Transfer Learning based Face Recognition using Deep Learning

M. Madhu Latha, K. V. Krishnam Raju

Abstract*: *Face Recognition technology is advancing rapidly with the current developments in computer vision. Face Recognition can be done either in a single image or from a tracked video. The task of face recognition can be improved with the recent surge of interest in the deep learning architectures. In this paper we proposed an Inception-V3 CNN using an approach called Transfer learning. Deep learning algorithms are trained to solve specific tasks and designed to work in isolated. The idea of transfer learning is to overcome isolated learning and leveraging knowledge acquired from one task to solve the related ones. Thus, instead of training CNN from scratch we use Google's pretrained Inception v3 model by applying transfer learning. Inception v3 model was trained on ImageNet dataset and transferred its knowledge to LFW dataset to perform our task face recognition. LFW dataset is a collection of 13000 images of 5749 individuals. Using all the classes result in less accuracy because some of the classes may have single images, so we limited the dataset to 10 individuals with at least 50 images in each class to train the model. We also used a own dataset of 10 individual classes with 100 images for each class.

Index Terms*: *Face Recognition, Deep Learning, Artificial Intelligence, Transfer Learning and Inception v3.

I. INTRODUCTION

Artificial Intelligence (AI) is a flourishing field with many applications and active research topics. The challenging task for artificial intelligence was proved that solving the tasks that are easy for people are difficult to the system and tasks that are difficult to people are easy to the system. AI solves intuitive problems that are like face recognition in images or videos. The solution to these problems was allowing system to learn from experience and understanding hierarchy of concepts, with each concept defined through its relation to other simpler concepts. The gathering of knowledge from experience, is an approach that avoids the need for system operators of formally specifying all the knowledge that the computer needs. The hierarchy of concepts enables the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, the deep learning [1] is an approach to AI.

The AI system requires a ability to acquire their own knowledge, by extracting features from raw data. This ability of acquiring knowledge is known as machine learning. Deep learning also called as a hierarchical learning widespread as a new area of the machine learning research. Deep learning technique is a type of machine learning that gives computer systems an ability to improve with knowledge and data. Deep learning has attained great power and flexibility by representing knowledge.

Nowadays Face Recognition [2] has legitimately broken in to widespread and benefits became more and more tangible. Face Recognition is the eminent biometric technique extensively used in many areas in daily life such as military, finance, public security, etc. Face recognition is the outstanding research topic under the area of computer vision. Computer vision [3] is a scientific field of multidisciplinary that shows how computer systems can be made gain high level understanding from given images or videos. From the engineering perspective, it explores to automate tasks that can be done by the human visual system.

Face recognition has been nurtured by convolutional neural networks (CNN) to a miraculous level. Deep learning CNN has made noteworthy advancement in Image classification. The basic idea behind face recognition is to permit the system with potentiality of detecting human faces fast and precisely in images or videos. Humans can detect faces automatically but coming to the system it was a challenging task. For Face recognition a preprocessed face image is compared with labeled face dataset to determine the similarity between the face images and then recognize the face.

In this paper we use Transfer Learning [4], the basic idea of transfer learning was simple is to grasp the pretrained model on the large dataset and transfer its knowledge to the smaller dataset. The transfer learning technique was done using Google's pretrained Inception v3 model with large ImageNet dataset to perform the face recognition without building and training a CNN from scratch and transfer its knowledge to our task. For face recognition using the transfer learning approach, we retrain the Inception v3 model of TensorFlow on LFW dataset of 10 classes and own dataset of 10 classes. The definition of transfer learning in the context of generalization as follows:

"Situation where what has been learned in one setting is exploited to improve generalization in another setting".

For Convolutional Neural Networks (CNN) we freeze the early convolutional layers of the network and to make predictions we only train the last few layers. The idea behind freezing a convolutional layer is extracting low level features.

Therefore training the inception v3 model on massive dataset called ImageNet .

Hence low level features are shared between the images, LFW dataset images

Revised Manuscript Received on June 01, 2019.

