Emotion detection in an interview based on facial analysis

Written by
Samavedam Manikhanta Praphul, Rasineni Hema Sumanth
Foundations of Artificial Intelligence
Khoury College, Northeastern University
Boston, Massachussetts 02115
samavedam.m@northeastern.edu, rasineni.h@northeastern.edu

Abstract

In this paper, we will explore detecting an emotion in an interview video using convolutional neural networks and determining the 3 dominant emotions in the interview. In this paper, we plan to train a custom model with different optimizers stochaistic gradient descent, adam optimizer and then we will contrast these model's performance against transfer learned VGGNet and LexNet models. Finally this paper also explores the impact of using data augmentation techniques such as image translation, image rotation, horizontal flipping an image for detecting the emotion of the person and perform fine-tuning and hyper parameter tuning.

1 Introduction

In post-COVID world, the majority of the interviews are conducted virtually. Gauging visual cues in virtual interviews is difficult as compared to physical interviews. So we intend to further enhance the virtual interview experiences by analyzing the facial expressions throughout the video for visual cues on determining the mood of the candidate in the interview. We are of the opinion that we can have additional information about candidate's mood throughout the interview process which can be useful in candidate selection in case of tie among two candidates by identifying the person who is more confident and happy about the position than the other. The idea is to supplement the interviewer but not intended to replace the interviewer for interview process. Having set the purpose of our exploration, now let us explore how to find the mood/emotion of the interviewee from an interview.

1.1 Emotion Detection

Detecting emotion in an interview is a difficult task and can be a CPU intensive task as it requires analyzing both the video and audio feed. Each parameter be it audio and video provide additional information which will improve the accuracy at identifying the emotion in a video. In other words, emotion detection in an interview requires to analyze for identifying the mood, confidence in the answers from the audio feed and track the face of the person in the video, process the face of the person for finding the emotion using facial expressions in the video feed. Finally superimpose the predictions from audio and video feeds for better result.

In this project, we are only exploring the possibility to identify the emotions of the interviewee based on facial expressions and refraining from speech analysis. Later we plan to enhance the model's performance by leveraging the new knowledge from audio analysis using memory models such as the recurrent neural networks (RNNs) for LSTM(Long Short-Term Memory). For identification of dominant emotions of the interviewee in an interview, we plan to break the task into 3 sub tasks.

- Breakdown the interview video into several snapshots taken at regular intervals.
- Maintain the record of the emotion present in the snapshot of the interview based on a pre-trained convolutional neural network (CNN).
- Analyze the record to identify the 3 dominant emotions in the video.

Further in this paper, we shall focus majorly on the identification of the emotion present in the snapshot as it is critical aspect of this 3 step process.

2 Background

Our project involves using convolutional neural network (CNN) to detect the emotion of the interviewee from the snapshot captured (image) from the interview at regular intervals. The power of a convolutional neural network comes from a special kind of layer called the convolutional layer. A convolutional layer can be visualized as containing several small square templates, called convolutional kernels, which slide over the image and look for patterns. This feature of CNNs allows a particular convolution kernel to find the patterns in any part of the image. This resilience of CNNs is called 'translation in-variance'.

We have chosen CNN, as we can use different filters to extract several local features which to a great extent depend on the adjacent pixels of the previous layer. So we get the benefit of finding the patterns in the images with ease and learn from a relatively very small set of parameters as compared

to a fully connected neural network. We gain heavily in reduced training time and simpler neural network for complex task of analyzing image data. Making use of the techniques like pooling we can get the required patterns without loosing much information in the discarded pixel data.

These techniques of convolutions and pooling allow to reduce the computations required to train the neural network as compared to fully connected networks without compromising on the model's performance such as accuracy, making CNNs computationally efficient and quicker to train as less number of learn-able parameters. Because of these benefits, CNNs are computationally and architecturally advantageous for processing images. As a result of all these reasons, we planned to use the CNNs for our work which also involves image processing to determine the emotion.

