Webpage navigation for differently abled people using Machine Learning

Aashish Thyagarajan
thyagarajan.a@northeastern.edu
001205798
Northeastern University

Hari Prasath Sivanandam
sivanandam.h@northeastern.edu
001880685
Northeastern University

Pavithran Ramachandran
ramachandran.p@northeastern.edu
001885506
Northeastern University

Abstract

The project aims in creating a means for navigating web-page using facial movements that could help differently abled people to navigate web-pages on a computer screen.

1 Introduction

The traditional navigation of web pages either in phone or computer requires hands or fingers to control mouse or touch screen. For differently abled people (who do not have accessibility to navigate screens) have options such as text-to-speech and voice controls for navigation. Our projects tries to create a navigation technique that requires facial movements that the computer can recognize and use as the navigation commands such as scroll up or down. The problem is hard to solve with traditional machine learning as the training data and the live data might vary drastically is frame size, face size, ambient light, face tone etc.

2 Related Work

Tensorflow can now be used in JavaScript that opens a door for creating Machine learning projects on the web browser with input data from sensors such as web-cam / front-cam, microphone, etc. We got inspired by this video which uses facial recognition to play the Pac-Man game in the web browser. We will use the same idea and experiment with the machine learning model(s) we have learnt in the course to create a navigation method using face gestures.

3 Dataset

The dataset was obtained from a Kaggle competition - YouTube Faces Dataset with Facial Keypoints. A brief description of the dataset is given below.

The dataset used is a processed version of the YouTube Faces Dataset, that basically contained short videos of celebrities that are publicly available and were downloaded from YouTube. There are multiple videos of each celebrity (up to 6 videos per celebrity). The original videos are cropped around the faces, plus kept only consecutive frames of up to 240 frames for each original video. (This is done for reasons of disk space, and also to make the dataset easier to use). Additionally, the facial key points for each frame of each video are extracted using this amazing 2D and 3D Face alignment library that was recently published. All videos with extremely bad keypoints labeling were removed. The total size of the dataset is 10GB. There are about 1293 videos in this dataset and overall these videos total into 155,560 single image frames.

Submitted to CS 6140 Machine Learning Fall-2018 Course.

4 Cleaning and EDA

4.1 Cleaning

The first task was to extract the frames from the video file and the landmark points from the csv files to have an one to one corresponding dataset. The initial dataset was ready to be cleaned and basic exploratory data analysis was done to understand the data in hand. Frames that had less than 68 landmark points were discarded as few of the points might have been outside the frame. Images that had much closer landmark points were tagged as "outliers" and were eliminated before EDA. After cleaning and validating the steps by plotting the image, the output is shown below:

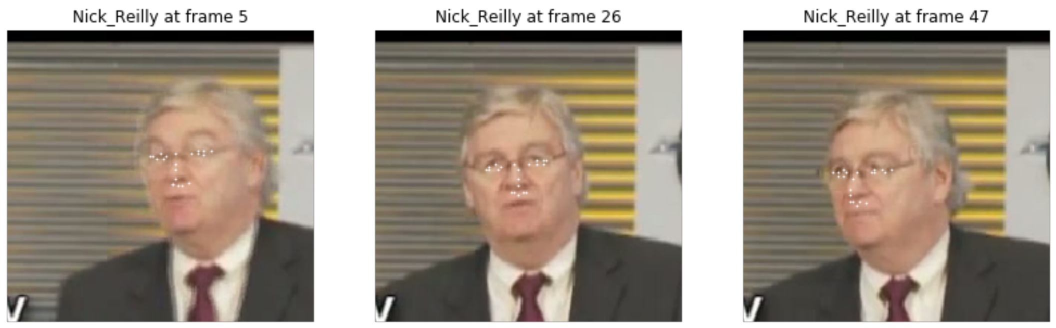


Figure 1: Plot after cleaning the dataset

4.2 Exploratory Data Analysis

The below parameters of the dataset are studied as a part of EDA.

- Number of videos per person
- Number of frames per video
- Image size of each frame
- Image size of the face in each frame

These parameters are plotted to get a better understanding of the dataset in hand.