M. Madhu Latha, Department of Computer Science and Engineering, SRKR Engineering College, Bhimavaram, India.
Dr K. V. Krishnam Raju, Department of Computer Science and Engineering, SRKR Engineering College, Bhimavaram, India.

are similar to the ImageNet dataset and the knowledge of the model acquired from ImageNet is transferred to our task.

II. LITERATURE SURVEY

Francois Chollet [5] proposed Xception Net also called as Extreme Inception Net. It is the improved version of the Inception Net. While considering the Inception Net the main idea behind the inception is decoupling the cross-channel correlations and spatial correlations in order to not map them jointly. In the Inception module decoupling of cross-channel correlation using 1x1 convolutions followed by spatial correlations using 3x3 convolutions. The author described Xception Net using depth-wise separable convolution. A depth-wise separable convolution performed using depthwise convolution performed for each channel (3x3 convolution) and a point wise convolution (1×1 convolution).

Anuradha.S.G et al [6] proposed Automated face detection and recognition used for identifying impersonation of candidates in examination system. The process is described in two phases 1.enrollment and 2.authentication. Again the enrollment process is consists of two steps 1.online registration and 2.face detection. In authentication phase image preprocessing is performed by using the Local Binary Patterns(LBP) which is used for converting an image into an array for further processing.

V. Arunkumar et al [7] proposed that by using face recognition the Automatic Teller Machine (ATM) security can be improved. To increase the reliability of ATM transactions the security model of a ATM combine a access card, PIN and face recognition. Illegal user access to ATM can be avoided by checking the user identity using face recognition. The user wearing mask can also be detected for any fault actions alarm rises. Face description done using Local Binary Pattern (LBP). LBP is used for extracting the feature vector and it is further used as a face descriptor. The process described after feature extraction from face images is face verification and recognition.

Liton Chandra Paul et al [8] proposed the face recognition by using a statistical approach called Principal Component Analysis (PCA). It is used in face recognition for reducing the no of variables. In PCA approach, the training set is represented as weighted eigenvectors called Eigen faces for each individual image. These eigenvectors are calculated by using a covariance matrix of a training set. Define the Eigen face space by selecting the highest Eigen values. Recognition is performed by inserting a new image for testing into the eigen-subspace and followed by classification is performed by calculating a minimum Euclidean distance.

Patrik KAMENCAY et al [9] proposed a Convolutional Neural Network (CNN) and evaluated performance between three image recognition methods like Principal Component Analysis (PCA), Local Binary Patterns Histograms (LBPH) and K–Nearest Neighbour (KNN). In this paper the final recognition accuracies of CNN for a considered methods PCA, LBPH, KNN are described. In this paper used ORL database of 40 classes and 10 images for each class. By training and testing the images results are obtained. Based on results the author demonstrated that the LBPH method performs better than PCA and KNN.

Nalini Nagendran et al [10] proposed how the autonomous cars are unlocked safely and securely. Maintain the acknowledged pictures in the picture database by using Support Vector Machine (SVM) classifier. Information from confront pictures through picture pressure using the two-dimensional discrete cosine change transformation (2D-DCT). A self-arranging map (SOM) used an unsupervised learning method is used to order DCT-based element vectors into clusters to distinguish if the picture is present or not present in the picture database. The face detection is done followed by face recognition. If the user identity is authentic then the user can start the ignition of the car and the access is denied to unauthenticated user.

III. RESEARCH METHODOLOGY

The ImageNet [11] large scale visual recognition challenge is a benchmark for object classification and detection on hundreds of classes and images. The challenge has taken place annually from 2010 to till now.

AlexNet is the winning network in 2012 ImageNet competition has been successfully applied to the huge areas of computer vision tasks like object-detection, segmentation, human pose estimation, video classification, object tracking, and super resolution. The success of this network drives into a new era of research that targeted on achieving higher performance of Convolutional Neural Networks (CNN). In 2014, neural network architectures performance enhanced significantly by using deeper and wider network. VGGNet [12] and GoogleNet [13] similarly acquired high performance in 2014 ILSVRC classification challenge. The inception model is represented in figure 1.

