Proceedings of the SMART–2019, IEEE Conference ID: 46866
8th International Conference on System Modeling & Advancement in Research Trends, 22nd–23rd November, 2019
College of Computing Sciences & Information Technology, Teerthanker Mahaveer University, Moradabad, India

Bengali Ethnicity Recognition and Gender Classification Using CNN & Transfer Learning

Md. Jewel[1], Md. Ismail Hossain[2] and Tamanna Haider Tonni[3]
[1,2,3]*Dept. of CSE, Daffodil International University Dhaka, Bangladesh*
E-mail: [1]jewel15-8071@diu.edu.bd, [2]ismail15-7838@diu.edu.bd,
[3]tonni15-654@diu.edu.bd

***Abstract*— In this paper, we have demonstrated how to apply CNN (Convolutional Neural Network) structured model and transfer learning to identify the ethnicity of Bengali people and it's a systematic process of gender classification too. We also applied several models of transfer learning like VGG16, Mobilenet, Resnet50, etc. to find out which model is more convenient to get our desired accuracy. But problems arise because there are many Indian people who look like and get dressed up like Bengali since in India many Bengali dwell in when many of them speak Bangla as well! (people of Kolkata along with some other provinces). So, the Bengali people are not only found in Bangladesh but also elsewhere in the world. That's why our model is based on facial images along with the tradition of their costumes. We tried to build a sophisticated model using CNN and transfer learning for this purpose and we got some tremendous performances applying transfer learning.**

Keywords: Data Augmentation, Ethnicity Recognition, Gender Classification, Convolutional Neural Network, Transfer Learning, Fine-Tuning, Deep Learning, Bottleneck Features

I. Introduction

Gender classification and ethnicity recognition have got attention in the research of computer vision recently. There are a handful of works on gender classification and ethnicity recognition from facial images using CNN (Convolutional Neural Network) and a few works have been accomplished using transfer learning where most of them do not include Bottleneck features. So, it must be figured out. Another reason is that, while classifying gender or ethnicity it varies from nation to nation where the tradition of their costumes plays a major role. In this paper, we are focusing on the Bengalis gender classification and ethnicity recognition applying CNN followed by several transfer learning models.

So, how are we going to do this? There are differences between male and female facial images where the clothing will definitely cause occlusion because clothing varies from gender to gender mostly in the Bengali people. Moreover, the hairstyle is another factor and hair length is more confusing. For all of these research CNN has been applied widely such as in pattern recognition [01], face detection [02], and action recognition [03].

Our proposed CNN model is a straightforward architecture along with the application of bottleneck features. In spite of having simplicity in our CNN model, we got competitive performance in it and applying transfer learning we achieved even more outstanding performance.

The arrangement of this paper goes as follows. In section 2, the related work is introduced, in section 3, the dataset is used, in section 4, the pre-processing of the dataset and our primary CNN model is explained (to estimate the number of nodes, etc.), in section 5, the transfer learning models are applied along with the proposed CNN model and at last, in section 6, the conclusion is drawn.

II. Related Work

Based on CNN most of the papers that have been published so far are basically for the prediction of age or gender but on ethnicity, there are only a few papers [04] available. A detailed age classification methods have also been described in those literature review papers [05], [06], [07], [08].

Actually, gender classification using images is thought to be a very influential piece of work using CNN because its uses are found all around the globe to classify gender using images and their ethnicity as well. Many papers have been written for a deep analyzation and we can get their ideas and approaches from these references [9]. Golomb et al. [10] were one of those researchers who applied CNN on a small dataset of facial images to classify gender. Yang and Moghaddam had also used SVM (Support-Vector Machine) [11] to classify gender with their best accord and Baluja and Rowley [12] adopted AdaBoost for implementing the same scheme from facial images. Toews and Arbel [13] also showed a lot of viewpoints for presenting a model that could extract many features to classify human gender and age. Gender classification can be classified from facial features, Wang et al. used 19 features [14]. Apart from these, Ekman et al. [15] distinguished two methods to study facial features. They were "message-based" and "sign-based". We should go through some basic definitions to understand this paper better, some important definitions have been described here,