3 Related Work

Deep learning-based approaches, especially those using CNNs, have achieved great success in image-related tasks in recent years because of its ability to extract relevant information from images. Very carefully fine-tuned and optimized CNN's are very efficient for classifying different subtle emotion which otherwise (due very subtle changes in face) becomes hard to classify using other models. Due to it's efficient performance on emotion analysis CNN's has been used in many state-of-the-art algorithms for this particular work(J.Goodfellow,2015). However, due to the small data set size CNN's might over-fit the data. to avoid training a complex classifier on small data sets which will overfit previous works in this area used transfer learning for this type of work where the weights of the CNN are pretrained with data for related tasks before fine-tuning them using the our data set(Z.Wei, 2018). And also we use data augmentation to minimize over fitting. This approach has achieved better results, then to directly training the network on the small data set. Different activation function for different deep neural net model are also extensively explored (E.Nwankpa,2018.).We also try to experiment with few widely used activation functions.

4 Approach

We plan to use Open-CV to obtain the regular snapshots of front face of the interviewee and then used the haarcascade for frontal face to crop out the face from the snapshot. We initialize a vector or an array as the 'emotions vector' for maintaining the number of occurrences of the emotions in the interview, this vector will be used to determine the emotions which were present for majority of the time of interview. For predicting the emotion of the interviewee, we have used 4 different models. For training our models, we have chosen to train from Kaggle data set FER-2013.

4.1 Own Custom model

First 2 models are models developed from scratch having a common base model and differ in the optimizers used to compile the models, hence these are custom models developed. The base model uses 4 convolutional layers with activation as rectified linear unit (ReLU) based on our experiment and each convolutional layer is followed by a batch normalization layer and a max pooling layer of size (2,2). Our custom model was over fitting for the data provided, so we have opted for drop out layers to regularize the over fit neural network. The convolution layers are then followed by a drop out layer of 10% and 4 fully connected layers having 'ReLU' activation function. Base Model summary is shown in "Figure 1: Custom Model Summary".

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 64)        832

 batch_normalization (BatchN (None, 64, 64, 64)        256
 ormalization)

 conv2d_1 (Conv2D)           (None, 64, 64, 64)        16448

 batch_normalization_1 (Batc (None, 64, 64, 64)        256
 hNormalization)

 max_pooling2d (MaxPooling2D (None, 32, 32, 64)        0
 )

 conv2d_2 (Conv2D)           (None, 32, 32, 128)       32896

 batch_normalization_2 (Batc (None, 32, 32, 128)       512
 hNormalization)

 conv2d_3 (Conv2D)           (None, 32, 32, 128)       65664

 batch_normalization_3 (Batc (None, 32, 32, 128)       512
 hNormalization)

 max_pooling2d_1 (MaxPooling (None, 16, 16, 128)       0
 2D)

 dropout (Dropout)           (None, 16, 16, 128)       0

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 256)               8388864

 dense_1 (Dense)             (None, 256)               65792

 dense_2 (Dense)             (None, 64)                16448

 dense_3 (Dense)             (None, 7)                 455

=================================================================
Total params: 8,588,935
Trainable params: 8,588,167
Non-trainable params: 768
_________________________________________________________________
None
```

Figure 1: Custom Model Summary

4.2 Transfer learning

The other two models are based on transfer learning technique. The pre-trained models used in out approach are VGG16 neural network and ResNet neural network. For transfer learning, we have only trained the last 4 layers of these pre-trained models and the previous layers are not retrained. We have explored training only the last 1, 2,3 layers but performance was better when we trained the last 4 layers giving us hope that more layers we trained the better, however when we tried to train more than 4 layers, it was taking a lot of time to train because of the parameters to be learned. So for this paper, we have restricted ourselves to train only the last 4 layers to get best of the performance and the time taken to train the model.

The transfer trained models are followed by fully connected network with 4 fully connected layers with activation function as ReLU. Each of these fully connected layers are uniformly initialized and are followed by a layer of Batch Normalization and a drop out layer. The model summary of

the transfer learning models shown in the figures 2,3

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 2, 2, 512)         14714688
_________________________________________________________________
dropout (Dropout)            (None, 2, 2, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 2048)              8192
_________________________________________________________________
dense (Dense)                (None, 32)                65568
_________________________________________________________________
batch_normalization_1 (Batch (None, 32)                128
_________________________________________________________________
activation (Activation)      (None, 32)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                1056
_________________________________________________________________
batch_normalization_2 (Batch (None, 32)                128
_________________________________________________________________
activation_1 (Activation)    (None, 32)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1056
_________________________________________________________________
batch_normalization_3 (Batch (None, 32)                128
_________________________________________________________________
activation_2 (Activation)    (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 231
=================================================================
Total params: 14,791,175
Trainable params: 7,151,623
Non-trainable params: 7,639,552
_________________________________________________________________
```

Figure 2: VGG16 Transfer model summary

Additionally, we have explored different activation functions such as tanh, ReLU, Sigmoid activations and found that activation function ReLU provides better performance than the others and hence have used the ReLU activation functions for the layers we have used.