This experiment is based on the Inception v3 [14] model of a TensorFlow [15] platform. Inception v3 is one of the trained models on the TensorFlow. It is a rethinking to the initial structure of computer vision after Inception v1, Inception v2 in 2015. The Inception v3 model is trained on the ImageNet datasets, containing the information that can identify 1000 classes in ImageNet. Inception v3 achieved such splendid results by using very deep neural network architecture, including inception modules, and trained on 1.2 million images.

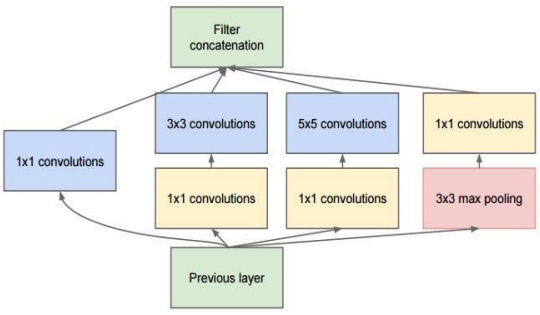

Fig1: Inception module.

Inception-v3 can be determined in to two parts as a feature extraction part with a Convolutional Neural Network (CNN) and

classification part with fully connected and Softmax layers.

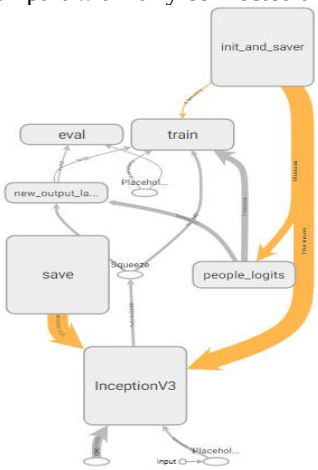

Fig 2: Overall Architecture of Inception-v3 using TensorBoard.

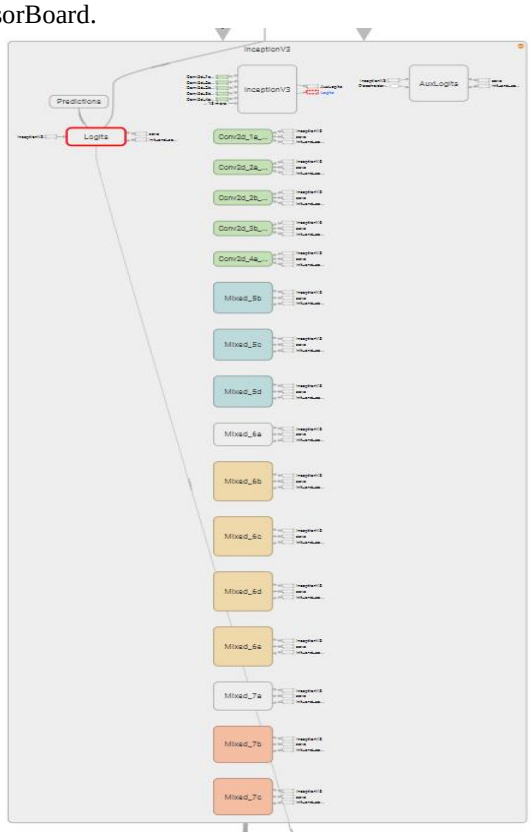

Fig 3: Expanded Inception-v3 node using TensorBoard.

A. Download Anaconda

Firstly we downloaded Anaconda [16] that can be described as a free and open source distribution of Python and R language in to our system with a processor of Intel i5, memory 8GB, with system type of 64-bit Operating System and x64 processor. It was widely used in research areas like, data science and applications of machine learning. Anaconda include more than 1400 packages, Anaconda Navigator which is virtual environment manager and Conda package. From a Anaconda repository, we can install any number of open source packages by using commands. Anaconda Navigator include many applications but we used Jupyter Notebook for our experiment.

B. Install TensorFlow

TensorFlow [17] is the open source framework for performing research in data science. We open the Anaconda repository and install TensorFlow packages by using the command “pip install tensorflow”.

C. Download Inception-V3 Model

In our experiment, we first have downloaded the latest version checkpoint of the pre-trained Inception model. Download a inception v3 model of inception v3 2016 08 28.tar.gz and create a directory to save the checkpoint of the model.