A. Deep Convolutional Neural Network

CNN [16], Convolutional Neural Network is formed by one or many input layers and hidden layers with one or many output layers. Actually, nowadays CNN has been used for remarkable strategies to extract features from images and in visual recognition [17]. After extracting features, a few subsequent layers combine them and finally, the feature maps are encoded in a 1D vector where they are categorized. Generally, we do downsampling in our model nodes till the last layer and in the last layer, our dense layer should be the number of nodes equal to the number of our initial classes. The neural network is trained by a backpropagation algorithm [18]. The equational representation is [19]:

$$C_j = \sigma (\Sigma W_{i,j} \otimes I_i + b_j) \quad (1)$$

This figure (Fig.1), shows the representation of the equation (1) where Cj and bj are the corresponding trainable bias, $W_{i,j}$ is equivalent to the size of the filters (n×m), I_i, is the feature map from the previous layer, σ introduces to non-linearities. Where σ is either sigmoid, $\sigma(x) = 1/(1+e^{-x})$ or hyperbolic tangent function, $\sigma(x) = \tanh(x)$.

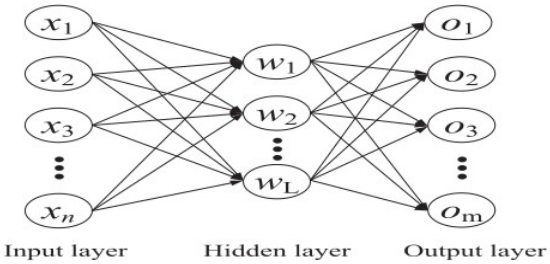

Fig. 1: Neural Network

Here, an image (as input) is fed into the input layer (Fig.1) then it gets passed through the hidden layers where the features get extracted. Finally, the output layer predicts the best probability [20].

III. Dataset

We collected more than 4,300 images of Bengali people (including Indian Bengali) of different ages from Google but we made sure the good quality of the pictures. We also ensured that the images were not racially mixed because it may cause a disaster for any CNN model. After collecting images the most tedious part was to label those images which means putting the similar type of images into the same folder so that we can apply CNN. Apart from this, sometimes there may present many irrelevant photos too and we had taken care of all of these to make our mission smoother.

IV. Preprocessing

We have got the data, but we can't feed these raw images right through our CNN network. First, we need to resize these images into the same size, and then we may want to convert them into grayscale. Apparently, the labels of 'Bengali Male' and 'Bengali Female' are not useful, we want to convert them into one hot array. Fig. 2 is a random sample chosen from the Bengali Female folder. After converting the images into grayscale, it looks like Fig. 2,

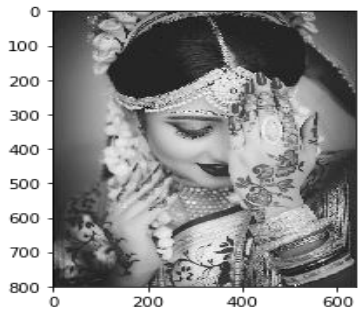

Fig. 2: In Grayscale View (Clear Image)

Now to scale this, we had to select such a size for these images, so that we could recognize the same image (an example is given in Fig.3) as a Bengali female (or male). We fixed the image shape to 99 which means, the shape of the images will be 99x99 during training.

Image shape: 99x99; So, this image is not so clear as before but a human can easily recognize.

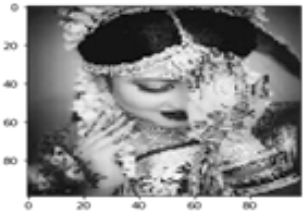

Fig. 3: After resizing (Hazy Image)

At the very beginning, we wanted to see how many dense layers, number of nodes and number of convolution layers would work fine and suitable for our proposed model. So, primarily we used the number of dense layers = { 0, 1, 2 }, layer sizes = { 32, 64, 128 } and convolutional layers = { 1, 2, 3 } that means, all and every possible combination from these numbers were run to see how better the new and primary model would work and how far we were from our destination.

