMM for ORL-Set 2

Fig. 10 shows the accuracy of Model CNN-INC-1-HMM for ORL-Set 2 dataset. When inception model is used instead of CNN model as deep learning model for the inputs for HMM, the lift obtained by CNN-INC-1-HMM is 0.5% over CNN-HMM. The accuracy of CNN-HMM is 95.5% and that by CNN-INC-1-HMM is 96%. When compared with the CNN model it is 7.5% lift in the accuracy when CNN-INC-1-HMM model is used. However, the model CNN-INC-1 model has yielded 93.5% which is the model without HMM. That means the deep learning model CNN-INC-1 has yielded an accuracy of 93.5% when there was no input to HMM from CNN-INC-1. But with input of the features from CNN-INC-1 to HMM (CNN-INC-1-HMM), accuracy has increased to 96%, which is 2.5% lift. This is a significant improvement in the accuracy. Similarly, when compared with SVD-HMM and LIN-SVD-HMM models, lift the accuracy of CNN-INC-1-HMM is 2.5% and 1.5%. Hence it can be concluded that the combination of CNN-INC model with HMM has improved the accuracy compared to the combination of SVD and HMM.

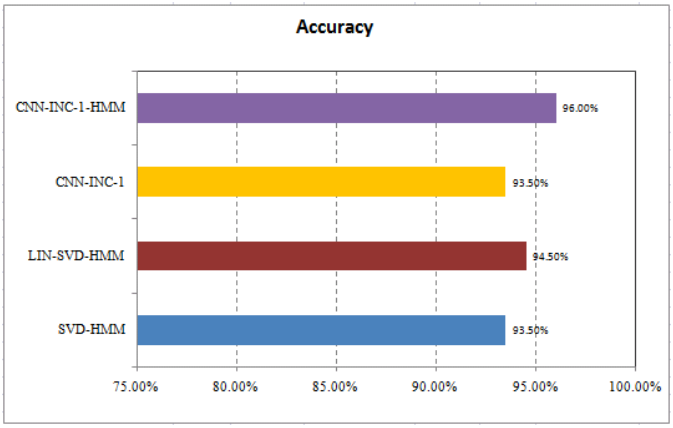


**Fig. 10:** Accuracy of Hybrid Model CNN-INC-1-HMM for ORL-Set 2

When CNN-INC-MEAN-2-HMM model was used the accuracy increased to 98.5% as shown in Fig. 11. As explained in the last chapter, in CNN-INC-MEAN-2 model, the features are averaged for two facial expressions. That means the features are averaged and then fed to the HMM model. Accuracy of CNN-INC-MEAN-2 model was at 98% for ORL-Set 2 that is without having the features fed to the HMM model. That means there is an improvement in the accuracy by 0.5% when HMM was used along with CNN-INC-MEAN-2. When compared with the CNN-INC-1-HMM model, CNN-INC-MEAN-2-HMM model has got a lift of 2.5% for ORL-Set 2. Compared with other HMM models like SVD-HMM and LIN-SVD-HMM models, the lift in the CNN-INC-MEAN-2-HMM model is 5% and 4% respectively.