D. Dataset Collection

For any data science projects fetching the appropriate dataset is the crucial aspect for achieving better performances. In our experiment of facial recognition we have taken 10 individual classes of Labeled Faces in the Wild (LFW) [18] dataset because some of the classes of LFW dataset are associated with single images still now there no powerful CNN for making predictions on single images. In ideal case each class can have thousands of images, but our system cannot handle those many computations. We also taken a student database of 10 classes from our college with each class having exactly 100 images. We have taken equal no of images because no of images in one class varies largely with the other class of 10 classes in LFW dataset. Now from Anaconda repository open Jupyter Notebook using python language we have downloaded the Labeled Faces in the Wild full dataset and stored in a directory. We have adopted the process of doing images deep-funneled, it is a process of aligning images this results in better representation of the dissimilarities between face images by reducing the variability in the background of images. The one of the most problems of recognizing a face in images is a intra class variability between images of the same individual can often be greater than inter class variability between images of different individuals. The proposed deep-funneled method has been expressed the improved accuracy on the task of facial identification compared to un-aligned face images.

Fig 4: Shows face images of LFW dataset.

E. Datasets Labelling

For our experiment, we have taken two datasets and label them by sorting an individual classes by the number of face images in the dataset and move them to a new directory in a hierarchy. Because of this, we can rapidly create the labels and images of training, validation and testing datasets.

```
{9: 'Tony_Blair',
 6: 'Gerhard_Schroeder',
 8: 'Colin_Powell',
 4: 'Hugo_Chavez',
 7: 'Donald_Rumsfeld',
 3: 'Junichiro_Koizumi',
 1: 'George_W_Bush',
 2: 'Jean_Chretien',
 0: 'John_Ashcroft',
 5: 'Ariel_Sharon'}
```

```
{0: ' sumathi',
 1: ' bhargavi',
 2: ' deepu',
 3: ' hema',
 4: '  jwala',
 5: ' kavitha',
 6: ' keerthi',
 7: ' lakshmi',
 8: ' madhu',
 9: ' srinu'}
```

Fig5: Shows a dictionary to map integer labels to classes of our both datasets.

```
{'9Tony_Blair': 144,
 '6Gerhard_Schroeder': 109,
 '8Colin_Powell': 236,
 '4Hugo_Chavez': 71,
 '7Donald_Rumsfeld': 121,
 '3Junichiro_Koizumi': 60,
 '1George_W_Bush': 530,
 '2Jean_Chretien': 55,
 '0John_Ashcroft': 53,
 '5Ariel_Sharon': 77}
```

```
{'0 sumathi': 100,
 '1 bhargavi': 100,
 '2 deepu': 100,
 '3 hema': 100,
 '4  jwala': 100,
 '5 kavitha': 100,
 '6 keerthi': 100,
 '7 lakshmi': 100,
 '8 madhu': 100,
 '9 srinu': 100}
```

Fig6: Shows no of images for each class in our datasets.

```
{'9Tony_Blair': 9.89010989010989,
 '6Gerhard_Schroeder': 7.486263736263736,
 '8Colin_Powell': 16.208791208791208,
 '4Hugo_Chavez': 4.876373626373627,
 '7Donald_Rumsfeld': 8.31043956043956,
 '3Junichiro_Koizumi': 4.1208791208791204,
 '1George_W_Bush': 36.4010989010989,
 '2Jean_Chretien': 3.7774725274725274,
 '0John_Ashcroft': 3.6401098901098905,
 '5Ariel_Sharon': 5.288461538461538}
```

```
{'0 sumathi': 10.0,
 '1 bhargavi': 10.0,
 '2 deepu': 10.0,
 '3 hema': 10.0,
 '4  jwala': 10.0,
 '5 kavitha': 10.0,
 '6 keerthi': 10.0,
 '7 lakshmi': 10.0,
 '8 madhu': 10.0,
 '9 srinu': 10.0}
```

Fig7:Shows the percentage calculation for composition of our datasets.