We used 'ReLu' (Rectified Linear Unit) as the activation function in this primary model and Sigmoid in the last layer, for validation we kept 20% data. We used 'Adam' optimizer and binary cross-entropy as the loss function. 2 max-pooling of 2D layers were 2x2 in size. After running for 20 epochs, in the batch size of 25, we got around 66% accuracy. That was not bad at all since we did not use any dropout layers or any other advanced processes and functions. The model was run for all the 27 (3x3x3=27) combinations where every single combination had 20 epochs. As a result, it took a bit long time for training but in the consequences, we got a clear view of our model and we understood that our model was going to perform better for

more nodes because for the size of 128x3x1 the model gave that accuracy, but the problem was in the loss function. Our validation loss was getting increased. We set and saved the history and we used Tensorboard to visualize the whole process.

The epoch validation accuracy is given here (Fig.4) for all the possible combinations (27) we have run so far.

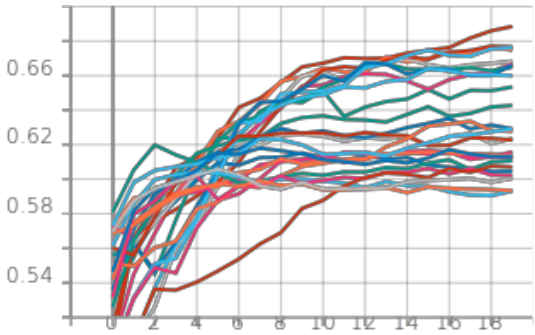

Fig. 4: Epoch Validation Accuracy

and in the second figure (Fig.5) from Tensorboard, the epoch validation loss is showed. Here it's very clear that the model is not working fine since it has totally become a case of overfitting (curves are going upwards). Now we will try to reduce this overfitting gradually.

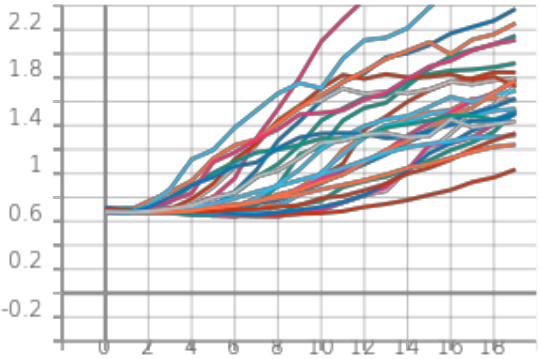

Fig. 5: Epoch Validation Loss

However, we have achieved the base of our primary CNN model.

V. Transfer Learning

Getting quality accuracy during the training of image classification is not a simple task. Because to do so, we need a lot of training data. So, in real life collecting a dataset is not good enough for deep learning because we may need millions of images to build a powerful network. But generally, it's troublesome and the solution is transfer learning, we can use a pre-trained model that was actually trained in a large dataset. The idea is that the trained model has already learned many features from a huge dataset and it is possible to use these features as a base of learning for new classification problems. We applied Resnet50, Mobilenet, etc. models.

B. Resnet50

Resnet50 is a trained network on millions of images from the 'Imagenet' database. It has many layers (around 50) which are used to classify images in more than 1,000 categories. So, to use the Resnet50 model, we used 'Imagenet' for our weights and then we set the top layer false which means, we will not keep the FC (fully connected) layers at the end of the model. For this, we resized the height and width of our image by 200x200 when our base model was Resnet50 and we used 3 channels, not like the grayscale images. After running for 20 epochs we achieved a fine accuracy of 85% when the loss was only 0.33.

C. Mobilenet

Mobilenet is such an architecture that is more suitable for different types of computer vision applications especially when there is a lack of enough computational power. The Mobilenet architecture was proposed by Google and nowadays it has been popularly used in computer vision.

To use Mobilenet our base model was 'Mobilenet' and we again used here the weights of 'Imagenet' when the top layer was not included. That means, discarding the last 1,000 layers. As we have seen our model performs better for more nodes so, we have used this time a dense layer of 1,024 nodes and the activation function was 'Relu'. We added dense layers so that the model can learn more complex functions. Then we added the second dense layer of the same size and the third dense layer of 512 nodes and our final dense layer consisted of 2 nodes where the activation function was 'Softmax' (final layer of softmax activation). We used the 'ImageDataGenerator' class and included all dependencies. We fixed the image size to 224x224, 'RGB' (Red, Green, and Blue) as color mode and the class mode was categorical when we set the shuffle of the dataset to true. Apart from this, we used the 'Adam' optimizer, loss function was categorical cross-entropy and the evaluation metric was 'Accuracy'. We got around 96% accuracy applying this model which was expected.

D. MobileNetV2

As we had got a high accuracy in Mobilenet so we considered it would be great to try with MobilenetV2 (version 2 of Mobilenet). So, we applied the MobilenetV2 to maximize our accuracy and minimize our loss. We resized the images by 160x160 and we had set it for the batch size of 32. To apply data augmentation we rescaled images by 1/255 and our class mode was binary this time and we also used 3 channels when our base model was MobileNetV2, which will be used as a pre-trained model of training. We again used the 'Imagenet' for weights and also added the learning rate of 0.0001, the metric was 'Accuracy' and loss was binary cross-entropy. After running for 10 epochs we got the validation accuracy of 85% and validation loss of

34%. We saved the history and displayed it in the training and validation accuracy figure and in the training and validation loss figure (Fig.6). From Fig.6, it's very evident and identical that the loss is decreasing gradually which was expectable since the model is overcoming the problem of overfitting.

Fig .6: MobileNetV2

Now from this stage, we tested furthermore for trying fine-tuning. We set the fine-tune value to 100 which means fine-tune is valid from this layer to onwards. So then, we had to make frozen all the previous layers and we also set these layers trainable value to false, so that they are not trained during the training. Then using Keras we set the 'RMSprop' (a gradient-based optimization technique, learning rate = 2e-5), binary cross-entropy for loss and 'Accuracy' was set for metrics. After running for further 10 epochs, over the last weights we got better accuracy. It's more obvious that the training loss is decreasing which was our concern and target.

So far we've applied 3 transfer learning pre-trained models, their summary is given in table 1,

TABLE 1: MODELS SUMMARY

Model	Epochs	Image- Shape	Validation Accuracy
Resnet50	20	200x200	85%
Mobilenet	15	224x224	96%
MobilenetV2	10	160x160	85%

E. Bottleneck Features in Keras and Tensorflow (for VGG16)

This process has been more simplified because we have already available some pre-trained models like VGG16 and their pre-trained weights. Keras has these in the Keras applications package. The core idea of transfer learning is to use the pre-trained model with their weights and at the same time removing the final FC layers from that model. After that, we have to use the remaining part of the model as our own model's features which will be applied in our dataset. These newly extracted features are often called as the Bottleneck Features. Keras blog has a well-guided document in image classification using this method. So now we are going to build a CNN model for classifying 2 categories of Bengali people like Bengali female and Bengali male. We have our dataset ready for this execution and we shall use the ImageDataGenerator along with the flow_from_directory() functionality of Keras. We have already created the folder called Bengali female and Bengali male where there are respective photos inside the folder. So, we will actually pass through these procedures,

- Firstly, we shall have to save the bottleneck features of the VGG16 model.
- Secondly, we shall train our network using the pre-saved features of Bottleneck so that we can classify images (this is also known as Top model).
- Finally, to make a prediction we will use both of the models (VGG16 and Top model).

1). Proposed CNN Model

But before implementing this operation of Bottleneck features, we immediately need the features. So, we are going to build our proposed CNN model first, then we shall use these features (collected from the proposed CNN model) as Bottleneck features. For this time, we shall also apply the data augmentation. Let's go through the definition first,

F. Adding Data Augmentation

To explain in brief, data augmentation is an efficient process that increases the diversity in the dataset significantly without collecting or using more data. For this purpose, there are many techniques available such as cropping, flipping, padding, scaling, shearing, zooming, etc. These techniques are often used widely in neural networks.

We also used dropout to build this CNN model. Because dropout and data augmentation can reduce the problem of overfitting. So, we shaped the images to 150x150 and used 3 conv2D (3x3) layers (32, 32, 64 nodes respectively) in our sequential model. We also added 2 dense layers (size, 64 and 1), 3 max-pooling of 2x2 size, 1 flatten layer, 'Relu' as the activation function and the rate of the dropout was 0.5 when our final activation function was 'Sigmoid'. Binary cross-entropy was set as loss, the optimizer was 'RMSprop' and 'Accuracy' was the metric as usual. But we added data augmentation this time of shear_range 0.2, zoom_range of 0.2 and finally, the horizontal flip was set to true. We got around 83% (epochs = 50) accuracy for this proposed model. From Fig.7, the loss and accuracy look absolutely good (no overfitting).

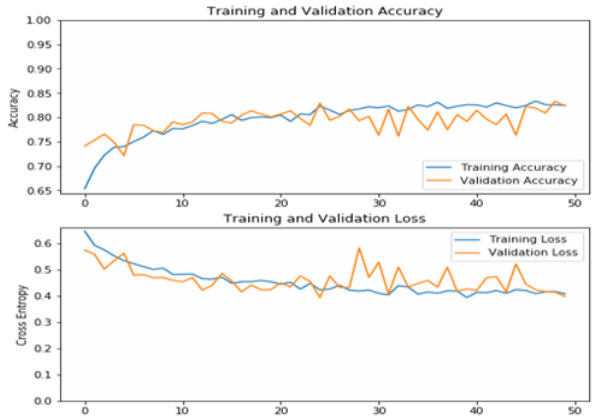

Fig. 7: Applying CNN

Now we shall implement the technique of Bottleneck features because we have already got the result from our proposed CNN model and we have the saved bottleneck features. Now we may use it for further processing. The features were saved in a .npy (extension) file so now, we may load them for use. We resized the image by 224x224 with the batch size of 16, dropout 0.5 when the activation function of the last layer was 'Sigmoid', 2 dense layers of 256 nodes and the number of classes were 2 for our dataset. Then one flatten layer was added in our sequential model, 'RMSprop' was the optimizer and loss was categorical cross-entropy with the 'Accuracy' metrics. After running this model for 50 epochs we got around 85% accuracy. This figure (Fig.8) shows the accuracy of our model which seems good but the second figure (Fig.9) of the model's loss seems getting overfitted (after 20 epochs). However, we have implemented our proposed bottleneck features model to classify images, which has given a decent validation accuracy.

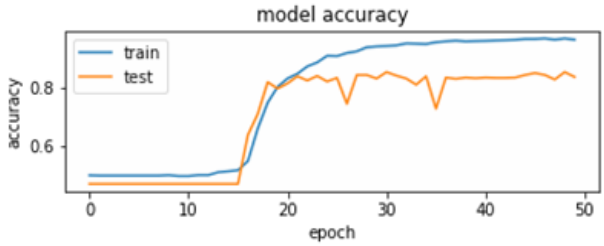

Fig. 8: Accuracy of Bottleneck Features

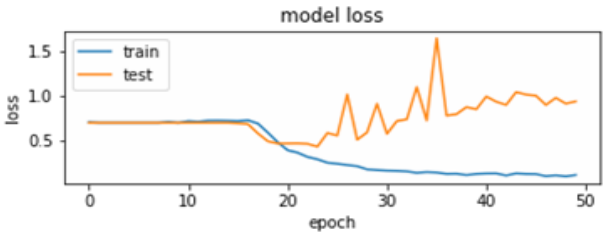

Fig. 9: Loss of Bottleneck Features

2). Keras Fine-tuning with VGG16

(*i) Freeze all layers*: After loading the VGG16 model with the weights of 'Imagenet', we made all the trainable layers frozen. Then for the convolutional base model, we loaded the model in our sequential model. Next, we added 1 flatten layer, 1 dense layer of 1024 nodes with the activation function 'Relu' and then we added a dropout layer of rate 0.5 and final dense layer of size 2.

So firstly, we tried without data augmentation with categorical cross-entropy as loss and 'Acc' was metric, the training batch size was 100 where the validation batch size was 10. After running for 20 epochs we got an accuracy of 86%. In Fig.10 and Fig.11, the validation accuracy and loss have been showed and it can be assumed that the model is not getting overfitted (from Fig.11).

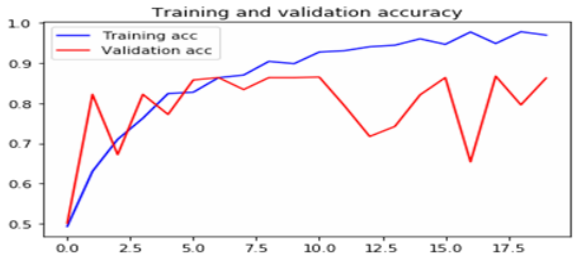

Fig. 10: Accuracy After Freezing all Layers

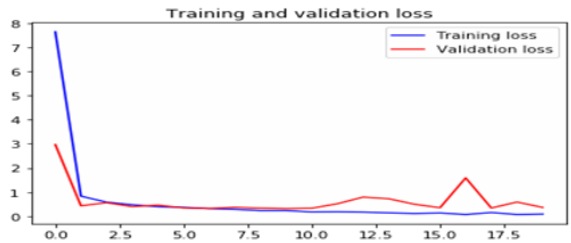

Fig. 11: Loss After Freezing all Layers

Now let's see some errors of prediction from our model, for each picture the original label and the prediction will be displayed with accuracy at the top. We will also mention the number of total wrong predicted images of the test dataset to grasp the overall concept of what we are going to do, we will try furthermore to reduce this problem and number of total wrong predicted images using various techniques in this paper.

Fig. 12: Sample of Wrong Prediction (Female)

Here (Fig.12), the Bengali female was declared as a Bengali male with a confidence of 63%. This could be the effect of her costume.

The total number of errors = 115 images (out of 835).

(ii) Last 4 layers without Data Augmentation: keeping all the previous state of our model (mentioned in B(i)), we just made the model trainable for the last 4 layers. Because it might help to overcome the problem of getting a huge number of wrong predicted images. We got 86% accuracy for this. Fig.13 and Fig.14 are representing their accuracy and loss respectively.

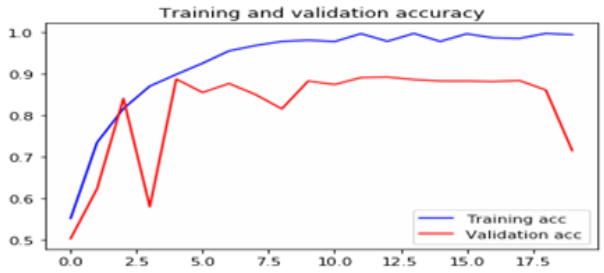

Fig. 13: Accuracy for Last 4 Trainable Layers (without data augmentation)

Fig. 14: Loss for Last 4 Trainable Layers (without data augmentation)

From figure (Fig. 14), it seems that the model is getting overfitted since after 18 epochs the validation loss jumped suddenly right there.

(iii) Training last 7 layers without Data Augmentation: After making the last 7 layers trainable without data augmentation the accuracy gets increased for the same model (from Fig.15), and loss decreased.

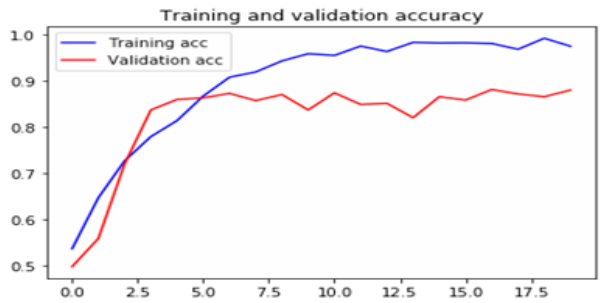

Fig. 15: Accuracy for Last 7 Trainable Layers (without data augmentation)

Wrong prediction's sample:

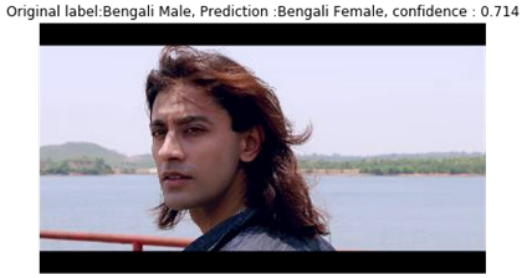

Fig. 16: Sample of Wrong Prediction (Male)

Here, the Bengali male was declared as a Bengali female (confidence: 71%). This could be the effect of his long hair.

Total wrong prediction (reduced): 100 images (out of 835).

(iv) Last 4 layers with Data Augmentation: Now, let's observe, how this model will behave with data augmentation, so we applied data augmentation for the last 4 layers and we got around 88% accuracy. Accuracy and loss are visible in Fig.17 and Fig.18 respectively.

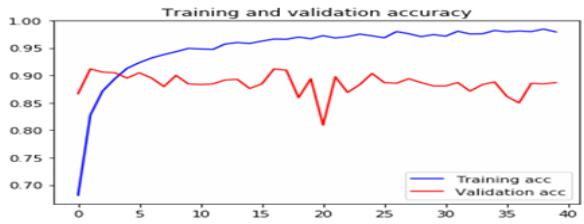

Fig. 17: Accuracy for L4 Trainable Layers (with data augmentation)

Fig. 18: Loss for Last 4 Trainable Layers (with data augmentation)

The accuracy seems good (Fig. 17) but from Fig. 18, it's evident that the model is getting more overfitted after 12 epochs. The wrong prediction is shown below as an example of this,

Fig. 19: Sample of Wrong Prediction (Female)

Here, the Bengali female was declared as a Bengali male with a confidence of 100%. This could be the effect of her different gestures.

In this case, the total number of wrong predictions (reduced): 95 images (out of 835).

(v) Last 7 layers with Data Augmentation: Similarly, for the same model, after running for 40 epochs, (for the last 7 layers) with data augmentation we got the accuracy of 88% ~ 89%. From the validation accuracy figure (Fig.20), it's clear that now this proposed model is performing better.

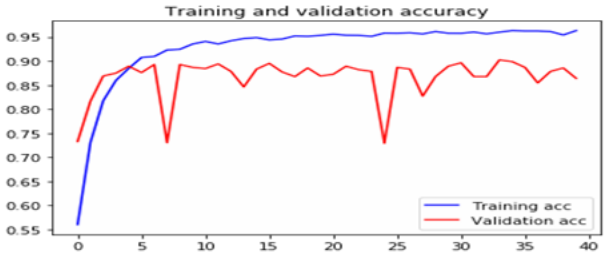

Fig. 20: Accuracy for Last 7 Trainable Layers (with data augmentation)

The validation accuracy at this stage looks very excellent compared with the all the previous 4 architectures of Keras fine-tuning.

To recapitulate, the Keras fine-tuning with VGG16, the summary is given in table 2,

TABLE 2: KERAS FINE-TUNING SUMMARY

Layers	Data Augmentation?	Accuracy
All layers (frozen)	No	86%
Last 4 layers(Trainable)	No	86%
Last 7 layers(Trainable)	No	85%
Last 4 layers(Trainable)	Yes	88%
Last 7 layers(Trainable)	Yes	89%

However, we have successfully reduced the number of total wrong predicted images to some extent.

VI. CONCLUSIONS

In conclusion, we can presume easily that getting high accuracy using CNN is arduous without transfer learning. Transfer learning has just given computer vision a new dimension. That's why in this paper we have made a Convolutional Neural Network-based model followed by many transfer learning approaches. The CNN model gave us around 83% accuracy for new testing images and we got terrific accuracy for different transfer learning models. In the future, we shall make improvements to our CNN and bottleneck features architecture.

ACKNOWLEDGMENT

We would like to show our gratitude to all the instructors of DIU NLP & Machine Learning Research Lab, Daffodil International University, for the proper guidance and the pearls of wisdom they shared.

REFERENCES

[1] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998).

[2] Osadchy, M., Cun, Y., Miller, M.: Synergistic face detection and pose estimation with energy-based models. The Journal of Machine Learning Research 8, 1197–1215 (2007).

[3] Ji, S., Xu, W., Yang, M., Yu, K.: 3D Convolutional Neural Networks for Human Action Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(1),221–231 (2013).

[4] Nisha Srinivas 1, Harleen Atwal 2 , Derek C. Rose 1 ,Gayathri Mahalingam 2 , Karl Ricanek Jr. 2 , and David S. Bolme 1, Age, Gender, and Fine-Grained Ethnicity Prediction using Convolutional Neural Networks for the East Asian Face Dataset, 2017 IEEE 12th International Conference on Automatic Face & Gesture Recognition,pp: 953-960.

[5] X. Wang, R. Guo, and C. Kambhamettu. Deeply-learned feature for age estimation. In 2015 IEEE Winter Conference on Applications of Computer Vision, pages 534–541, Jan 2015.

[6] X. Yang, B.-B. Gao, C. Xing, Z.-W. Huo, X.-S. Wei,Y. Zhou, J. Wu,and X. Geng. Deep label distribution learning for apparent age estimation. In The IEEE International Conference on Computer Vision(ICCV) Workshops, December 2015.

[7] D. Yi, Z. Lei, and S. Z. Li. Age estimation by multi- scale convolutional network. In Asian Conference on Computer Vision, pages 144–158.Springer, 2014.

[8] Y. Zhu, Y. Li, G. Mu, and G. Guo. A study on apparent age estimation. In The IEEE International Conference on Computer Vision (ICCV) Workshops, December 2015.

[9] G. Levi, T. Hassncer, Age and gender classification using convolutional neural networks, in: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015, pp. 34– 42.

[10] B.A. Golomb, D.T. Lawrence, T.J. Sejnowski, Sexnet: a neural network identifies sex from human faces, in: Proceedings of the 1990 Conference on Advances in neural information processing systems, 3, 1990, pp. 572–577.

[11] B. Moghaddam, M.-H. Yang, Learning gender with support faces, IEEE Trans. Pattern Anal. Mach. Intell. 24 (5) (2002) 707–711, doi: 10.1109/34.10 0 0244.

[12] S. Baluja, H.A. Rowley, Boosting sex-identification performance, Int. J. Comput. Vis. 71 (1)(2006) 111–119, oi: 10.1007/s11263- 006-8910- 9.

[13] M. Toews, T. Arbel, Detection, localization and sex classification of faces from arbitrary viewpoints and under occlusion, IEEE Trans. Pattern Anal. Mach. Intell. 31(9)(2009)1567–1581, doi: 10.1109/ TPAMI.2008.233.

[14] M.Wang, Y. Iwai, and M. Yachida. Expression Recognition from Time-Sequential Facial-Images by Use of Expression Change Model. Proc. 3rd IEEE International Conference on Automatic Face and Gesture Recognition, pages 324–329,1998.

[15] J. F. Cohn, Z. Ambadar, and P. Ekman. Observer-based measurement of facial expression with the facial-action coding system. The handbook of emotion elicitation and assessment, pages 203–221, 2007.

[16] Y. Lcun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[17] F. Jialue, X. Wei, W. Ying, G. Yihong, Human tracking using convolutional neural networks, IEEE Trans. Neural Netw. 21 (10) (2010) 1610–1623.

[18] Y. Cao, Y. Chen, D. Khosla, Spiking deep convolutional neural networks for energy-efficient object recognition, Int. J. Comput. Vis. 113 (1) (2015) 54–66.

[19] Choon-Boon Ng, Yong-Haur Tay and Bok-Min Goi, A Convolutional Neural Network for Pedestrian Gender Recognition, C. Guo, Z.-G. Hou, and Z. Zeng (Eds.): ISNN 2013, Part I, LNCS 7951, pp. 558–564, 2013.

[20] Dongmei Hana, Qigang Liu, Weiguo Fan, A New Image Classification Method Using CNN transfer learning and Web Data Augmentation, Expert Systems With Applications (2017), doi: 10.1016/j.eswa.2017.11.028.