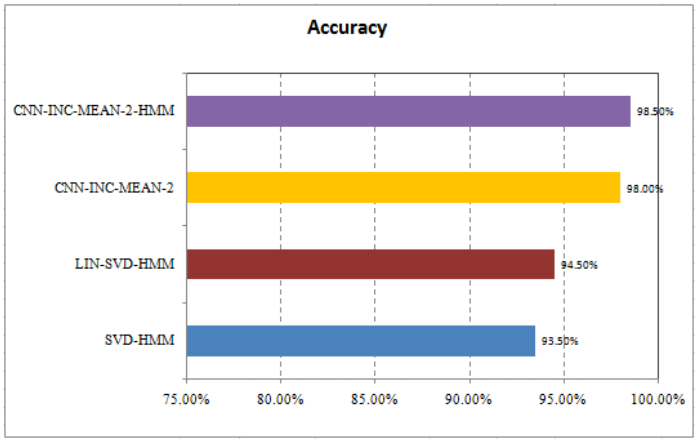


**Fig. 11:** Accuracy of Hybrid Model CNN-INC-MEAN-2-HMM for ORL-Set 2

As it is already shown in the above paragraphs and last chapter, the CNN-INC-MEAN-3-HMM model yielded 99.5% for the ORL-Set 1. In case of ORL-Set 2, CNN-INC-MEAN-3 model, with no HMM in it has yielded the accuracy of 100% as shown in Fig. 11. Hence even without feeding the features of CNN-INC-MEAN-3 into HMM, the accuracy is already at maximum. When the features of CNN-INC-MEAN-3 are fed to HMM, the accuracy still stood at 100%. In case of ORL-Set 1, there was a lift of 0.5% over the CNN-INC-MEAN-3 model, but in case of ORL-Set 2, the lift is zero percentage since the accuracy is already at 100%.The lift is maximum over the other HMM models for CNN-INC-MEAN-3-HMM with 6.5% and 5.5% over the SVD-HMM and LIN-SVD-HMM.

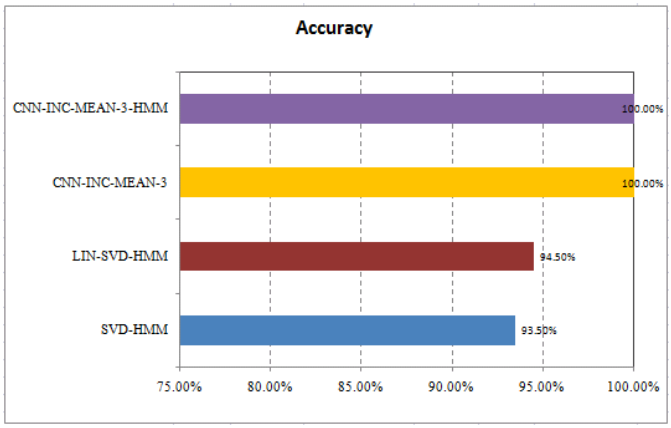


**Fig. 7.12:** Accuracy of Hybrid Model CNN-INC-MEAN-3-HMM for ORL-Set 2

For ORL-Set 1, the accuracy had a negative lift when CNN-INC-MEAN-4-HMM was compared with CNN-INC-MEAN-4. The negative lift was at -0.5%. This was lower than the CNN-INC-MEAN-3-HMM. But when CNN-INC-MEAN-4-HMM was run on the ORL-Set 2, the accuracy stood at 99% which is equal to the accuracy of CNN-INC-MEAN-4 as shown in Fig. 13. Hence it is concluded that fall in the lift for CNN-INC-MEAN-4-HMM in ORL-Set 1 was only random and it is not recurring when ORL-Set 2. CNN-INC-MEAN-3-HMM model and CNN-INC-MEAN-4-HMM have yielded 100% and 99% accuracy on ORL-Set 2, respectively. In case of ORL-Set 1, the CNN-INC-MEAN-3-HMM and CNN-INC-MEAN-4-HMM models performed equally at 99% and 98.5% respectively. Hence CNN-INC-MEAN-3-HMM model is the best model compared to CNN-INC-MEAN-4-HMM model both for ORL-Set 1 and ORL-Set 2.

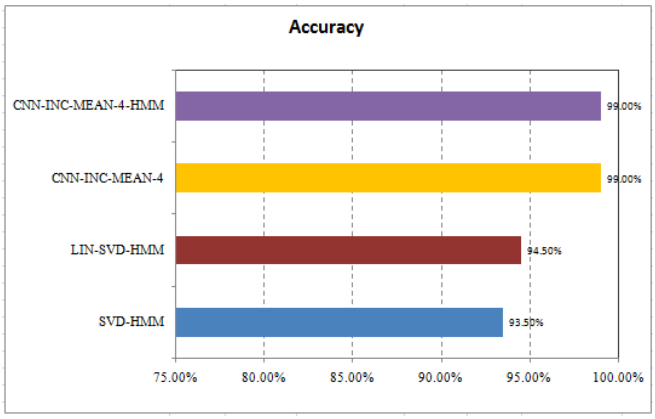


**Fig. 13:** Accuracy of Hybrid Model CNN-INC-MEAN-4-HMM for ORL-Set 2

As explained in the previous section, accuracy of CNN-INC-MEAN-5-HMM model is less than CNN-INC-MEAN-3-HMM and CNN-INC-MEAN-4-HMM by 1.5% and 0.5% for ORL-Set 1, respectively. In case of ORL-Set 2, accuracy of CNN-INC-MEAN-5-HMM model is less than CNN-INC-MEAN-3-HMM by 1% and equal to and CNN-INC-MEAN-4-HMM. Overall, it can be concluded that CNN-INC-MEAN-3-HMM has yielded the best accuracy at 99% for ORL-Set 1 and 100% for ORL-Set 2 among all the models presented in last two chapters and also in the present chapter.

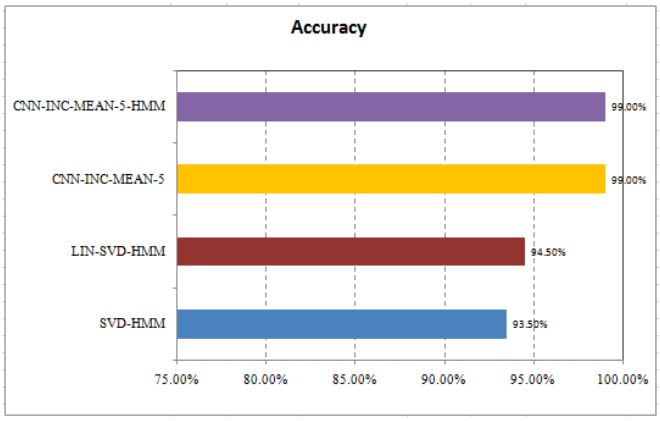


**Fig. 14:** Accuracy of Hybrid Model CNN-INC-MEAN-4-HMM for ORL-Set 2

**Table 1:** Accuracy of 6 hybrid DNN/HMM algorithms measured on Two Datasets

| Sl. No | Model | ORL-Set 1 Accuracy | ORL-Set 2 Accuracy |
|---|---|---|---|
| 1 | CNN-HMM | 96.50% | 95.50% |
| 2 | CNN-INC-1-HMM | 97.00% | 96.00% |
| 3 | CNN-INC-MEAN-2-HMM | 97.50% | 98.50% |
| 4 | CNN-INC-MEAN-3-HMM | 99.50% | 100.00% |
| 5 | CNN-INC-MEAN-4-HMM | 98.50% | 99.00% |
| 6 | CNN-INC-MEAN-5-HMM | 98.00% | 99.00% |

Table 1 shows the accuracy of six DNN/HMM hybrid models. All the six models are proposed in this research work. It can be noticed from Table 1 that accuracy of method CNN-INC-MEAN-3-HMM is at 99.5% and 100% on ORL-Set 1 and ORL-Set 2 respectively. The hybrid method has the advantages of both sequence based models like HMM and deep learning models. The sequence based models are faster in execution and deep learning models are robust. Though deep learning methods require large computational power, these algorithms can be run faster in a cloud environment. The present research work was developed in Google Colab which is a public cloud for computations.

## CONCLUSION

In this paper, six hybrid models are presented, namely, CNN-HMM, CNN-INC-1-HMM, CNN-INC-MEAN-2-HMM, CNN-INC-MEAN-3-HMM, CNN-INC-MEAN-4-HMM and CNN-INC-MEAN-5-HMM. The models were run on two datasets derived from the ORL dataset, namely, ORL-Set 1 and ORL-Set 2. The performance of the six models were measured on ORL-Set 1 and ORL-Set 2 and compared with the other HMM models like SVD-HMM and LIN-SVD-HMM and deep learning models CNN, CNN-INC-1, CNN-INC-MEAN-2, CNN-INC-MEAN-3, CNN-INC-MEAN-4 and CNN-INC-MEAN-5. It has been observed from the results that hybrid models yielded equal or better results than the HMM models and deep learning models except for CNN-INC-MEAN-4-HMM and CNN-INC-MEAN-5-HMM for ORL-Set 1. However, all the hybrid models performed better than deep learning and HMM models for ORL-Set 2. Of all the hybrid models, CNN-INC-MEAN-3-HMM is the best model with 99.5% and 100% recognition accuracy for ORL-Set 1 and ORL-Set 2 respectively.

## REFERENCES

[1] D. H. Hubel and T. N. Wiesel, "Receptive felds, binocular interaction, and functional architecture in the cat's visual cortex," Te Journal of Physiology, vol. 160, pp. 106–154, 1962.

[2] K. Fukushima, "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unafected by shif in position," Biological Cybernetics, vol. 36, no. 4, pp. 193–202, 1980.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Hafner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2323, 1998.

[4] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," Neural Computation, vol. 1, no. 4, pp. 541–551, 1989.

[5] M. Tygert, J. Bruna, S. Chintala, Y. LeCun, S. Piantino, and A. Szlam, "A mathematical motivation for complex-valued convolutional networks," Neural Computation, vol. 28, no. 5, pp. 815– 825, 2016.

[6] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free? - Weakly-supervised

learning with convolutional neural networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, pp. 685– 694, June 2015.

[7] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15), pp. 1–9, Boston, Mass, USA, June 2015.

[8] Y. L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in Proceedings of the ICML, 2010.

[9] D. Scherer, A. Muller, and S. Behnke, "Evaluation of pooling ¨ operations in convolutional architectures for object recognition," Lecture Notes in Computer Science (including subseries Lecture Notes in Artifcial Intelligence and Lecture Notes in Bioinformatics): Preface, vol. 6354, no. 3, pp. 92–101, 2010.

[10] H. Wu and X. Gu, "Max-Pooling Dropout for Regularization of Convolutional Neural Networks," in Neural Information Processing, vol. 9489 of Lecture Notes in Computer Science, pp. 46– 54, Springer International Publishing, Cham, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in Computer Vision – ECCV 2014, vol. 8691 of Lecture Notes in Computer Science, pp. 346–361, Springer International Publishing, Cham, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in convolutional networks for visual recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1904–1916, 2015.

[13] W. Ouyang, X. Wang, X. Zeng et al., "DeepID-Net: Deformable deep convolutional neural networks for object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, pp. 2403–2412, USA, June 2015.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classifcation with deep convolutional neural networks," in Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12), pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14), pp. 580– 587, Columbus, Ohio, USA, June 2014.

[16] D. Chen, X. Cao, F. Wen, and J. Sun, "Blessing of dimensionality: high-dimensional feature and its efcient compression for face verifcation," in Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13), pp. 3025– 3032, June 2013.

[17] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verifcation," in Proceedings of the 14th IEEE International Conference on Computer Vision (ICCV '13), pp. 3208–3215, December 2013.

[18] T. Berg and P. N. Belhumeur, "Tom-vs-Pete classifers and identity-preserving alignment for face verifcation," in Proceedings of the 23rd British Machine Vision Conference (BMVC '12), pp. 1–11, September 2012.

[19] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, "Bayesian face revisited: a joint formulation," in Computer Vision—ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7–13, 2012, Proceedings, Part III, vol. 7574 of Lecture Notes in Computer Science, pp. 566–579, Springer, Berlin, Germany, 2012.

[20] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," IEEE Transactions on Neural Networks and Learning Systems, vol. 8, no. 1, pp. 98–113, 1997.

[21] X. Wu, R. He, Z. Sun, and T. Tan, A light CNN for deep face representation with noisy labels, https://arxiv.org/abs/1511.02683.

[22] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," in Proceedings of the British Machine Vision Conference 2015, pp. 41.1-41.12, Swansea.

[23] F. Schrof, D. Kalenichenko, and J. Philbin, "FaceNet: a unifed embedding for face recognition and clustering," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15), pp. 815–823, IEEE, Boston, Mass, USA, June 2015.

[24] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: closing the gap to human-level performance in face verifcation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14), pp. 1701–1708, Columbus, Ohio, USA, June 2014.

[25] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[26] Your Paper 1

[27] Your Paper 2