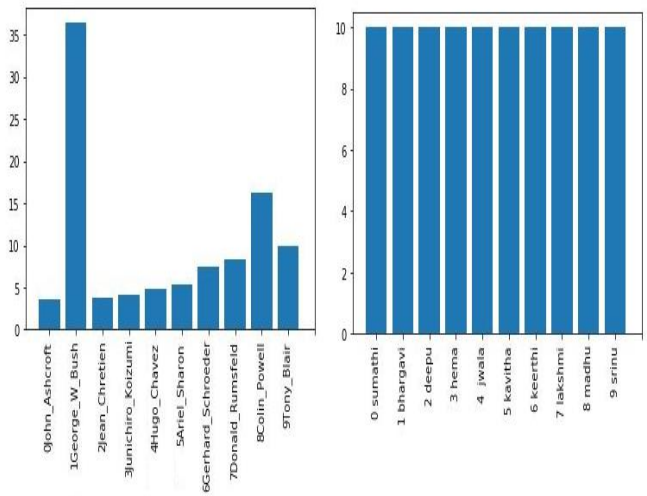

Fig 8: Shows bar graph visualization of our datasets composition.

F. Designing And Splitting Datasets As Training, Validation And Testing Sets

In the world of a machine learning the problem was it requires large scale datasets for performing image classification using CNN. But, we have a very small amount of data on some of the individuals and the 53 image's class was least among the classes. But, we can use a technique like data augmentation to artificially enlarge the dataset. We can also observe that our LFW dataset has uneven distributions of classes. By comparing a George W Bush and Colin Powell the difference is more than half of the images between them this may vary the performance between classes. Because of this reason we have taken own student dataset with equal no of images in all the individual classes.

Now we have 1456 face images of 10 different individual classes are labeled properly with a number of images and names of the classes. Due to this labeling we can create the datasets as training, validation, and testing sets. The later step we had done is splitting the datasets into training, validation and testing. I want to maintain as many images as possible as training data, but we also need a fair size of validation set to implement early stopping in order to prevent over fitting on the training data. The testing set should also be a reasonable amount of size in order to determine a correct accuracy of a trained model. We maintain 70% data of each class for a training set, %5 of data for a validation set, and we retain remaining 25% of data for a testing set. Finally, the no of images in each data split was 1027 for training, 61 for validate and 361 for testing.

We can determine this entire process as data wrangling and is now completed properly. So we have the apt training, validation, and testing images and labeled datasets. While running the training dataset we will use the validation set for early stopping implementation. For running the training dataset we through one small batch at a time in to the model and then test the validation loss at each epoch with the full validation set. The validation set is therefore limited by the number of input images through the network at one time.

The test set is used to determine the accuracy we will avail a small batch size and iterate through the set in order to test on all the instances.

G. *Preprocess Images For Using In Inception CNN*

We have to preprocess images for using in Inception CNN that has been trained on images that are 299 x 299 x 3 (height x width x 3 color channels) where the single value in each individual color channel is the pixel intensity of that color at that channel location. The range of Color intensities are from 0-255.The inputs that are taken by the Inception CNN are normalized between 0 and 1 by simply dividing the raw values by 255. The face images of our experiment are an array size of 255 x 255 x 3 we have to resize in order to make image inputs appropriate to Inception CNN and the values normalized. We will input these processed images during the step of a training process.

H. Determining Logits, Endpoints, Predictions And Output Layer

In our experiment, we are using the transfer learning method []. This method retains the parameters of a

previous layers and removes the final layer of Inception-v3 [] model. For performing the tasks we again retrain a final layer.

While defining a Tensor Flow model, we cannot input the data directly to our model. We should create a placeholder that acts like a starting point for our input data.

The next step is finding out what is the final trainable layer of the network that was obtained before outputs. Unfreeze the final layer of the network and use those parameters to learn the new classes. For obtaining the logits and the endpoints we had used Inception-v3 function. The logits can be defined as un scaled outputs obtained from the neural network. While coming to the end points a dictionary of values returned from various stages of a final layer of the network. By

considering end points of a dictionary we can classify the precise level of the network that we want to train.

PreLogits are the neurons that we want to train and they are obtained right before the predictions. After obtaining the PreLogits, the major activity of transfer learning method that we had performed was unfreezing the PreLogits layer. From all of the prior layers will retain the weights and biases that they had learned on the 1.2 million images of ImageNet and on which Google trained the Inception-v3. For our task of recognizing the faces new weights and biases are only learned by the new single layer.