As mentioned, in order to prevent over fitting of the model apart from regularization, we have also performed data augmentation techniques such as horizontal flipping, random zoom (randomly cropping the images), random shear turn (tilting the images in either way) which aided in reducing the over fitting of the model.

Once these models predict the emotion, we update the occurrence of the emotion in the emotions vector by increasing the occurrence by 1 unit. Simultaneously we place the detected emotion in the image as label around the bounding boxes of the video feed for the user to know the emotion detected.

Finally after completing this repetitive process of taking snapshots at regular intervals and updating the emotions vector with the predicted emotion throughout the video, we obtained the emotions present in the video along with their occurrence. We then find the dominant emotion based on the maximum occurrence of the emotion in the emotions vector. We thus find the dominant emotions in the interview and accomplish the task of identifying the dominant emotions in the interview.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 2, 2, 2048)        23587712
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 2, 2048)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
batch_normalization_4 (Batch (None, 8192)              32768
_________________________________________________________________
dense_4 (Dense)              (None, 32)                262176
_________________________________________________________________
batch_normalization_5 (Batch (None, 32)                128
_________________________________________________________________
activation_3 (Activation)    (None, 32)                0
_________________________________________________________________
dropout_4 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 32)                1056
_________________________________________________________________
batch_normalization_6 (Batch (None, 32)                128
_________________________________________________________________
activation_4 (Activation)    (None, 32)                0
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_6 (Dense)              (None, 32)                1056
_________________________________________________________________
batch_normalization_7 (Batch (None, 32)                128
_________________________________________________________________
activation_5 (Activation)    (None, 32)                0
_________________________________________________________________
dense_7 (Dense)              (None, 7)                 231
=================================================================
Total params: 23,885,383
Trainable params: 1,335,815
Non-trainable params: 22,549,568
_________________________________________________________________
```

Figure 3: ResNet Transfer learning model summary

5 Experiments and Results

In this section, we will explore the experiments performed by us and results obtained. We have divided each experiment as sub-section of this section for ease of access.