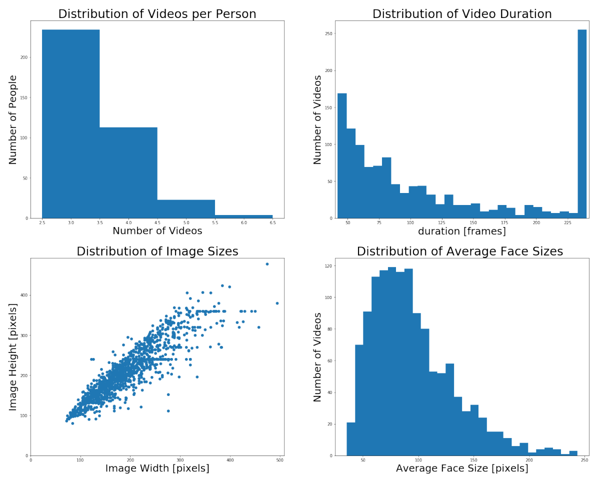


Figure 2: Basic EDA plots

From the above plot it is evident the size of faces vary from each video and even in frames of each video. The faces are are normalized to see the average face shape of the entire dataset. The below plot shows the average shape of face.

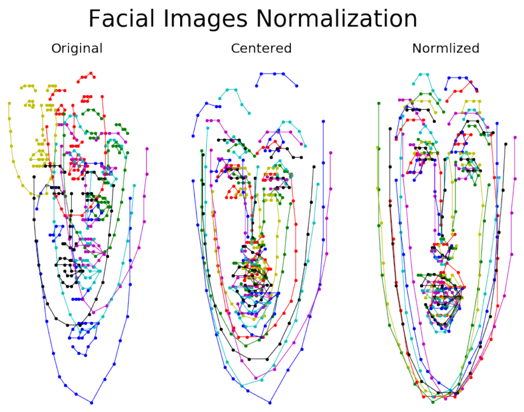


Figure 3: Normalization of Face images

A classical method such as K-Means clustering is applied on the dataset and clustered into 3 clusters i.e. Face facing straight, left and right in the frame.

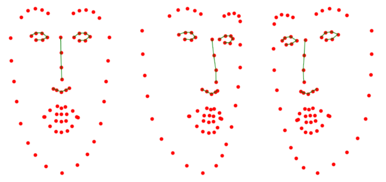


Figure 4: Face orientations from the normalized dataset

5 Pre Processing

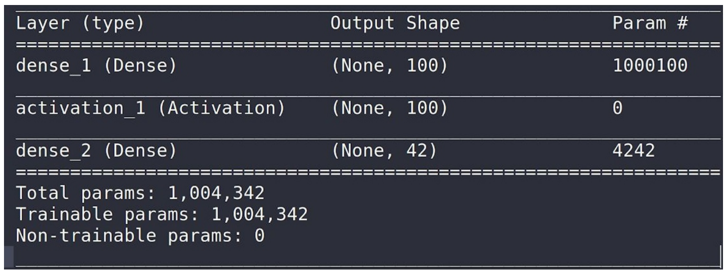
The pre-processing includes the following steps:

- Identify if a face is present in the image.
- If a face is found, extract the facebox.
- Convert the facebox image to grey scale.
- Resize the image to 100×100 image. The resultant array is a numpy array of shape $(100, 100, 1)$.
- Extract the 21 landmark points from the superset of 68 points. (discarding points of eyebrows, jaw-line etc.)
- The x and y co-ordinates of the landmark points are adjusted so as to fit the 100×100 reshaped face image.
- Reshape the landmark points to shape of $(42, 1)$ i.e.21 points each having x and y co-ordinates are flattened (21×2).
- The grey scale image is normalized between $(0, 1)$
- The x and y co-ordinates are formalized between $(-1, 1)$

The face image and the coresponing landmark points are now ready to be trained in Artifical Neural Network models.

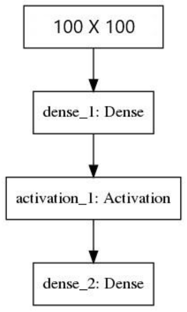
6 Simple Neural Network

To start building a model, a simple neural network that has a fully connected Dense layer that accepts the image is used.The input layer has the shape of 10000. The activation layer is Relu. The ouput layer is again a Dense layer that outputs the 42 values (21 {x, y} pairs).