Actually our current predictions obtained are ('Predictions':< tf.Tensor 'InceptionV3/ Predictions/ Reshape 1:0' shape= (?, 1001) dtype=float32>}) have a shape of (?, 1001) indicates an 1000 classes in the ImageNet data and the background class that is created when the argument is training was True. With the retained predictions we create a new output layer.

Define function to create batches of processed images. The function looks like it correctly process the images and returned the batch as we specified. We can use this function when training to iterate through the training and testing set respectively. We process the entire validation set ahead of time because we will be passing all the validation examples through the network at once during training.

I. *File Writing Operations To Tensor Board*

TensorBoard[19] is a visualization tool used for understanding and debugging tensor flow programs. TensorBoard can be operated by reading the summary data from the tensor flow files In our experiment, we used this tool to visualize graphs and also we want to visualize the training accuracy, and validation accuracy and if we are over fitting or not can be determined by the loss and to gauge the progress over the iterations. We can also use TensorBoard tool to analyze the structure of the network in order to do so, we need to specify the files to track and write to the File Writer at appropriate times. This can be done by using tensorflow summary operation like "tf.summary.FileWriter".

J. *Training And Validating Dataset*

For performing training we restore all the parameters of the entire original pre-trained inception-v3 CNN and only the final layer of neurons(PreLogits) that we obtained by prediction which learns the highest level features are trained.. This can be done using saver function before modifying the network. The tensorflow command is "inception_saver.restore()". Then mention the epoch count so that each epoch will iterate through all the instances during training. After that train the trainable layer in order to adjust the new weights and biases of the trainable layer to learn the new classes of our task. Save the trained model for using it on the test set for performance evaluation. For later visualization of training and to display the progress we write the summary of the training to the file writer of the TensorBoard. For doing this provide the directory where checkpoints are stored to the file writer using tensorflow command "tf.summary.FileWriter()" and add model summary to file writer using command "filewriter.addsummary()".

K. *Test Dataset*

The testing set performance can be evaluated by using our new pre trained model. For doing this first we should save our pre trained model using a command "saver. restore (new trained path)". We created a dictionary for storing the accuracies. As we cannot feed all the images that are to be tested through the model at once so that we iterate through the full test set one batch at a time. We calculate the final accuracy by performing mean of all the batch accuracies.

L. *Data Augmentation*

As we know that data science project requires large amount of data for better performance. Our dataset size was not relatively large, but we can do data augmentation for increasing the dataset size. The aim of the data augmentation [20] is to better the performance of accuracy by augmenting the dataset. There are many approaches to increment the amount of data. In our experiment we expand the dataset artificially by applying transformations to the images that should not affect the feature's content in the image. If two images of faces are the same, then our algorithm should be able identify that they represent the same individual rather than predict that an individuals are different because they were oriented differently. Therefore, our network should be invariant to certain shifts of the images. In effect, we are creating a

network that is robust to manipulations of the orientation of the images and only learns to differentiate the content in the faces themselves. We can train the network from where it left off on the unaugmented dataset. We will try to retrain from scratch first to compare the accuracy we can achieve.

- *Shifting Images*

The first transformation will be to shift the image by 30 pixels in each direction. This will create four additional images for each original image, effectively making the dataset 5x bigger. This shift should help to make the algorithm robust to changes at the location of a face within the image. Tensorflow has built in functions to augment data, particularly images, but we can also use other libraries so we do not have to run the process in a Tensorflow session. For the shift transformation, the image will be shifted by the specified [dx, dy] and then the overlapping pixels will be set to a value of 255. Looking at the plotted images, I would say that shifts have been successful. It may not seem like a large change, but at the very least, we now have a dataset that is five times larger. Perhaps it is better to shift the images by more or less, or to do shifts in two directions at

once, but for now, we will start with the four simple left, right, down, up shifts picture below.

Fig 9: Shows shifting of original image in to four shifts.

M. Training And Validating Sets From Unaugmented Checkpoint

Another strategy we can use is to start off the training with the augmented dataset from the endpoint of a unaugmented training. What we are doing here is hoping that the same weights learned for one trainable layer during the unaugmented training will be useful on the augmented set as well. Ideally, this could speed up training (fewer epochs will be required for convergence on an optimal set of parameters) and it could result in an overall better network. Again, we will create different TensorBoard File Writer directories and save locations so we can visualize the evolution of training cycles.

N. *Testing Set*

The testing set performance can be evaluated by using a trained set. For doing this first we should save our unaugmented training path using a command "saver. restore(trained path)". We created a dictionary for storing the accuracies. As we cannot feed all the images that are to be tested through the model at once so that we iterate through the full test set one batch at a time. We calculate the final accuracy by performing mean of all the batch accuracies.

IV. RESULTS

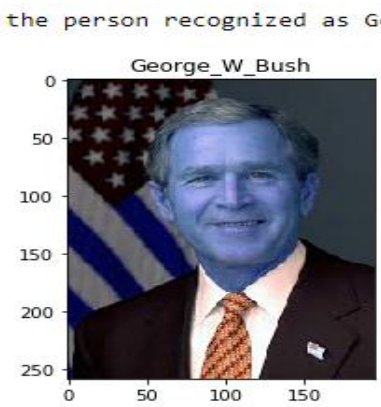

Fig 10: Shows that our face recognition system correctly recognized that the given person image as George W Bush.

```
Predictions:
George_W_Bush          : 97.63%
Colin_Powell           : 0.99%
Ariel_Sharon           : 0.85%
Tony_Blair             : 0.34%
Jean_Chretien          : 0.10%
Gerhard_Schroeder      : 0.08%
Donald_Rumsfeld        : 0.01%
Hugo_Chavez            : 0.00%
Junichiro_Koizumi      : 0.00%
John_Ashcroft          : 0.00%
```

Fig11: Shows that as the predicted accuracy for George_W_Bush was 97% which was high compared to other persons. so that the system recognized the person as george_w _bush.

Classes	Accuracies
0 John_Ashcroft	76%
1 George_W_Bush	97%
2 Jean_Chretien	77%
3 Junichiro_Koizumi	78%
4 Hugo_Chavez	81%
5 Ariel_Sharon	82%
6 Gerhard_Schroeder	84%
7 Donald_Rumsfeld	88%
8 Colin_Powell	96%
9 Tony_Blair	92%

Table1: Shows the predicted accuracies of taken 10 sample classes of LFW dataset.

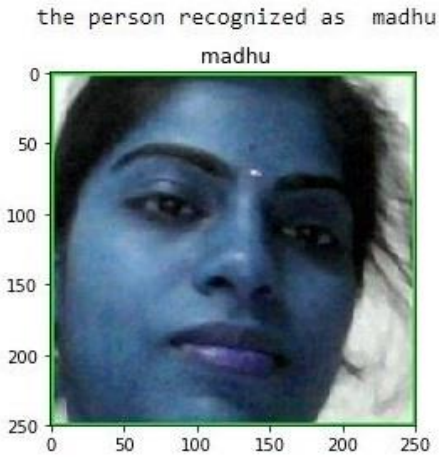

Fig12: Shows that our face recognition system correctly recognized that the given person image as madhu

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

```
Predictions:
madhu                    : 99.95%
keerthi                  : 0.02%
lakshmi                  : 0.02%
sumathi                  : 0.01%
srinu                    : 0.00%
deepu                    : 0.00%
bhargavi                 : 0.00%
jwala                    : 0.00%
hema                     : 0.00%
kavitha                  : 0.00%
```

Fig13: Shows that as the predicted accuracy for madhu was 99% which was high compared to other persons. So that the system recognized image of a person as madhu.

Classes	Accuracies
0 Sumathi	96%
1 Bhargavi	99%
2 Deepu	95%
3 Hema	99%
4 Jwala	99%
5 Kavitha	96%
6 Keerthi	96%
7 Lakshmi	97%
8 Madhu	99%
9 Srinu	99%

Table 2: Shows the predicted accuracies of taken 10 sample classes of own dataset.

V. FUTURE WORK

The face recognition classification approach that we have developed using Google's Inception V3 model has given us splendid accuracy. The future work for the proposed work is developing application to the face recognition like attendance management system using video surveillance.

VI. CONCLUSION

In this paper we proposed the face recognition system using Inception-V3 model of tensorflow platform based on transfer learning approach to train CNN model. After performing all experiments the results obtained from our proposed system recognized the human face images well. Hopefully in future we can improve this method and achieve greater accuracy.

REFERENCES

1. Ian Goodfellow, Yoshua Bengio and Aaron Courville, accompanied by the website www.deeplearningbook.org, pp. 96-365, 422, 438-473.
2. Josh Patterson and Adam Gibson, Deep Learning A Practitioner's Approach, published by O'Relly media,Inc, pp. 1-141,293-304.
3. Li Deng and Dong Yu, Deep Leaening: Methods and Applications, published as Foundations and Trends in Signal Processing, Volume 7, Issues 3-4, ISSN: 1932-8346, pp. 320-324.
4. Chuanqi Tan et al, A Survey on Deep Transfer Learning, arXiv: 1808.01974v1, 2016.
5. Francois Chollet Google, Xception: Deep Learning with Depthwise Separable Convolutions, arXiv: 1610.02357v3, 2017.
6. Anuradha.S.G, Kavya.B, Akshatha.S, Kothapalli Jyothi, Gudipati Ashalatha, Automated Face Detection & Recognition for Detecting Impersonation of Candidate in Examination System, International Journal of Scientific & Engineering Research, Volume 7, Issue 3, ISSN 2229-5518, March-2016.
7. Arunkumar, Vasanth kumar, naveenly king, Aravindan, ATM Security Using Face Recognition, International Journal of Current Engineering and Scientific Research (IJCESR), ISSN: 2394-0697, VOLUME-5, ISSUE-4, 2018.
8. Liton Chandra Paul, Abdulla Al Sumam, Face Recognition Using Principal Component Analysis Method, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), ISSN: 2278-1323, Volume 1, Issue 9, November 2012.
9. Patrik Kamencay, Miroslav Benco, Tomas Mizdos, Roman Radil, A New Method for Face Recognition Using Convolutional Neural Network, Digital Image Processing And Computer Graphics, DOI: 10.15598, VOLUME: 15, SPECIAL ISSUE, 2017.
10. Nalini Nagendran, Ashwini Kolhe, Security and Safety With Facial Recognition Feature for Next Generation Automobiles, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7 Issue-4S, November 2018.
11. Olga Russakovsky et al, ImageNet Large Scale Visual Recognition Challenge, arXiv: 1409.0575v3, 2015.
12. Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks For Large-Scale Image Recognition, Published as a conference paper at ICLR, arXiv: 1409.1556v6, 2015.
13. Christian Szegedy et al, Going deeper with convolutions, arXiv: 1409.4842v1, 2014.
14. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, et al. Rethinking the Inception Architecture for Computer Vision, arXiv:1512.00567v3, 2015.
15. Mart´ın Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv:1603.04467v2, 2016, Available: www.tensorflow.org.
16. Anaconda software Available: https://www.anaconda.com/download.
17. Google Brain et al. TensorFlow: A System for Large-Scale Machine Learning. In proceedings of the twelfth USENIX Symposium on Operating Systems Design and Implementation (OSDI '16). ISBN 978-1-931971-33-1, 2016, pp. 265-267,277.
18. LFW dataset Available:
19. Indra den Bakker, Python Deep Learning Cookbook, published by packt publishing Ltd, ISBN 978-1-78712-519-3, pp. 265-278,365,422.
20. Navin Kumar Manaswi, Deep Learning With Applications Using Python, Available: https://doi.org/10.1007/978-1-4842-3516-4, pp. 171-197.

AUTHORS PROFILE

M. Madhu Latha completed B.Tech in Computer Science and Engineering from Jawaharlal Nehru Technological University Kakinada, India. She is currently pursuing M.Tech in Computer Science and Engineering from Jawaharlal Nehru Technological University Kakinada, India.

K.V.Krishnam Raju received his B.Tech and M.Tech in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad, India and Nagarjuna University, India. He also received Ph.D degree from Jawaharlal Nehru Technological University Kakinada, India. He is currently working as associate professor in the department of computer science and engineering in SRKR engineering college.