5.1 Activation Function

We begun with experimenting with different activation functions for the hidden layers. For this paper, we are comparing the activation functions ReLU and TanH for our custom model. Comparative study of accuracy of the models using these activation functions as shown in figure 4.

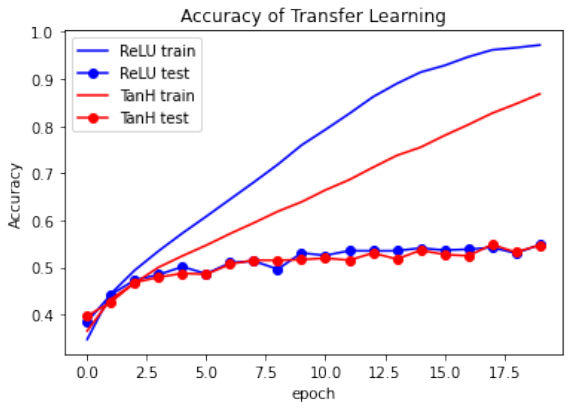


Figure 4: Accuracy with different activation functions

Similarly the loss values across epochs for the loss of the custom model with different activation functions as shown in figure 5.

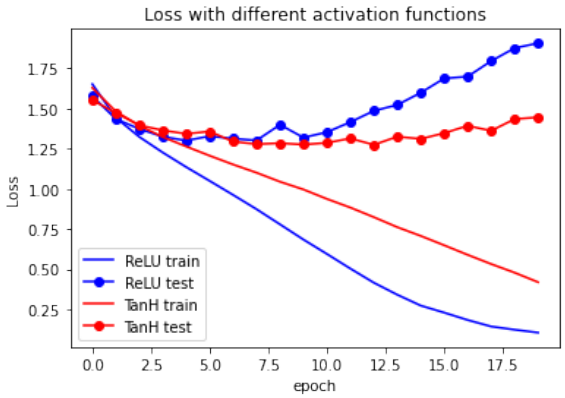


Figure 5: Accuracy with different activation functions

As seen from these figures, we observe that ReLU activation function enables the model to achieve higher accuracy with little training as compared to tanH activation function. The deviation in the Loss functions for train and test data sets is due to over-fitting of the data, so overall for our project ReLU activation is better for training the model.

5.2 Optimizer

In this experiment, we have compared the model performance with two different optimizers namely Stochastic Gradient Descent (SGD) and Adam to understand its effect on the model performance. We have trained the custom model for 30 epochs the number of epochs taken to perform the comparative study. The accuracies of the custom model with different optimizers is shown in the figure 6.

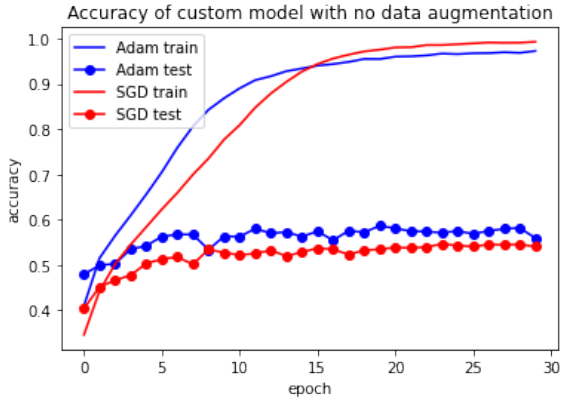


Figure 6: Accuracy of custom model with different optimizers.

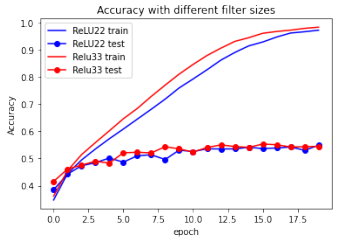
Similarly the loss values across epochs for the custom model with different optimizers are shown in the figure 7.

Figure 7: Loss of Custom model with different optimizers

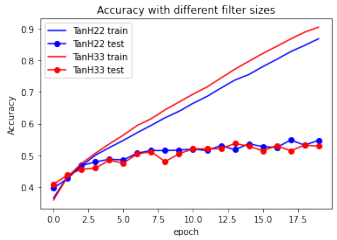
By closely observing the plots, it is quickly evident that Adam optimizer is quick to train for higher accuracy which is justified by the momentum in learning rate. For majority of the time, the Adam optimizer out performs the SGD optimizer, however the there isn't much difference in the model performance with the optimizer as SGD optimizer performs better after several epochs in accuracy and loss metrics.

5.3 Filter Size

In this experiment,we have compared the model performance with two different filter kernel sizes (2,2) and (3,3) to understand its effect on the model performance. We have trained the custom model for 30 epochs to perform the comparative study and the accuracies of the custom model with different filter kernel filter sizes are shown in the figure 8.



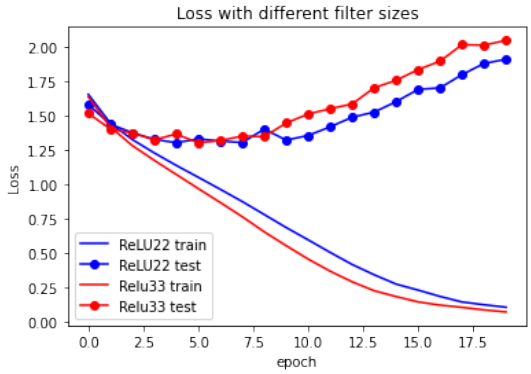
(a) ReLU



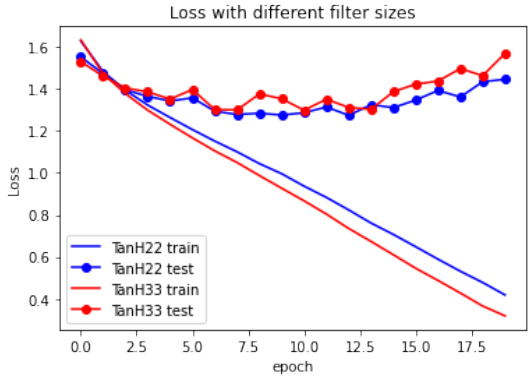
(b) TanH

Figure 8: Accuracy with different filter kernel sizes

Similarly the loss values across epochs for the custom model with different filter kernel sizes are shown in the figure 9. Clearly from these plots there is minimal impact of



(a) ReLU



(b) TanH

Figure 9: Loss with different filter kernel sizes