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               1000100
_________________________________________________________________
activation_1 (Activation)    (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 42)                4242
=================================================================
Total params: 1,004,342
Trainable params: 1,004,342
Non-trainable params: 0
```

(a) Model Parameters



(b) Model Architecture

Figure 5: Simple Neural Network

The validation and training loss calculated in terms of "mean square error" is used as the estimate to find how well the model performs. Since it is a simple Neural Network with just one Activation layer, the model seems to struggle with new images that are not used for the training. The training loss fluctuates as the weights and biases for the neural network computed in the previous iterations were wrong when a new image is tested using the model. This proves that the problem needs a complex Neural Network.

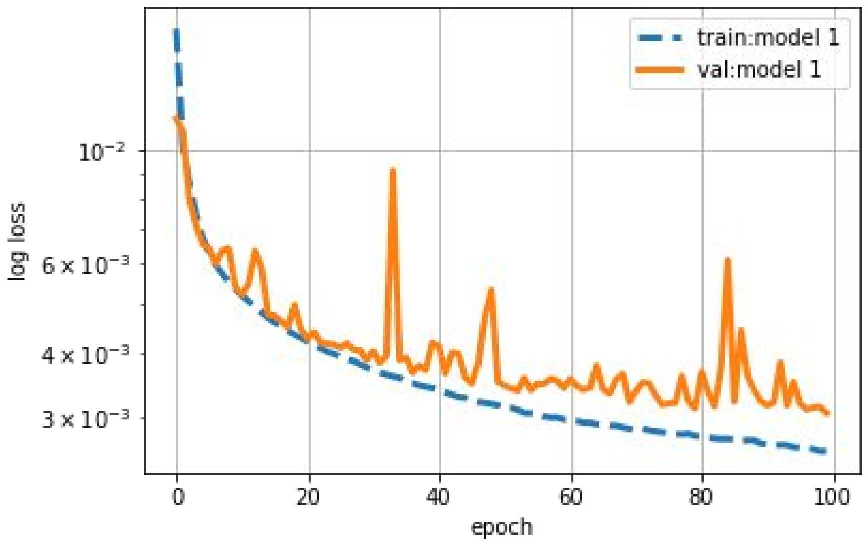
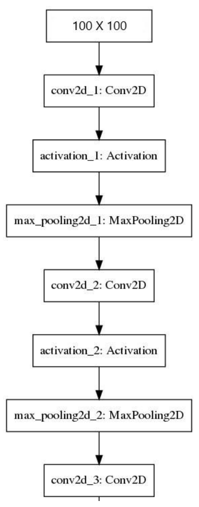


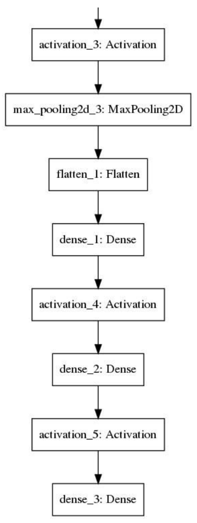
Figure 6: Simple Neural Network Validation and Training Loss

7 Convoluted Neural Network

From the results above it is evident that a Convoluted Neural Network might help in achieving a better model with a comparatively lower loss. The input and output layer's dimensions will be the same, but instead of a Dense layer, Convoultional Layer will be applied. The Conv2D of Keras library is used here. The activation method is Relu activation. Apart from this, a dropout layer is added to have a check on the over fitting of the model based on the input data. The model is trained with $80,000$ images with a batch size of $10,000$. It took the model 10 hours to complete 200 epochs and achieve a validation error of 10^{-3}. The industry standard for error in detecting landmark points of the order 5×10^{-3}. The CNN architecture is given below.



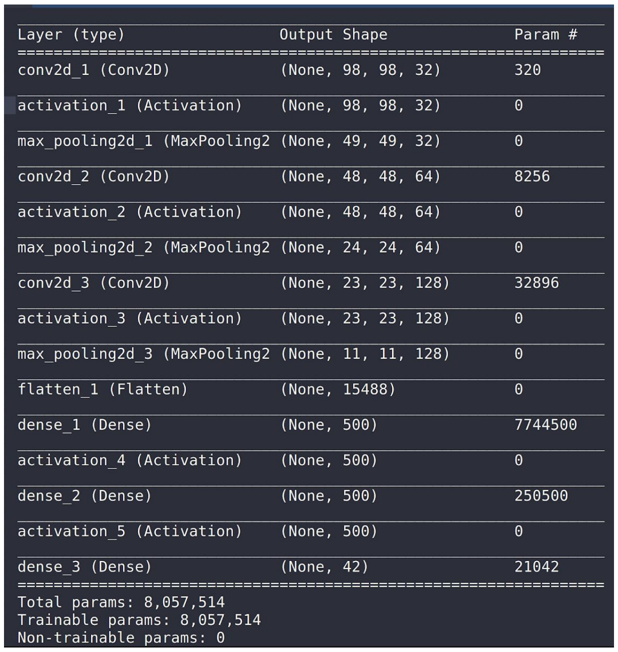
(a) CNN architecture part 1



(b) CNN architecture part 2

Figure 7: Convoluted Neural Network

The Conv2D layer increases the dimensions of the input, and hence the activation layer has 3 dimensional array to process. The Activation layer assigns the value of max(0, $A[i][j][k]$) and passes this value to the next layer. The number 0 represents that the neuron was not set, while the positive number implies that the neuron got activated in the previous layer. The MaxPooling Layer reduces the dimension of the input, and uses the sliding window and passes the maximum value in that window as the input to the next layer. The Flatten Layer simply flattens the 3D matrix to a 2D matrix (or 2D matrix to a vector). Similar to the simple neural network model the output is 1×42. The output is rearranged in the form of pairs and re-scaled to fit the input image. The model parameters are seen in the below figure.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 98, 98, 32)        320
_________________________________________________________________
activation_1 (Activation)    (None, 98, 98, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 64)        8256
_________________________________________________________________
activation_2 (Activation)    (None, 48, 48, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 23, 23, 128)       32896
_________________________________________________________________
activation_3 (Activation)    (None, 23, 23, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 11, 11, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 15488)             0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               7744500
_________________________________________________________________
activation_4 (Activation)    (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               250500
_________________________________________________________________
activation_5 (Activation)    (None, 500)               0
_________________________________________________________________
dense_3 (Dense)              (None, 42)                21042
=================================================================
Total params: 8,057,514
Trainable params: 8,057,514
Non-trainable params: 0
```

Figure 8: Convoluted Neural Network Parameters

The performance of the model is evaluated validation and test loss. The CNN model reach a validation loss of 10^{-3} within 200 epochs.

7.1 Quantitative results

The comparison below shows that the CNN outperforms the simple neural network model and the abrasions in the validation loss is not present, meaning the model does not over fit the training dataset.

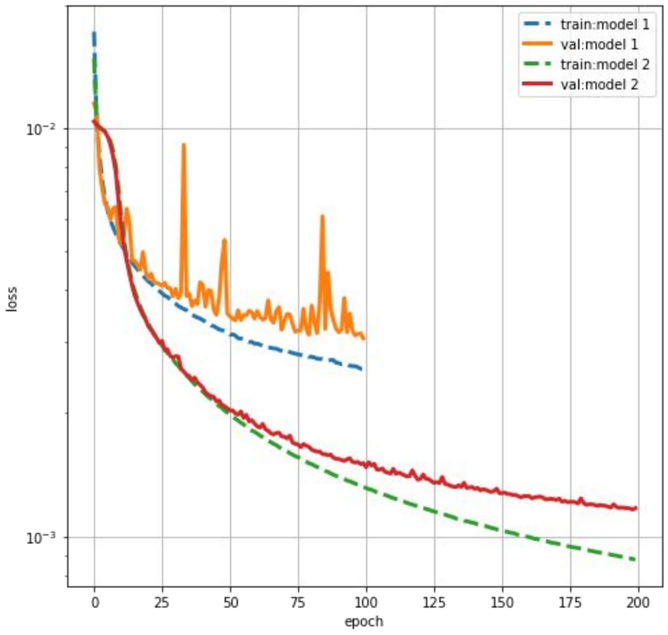


Figure 9: Validation and Test Loss

8 Testing Model

For validation, the trained model is tested using random image samples taken from the dataset. The MSE value would not suffice as it would give only the deviation of the predicted value from the original value, but since the problem is related to image processing, the visual inspection is required.

The following steps are required to validate the predicted output.

- Get the 42 prediction output from the model.
- Rearrange the points as pairs ({x, y} co-ordinates).
- Re-scale the value in terms of 100.(The output is in the range {-1, 1}).
- Shift the point to the original position.

8.1 Qualitative results

The faces were plotted along with the landmark points and seems to be closer to the original value from the test data. This is to measure the quality of the model prediction with the actual face.

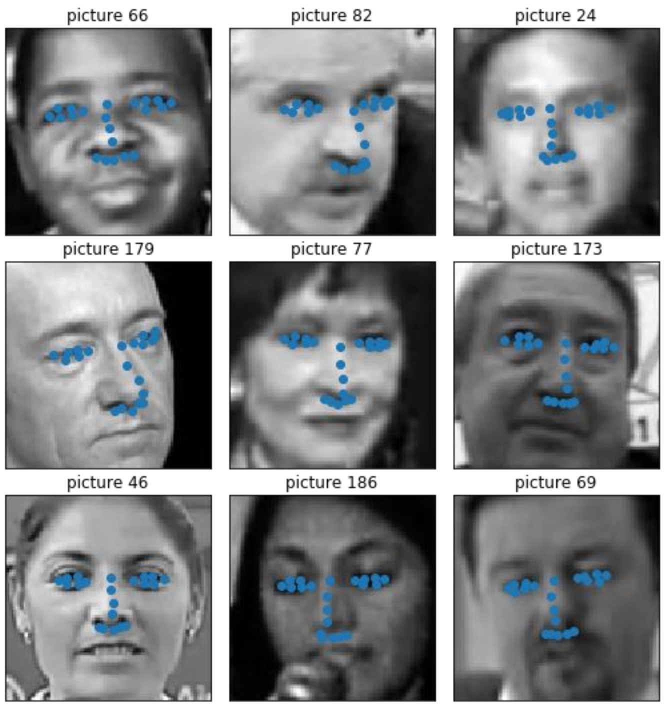


Figure 10: Visual Validation of the CNN model

9 Controlling mouse using landmark points

After the post processing step, the landmark points are plotted on a 2D matrix. The nose and eye points are tracked continuously and the movement with respect to the x-axis and y-axis are calculated. The change in x-axis translates into horizontal mouse movement and the change in y-axis translates to vertical scroll (positive for upward scroll and negative for downward scroll). This is instructed to the operating system using PyAutoGUI package. The latency between the facial movement and the action of mouse on the screen is noted to be 0.65 seconds which is comparable to the actual PC mouse. Thus the facial landmarks were extracted using Deep Learning and is used for controlling the mouse of a PC.

10 Demo

A brief visual demo of the project can be seen in the following link. You can see the mouse being moved left and right as per the changes in the head moment horizontally and scrolling up and down as the head moves vertically.

11 Future work

The future work for the current project is listed below:

- For mouse click, the relative position upper and lower eyelids are necessary. If the difference in the eyelid position is below certain threshold, activate click (left/ right). The Eye Aspect Ratio needs to be estimated and could be used to simulate the mouse click events.
- Build separate models for each facial structure (eye, nose, mouth, ...) and use the combination of models to better evaluate the face.
- The application runs only on PC, but the model and guiding functionality can be ported for mobile applications as well. (Tensorflow for Android)

12 Conclusion

The simulation of mouse movement and scrolling feature was successfully implemented using a Facial Landmark detection. A deep learning model with a validation loss in the order of 10^{-3} was trained and tested with images from the web cam. The facial points movement was translated to mouse gestures using PyAutoGUI package. The latency between movement and its corresponding translation on the PC was around 650 milliseconds (measured when then web-cam data is not plotted back on the screen).

References

[1] Wu, Y., & Ji, Q. (2018). Facial Landmark Detection: A Literature Survey. International Journal of Computer Vision, 1-28.

[2] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR, abs/1704.04861.

[3] 300 Videos in the Wild (300-vw) Challenge & Workshop (iccv 2015) `https://ibug.doc.ic.ac.uk/resources/300-VW/`

[4] 300 Faces In-the-wild Challenge (300-w), Iccv 2013 `https://ibug.doc.ic.ac.uk/resources/300-W/`

[5] A. Bulat and G. Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In International Conference on Computer Vision, 2017.