filter kernel sizes on the model performance. Hence we shall proceed with filter of kernel size (2,2), 'ReLU' activation function for further model development.

From the plots explored till now, it clear that the custom model is being over-fit for the training data set and thus performing worse on the test data set. In this paper, we will now further experiment with different techniques to reduce/prevent the over-fitting of the model. Precisely, we will explore the data augmentation technique and regularization technique to incorporate the variance in the training data-set to reduce the over-fitting of the neural network.

5.4 Data Augmentation

In this sub section, we opted to use the data augmentation to incorporate additional variance in the training data set. The data augmentation techniques experimented are randomly flipping of the image, randomly zooming into image (cropping image by maximum of 20%) and randomly rotating the image either side (by a maximum of 30%). Comparative study of the accuracy of the custom model with data augmentation over 30 epochs is shown in figure 10. We observe that the data augmentation has reduced the over fitting of the

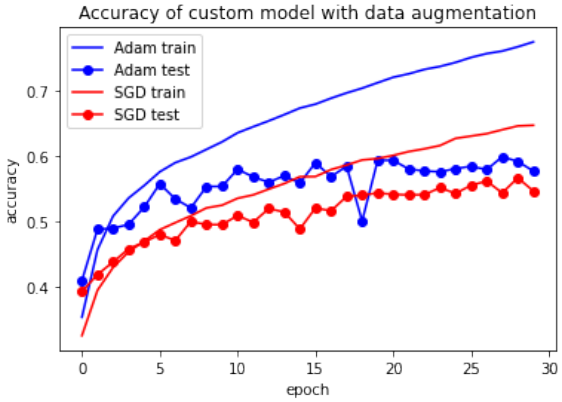


Figure 10: Accuracy of custom model with data augmentation

model, so the model is able to better generalize and so the test accuracy has improved and is close to training accuracy. Similarly the loss value across epochs for the custom model with Adam and SGD optimizers with data augmentation are shown in figure 11.

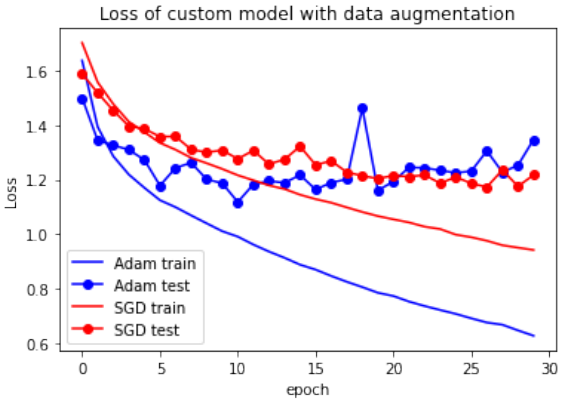


Figure 11: Loss of Custom model(with data augmentation)

Model with SGD optimizer performs well than the model with Adam Optimizer.

5.5 Transfer Learning

Finally we experiment with transfer learning using Residual Neural Network (famously known as ResNet50 or ResNet) which has 50 hidden convolution layers and Visual Geometry Group16 (famously known as VGG16) which has 16 weight layers. As mentioned in the approach section, we are training only the last 4 layers of the pre-trained models for our specific training data set. We have observed that by training these last 4 layers we get decent model performance while reducing the time required to train these models. Initially we have trained these for model for 5 epochs, the metrics of each transfer trained neural network shown in the figure 12, 13 for ResNet and VGG16Net respectively.

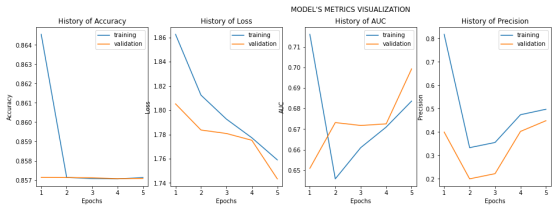


Figure 12: ResNet Transfer learn accuracy

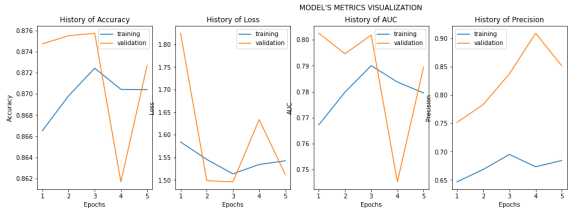


Figure 13: VGG16Net Transfer learn accuracy

We observe that the performance of these models can be improved further if we train these models for some more epochs to have a better model performance. So we have trained these models for 20 epochs and conducted the comparative study of accuracy of the transfer learned models shown in the figure 14. Similarly the loss value across

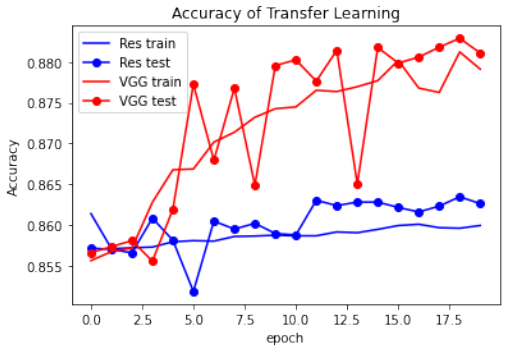


Figure 14: ResNet & VGG16Net Transfer learn 20 epochs

epochs for the models transfer learned from ResNet and VGG16 net are shown in the figure 15.

We clearly observe that the model transfer trained from VGG16 net has a higher accuracy than the model transfer trained from ResNet. However the model transfer trained from ResNet is more consistent in the performance as compared to model transfer trained from VGG16. This results are consistent with the results obtained after 5 epochs.

6 Conclusion

Based on the experiments conducted and the results observed, we have concluded that the transfer learning models have outperformed the small custom model developed and trained. Clearly the custom model, we have chosen has very few layers compared to the pre-trained ResNet and

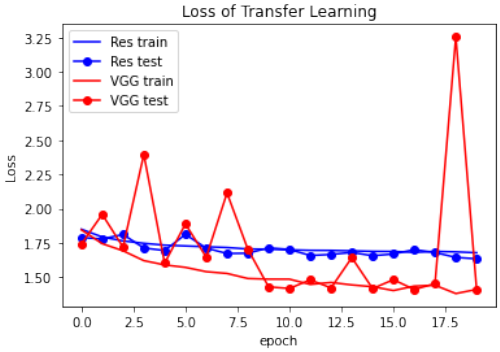


Figure 15: Loss of ResNet & VGG16Net for 20 epochs

VGG16Nets. So clearly larger networks are capable of identifying emotion better than small networks. Additionally, we have observed that our model is easily over-fit for the small training data set of FER-2013, so the neural network face this issue of over fit.

7 Future Scope

In future, we plan to extend the modem to identify more expressions apart from 7 emotions trained. As discussed in the beginning of the paper, we can additionally taking the knowledge from the audio about the emotion and then use the hybrid response of both facial analysis and audio analysis (using RNN) to identify the emotion in the interview video. Finally we can develop the model further to identify if someone is trying to fake things up in the interview.

8 Contribution

- Samavedam Manikhanta Praphul
  - Data Preparation
  - Data augmentation
  - Model comparison
  - Parameter tuning
  - Presentation and Paper
- Rasineni Hema Sumanth
  - Model Building
  - Validations
  - Hyper parameter tuning
  - Model comparison
  - Presentation and Paper

9 Acknowledgment

We would like to thank Prof. Robert Platt for providing us with an opportunity to explore the field of convolutional neural networks and Xupeng Zhu for guiding us to explore the data augmentation for this project.

10 References

I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler,

D.-H. Lee, Y. Zhou, C. Ramaiah, F. Feng, R. Li, X. Wang, D. Athanasakis, J. Shawe-Taylor, M. Milakov, J. Park, R. Ionescu, M. Popescu, C. Grozea, J. Bergstra, J. Xie, L. Romaszko, B. Xu, Z. Chuang, and Y. Bengio. Challenges in representation learning: A report on three machine learning contests. Neural Networks, 64:59 – 63, 2015. Special Issue on 'Deep Learning of Representations'.

neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

Research on the Deep Learning of the Small Sample Data based on Transfer Learning School of computer, Guangdong University of Technology, Guangzhou 510006, China.

Synergy Innovation Institute of GDUT and Heyuan, Heyuan, 517001.China.

Activation Functions: Comparison of Trends in Practice and Research for Deep Learning Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall.