COMP-1804 Applied Machine Learning

Michael Briggs
001002629
Data Science MSc
University of Greenwich
London, Greenwich
mb5878n@gre.ac.uk

***Abstract*—This document contains an investigation for end to end multi label, multi class machine learning models that identify different properties of facial features from an image. While observing the effects of annotation of images, data cleaning and augmentation, data ingestion pipelines, testing results and different model architectures.**

***Index Terms*—Machine Learning, Neural Network, Deep Neural Network, Convolutional Neural Network, Facial Recognition**

I. Introduction

Facial feature recognition is a complex task that machines can accomplish that has many implications for public use. This investigation looks into five features that are within a small dataset of 2000 images and training a network to identify if the person in the image has glasses or not but also distinguish between sunglasses and normal glasses. Check to see if there are wrinkles or freckles present, hair colour is also defined with nine labels, brown, black, grey, blond, red, white, mixed and other. The type of hair is also a goal for this model with four labels for this class being bald or shaved, has few hair, has thick hair and not visible.

With these parameters laid out, it is clear this is a complex problem to be solved using computer vision with variations that each individual face, but learning outcomes from completing this task can have a multitude of implications from stronger verification security using an individuals facial features or leading to even more complex tasks such as recognising emotions which does have ethical implications regarding how this data is collected, stored and used as each face is unique and could be used to quickly identify someone or used maliciously to clone someones identity.

However, ethical issues aside, by implementing a Convolutional Neural Network (CNN) the expectation is to get a result that is better then a random guess.

II. Related Works

Although DeepFace [1] paper was released in 2014 it does bring up pipeline structure for facial recognition, but they then use a method of reconstructing facial images into a 3d model and predicting the labelled data correctly. Employing a Deep Neural Network (DNN) which they state as having over 120 million parameters they are able to get results that reached an accuracy of 97.35% on Labelled Faces in the Wild [2] dataset that contained four million images at the time of publishing their paper. Interestingly they use their 3d modelling technique to map 2d images and apply *"Frontalization"* which makes the face that has been recognised map to a front facing position.

Research has also been carried out in the paper titled Fast In-the-Wild Hair Segmentation and Color Classification [3] where they implement an augmented version of U-Net [4] to segment hair and colour from faces, using different colour spaces they are able to archive an accuracy of 89.6% by applying the LAB colour space to the images.

Batch size can have a great effect on how models react to training, as Kendel [5] has proved by conducting experiments on different batch sizes and learning rates to conclude with a recommendation of 32 or 64 for batch sizes and to use batches that are to the power of 2 as to take advantage of Graphics Processors and their faster training times.

III. Dataset Preparation

As this is an unlabelled dataset, an end to end process from image annotation to results will need to be undertaken. Annotation, exploration, image preprocessing and augmentation need to be applied to this dataset before training can begin and results be gathered.

A. Image Annotation

The dataset that has been used for this task includes an initial 2000 images that are both grey scale images and Red Green Blue (RGB) images, but none with any meaningful labels to begin with. Using a provided annotation program each of the images were labelled manually. This does however introduce a bias to the process of labelling data as it does rely on the person who labels that data to interpret the image 100% correctly without errors. Compiling all the images into a video using the provided program helpfully outputs the locations of each file that has been used to create the video of images. By loading this video into the annotation application and using a web based interface, annotations were added individually to each image, the output for these annotations are saved as a text file which needs to be cleaned before use.

To clean the output of the annotations, a second provided script is run that takes the output of the annotations and cleans out any data that is not required or needed for the process of feeding this data into a network saving it in a dataframe.

Also during this step, the locations of each of the images that is stored in a text file is also loaded and added to the same dataframe as the annotations. This dataframe that has the image locations and the corresponding annotations for that image are then saved to disk in a CSV format which will allow data to be loaded into a machine learning process.

B. Initial Exploration

It is only after these step have been completed, information on the dataset can be visualised and extrapolation of information contained within used, for instance Figure 1 shows that 1.45% of the dataset is marked as non human and would add no value to the training process and needed to be removed during the data cleaning process.

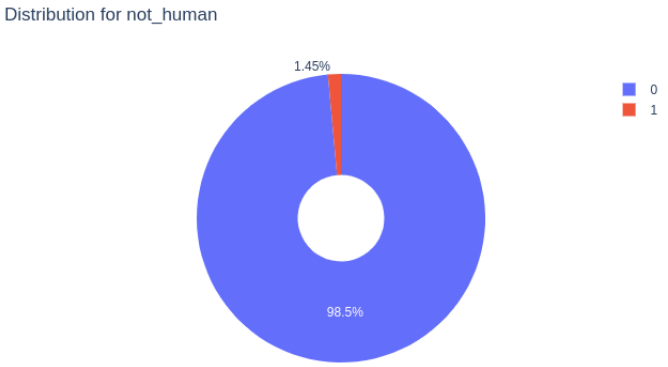

Fig. 1. Distribution of human and non human images.

Figure 2 shows the level of imbalance that is within the dataset just by hair colour alone, which makes the task of creating a model that can generalise well extremely difficult. After asking for a second dataset to complement the existing dataset, handpicked images were selected to to and bring a balance to the hair colours which added 157 more images to the grey, blonde, red, white, mixed and other class for hair. This can be seen in Figure 3 but still does help bring balance to this dataset.

Fig. 2. Distribution of images by hair colour

To try and rectify this imbalance a binning process was applied using hair colour as a method to balance with the limit for this seen in as a red line in Figure 4 and removing any images above this line from the dataset in a programmatic fashion with the result of this preparation seen in Figure 5

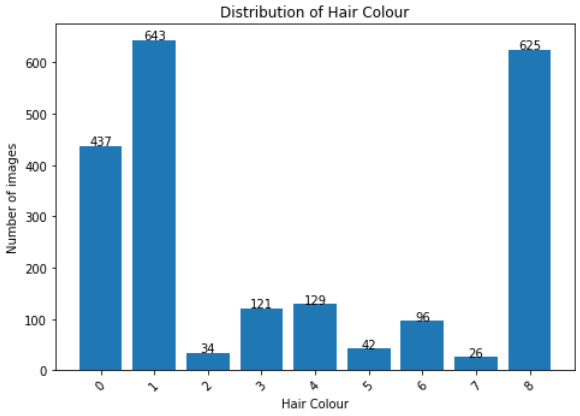

Fig. 3. Distribution of images by hair colour after extra data

This does however leave very few images from a small dataset to begin with to complete a complex task so this method of data augmentation was reversed and removed from the process.

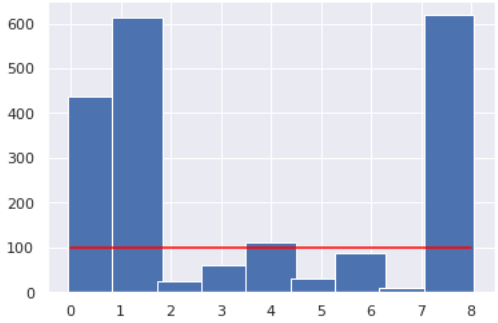

Fig. 4. Efforts to augment the dataset to a more balanced format

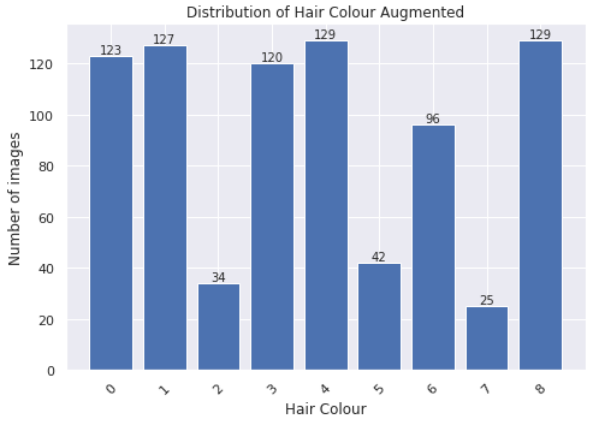

Fig. 5. A more balanced dataset based on hair color

With the reversal of this process completed, keeping the extra images that had been prepared in the dataset was made final with a total of 2154 images to be used for training, testing and validation of any trained neural network, a small but meaningful increase in images over the original dataset.

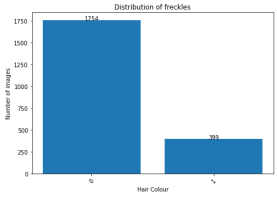

Fig. 6. Distribution of freckles

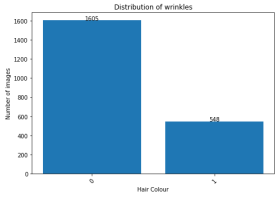

Fig. 7. Distribution of wrinkles

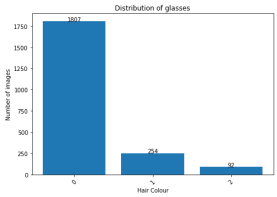

Fig. 8. Distribution of glasses

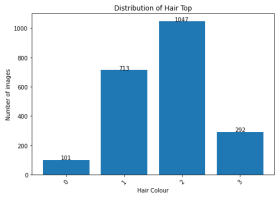

Fig. 9. Distribution of hair top

Figures 6, 7, 8 and 9 show the dataset in its full with all the distributions for each class.

Hair colour distribution								
brown	black	gray	blond	red	white	mixed	other	not visible
437	643	34	121	129	42	96	26	625

TABLE I
DATASET HAIR COLOUR DISTRIBUTION

Hair top distribution			
Bald or shaved	Few hair	Thick hair	Not visible
101	713	1047	292

TABLE II
DATASET HAIR TOP DISTRIBUTION

Freckle distribution		Wrinkles distribution	
Does not have	Has	Does not have	Has
1630	367	1605	548

TABLE III
DATASET FRECKLE AND WRINKLE DISTRIBUTION

Glasses distribution		
Does not wear	Normal	Sunglasses
1807	254	92

TABLE IV
DATASET GLASSES DISTRIBUTION

C. Preprocessing

This dataset is stored on a local disk as jpg files that are different dimensions, which are not an ideal format to feed into a Neural Network. Loading batches of images not only saves on resources as the entire dataset does not have to be loaded into memory and preprocessed in an instant. Loading the images uses the CSV file that has been prepared and has locations for each image and the annotations that are attached to them, using this information images are loaded into an array using Numpy. [6]

Preprocessing images makes the images uniform in size shape, but also allow a time to apply different masks or allowing Gaussian Blurs to be added by implementing the Image Augmentation Library (IAA) [7] which smooth edges out. Figure 10 is an output of this preprocessing step with the original image shown on the left and the preprocessed image on the right. These processed images have been resized to 150 pixels by 150 pixels for height and width using the Open Computer Vision (CV2) [8] library.

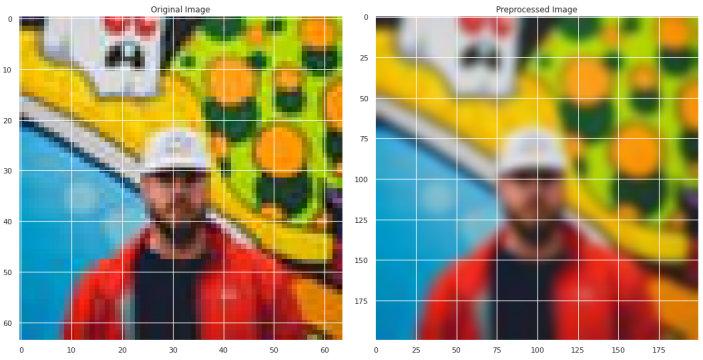

Fig. 10. Original image and preprocessed image.

With uniformity of images guaranteed due to preprocessing, the size and the shape of each array that is passed though the network is known before hand allowing this knowledge to be used when the model is being built.

D. Augmentation

Image augmentation is the process of making changes to an image, and in this case allows 2000 to become an infinite number of images as they will all be different from the original

image that has been passed though the augmentation process. IAA makes the process of applying different augmentations simple and four augmentations have been applied to this data pipeline to try and increase the models ability to generalise on images that it has not trained on. Each augmentation has its own function, and within each function there is a chance of a random scale of augmentation which increases the different images the model will see.

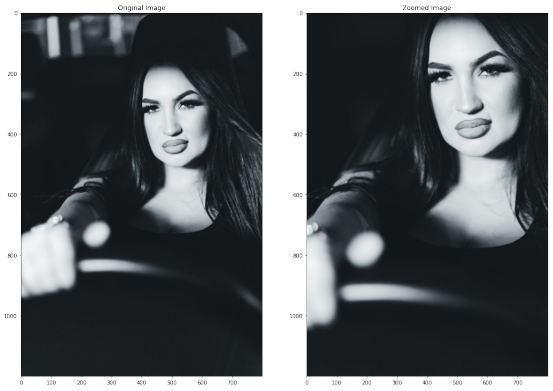

Fig. 11. Original image and zoomed image.

Figure 11 is an image from the dataset that has only been passed though the zoom augmentation, this changes the pixel locations and to a model makes it a different image altogether, however zooming in too far will cut out any useful information that could have potentially used so this particular augmentation is set to a zoom level of original size and 1.4 time zoom.

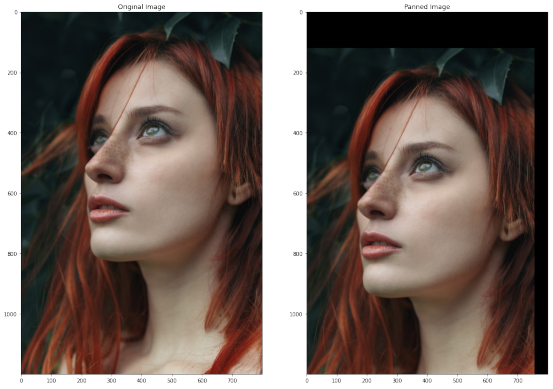

Fig. 12. Original image and panned image.

Panning an image shifts the entire image along its x and or y axis rather then zooming into the image itself. Figure 12 shows the result of passing an image though this process that is built into this pipeline, but panning an image too far will again, like zooming too far, will loose information that could be useful to a model, with that in mind and limit of 10% has been used to pan an image along the x and or y axis in a positive and negative fashion.

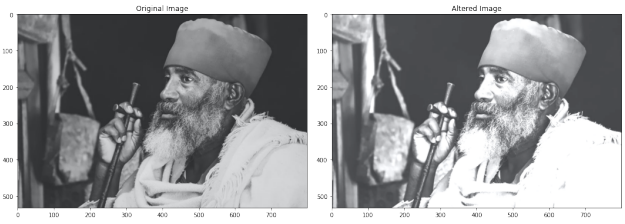

Fig. 13. Original image and brightened image.

Brightness of an image is the intensity value of each pixel that makes it, changing these values fundamentally alters an image for a machine learning model, Figure 13 is the result of passing an image through this augmentation. This may look like a small augmentation, but as a model reads the pixel intensity, it becomes a different image entirely for it to ingest. IAA allows two values to be passed that it will apply across all pixels in the image, for this the values 1.0 and 2.6 have been used, as a value below 1.0 will darken an image. This was intentional as there are many grey scale images and very dark images within this dataset and making an image darker will loose its information.

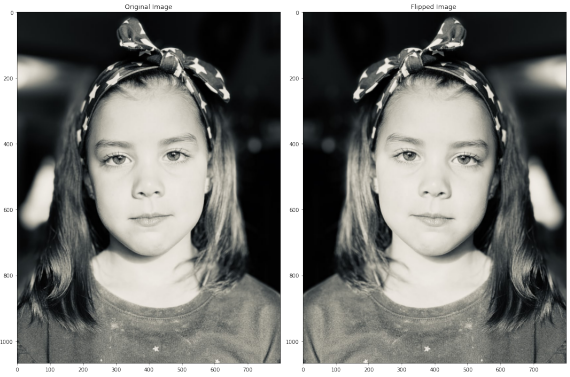

Fig. 14. Original image and flipped image.

By flipping an image on its y axis, all the pixels are reversed resulting in an output seen in Figure 14, as a human we can quickly recognise that these images are the same, but for a machine learning model this is not the case.

Fig. 15. Original image and gamma augmented image.

For the augmentation these are the five that have been applied to the pipeline, that are all in their own function so they can be applied independently from one another allowing a greater spread of possible augmentations that could be applied. There is a fifth function that makes calls to these individual functions based on a random number as to increase the chances that a random augmentation will be applied to an image as it it preprocessed.

E. Batch Generation

To train the model batches of images are selected at random by the index of the image, this shuffles the images and allows these images to be sent to be preprocessed and to be augmented before they are then fed into either the training process or the validation process. Figure 16 shows how the batch generator controls the flow of data for both training data and validation data based on the image files and the CSV being in place.

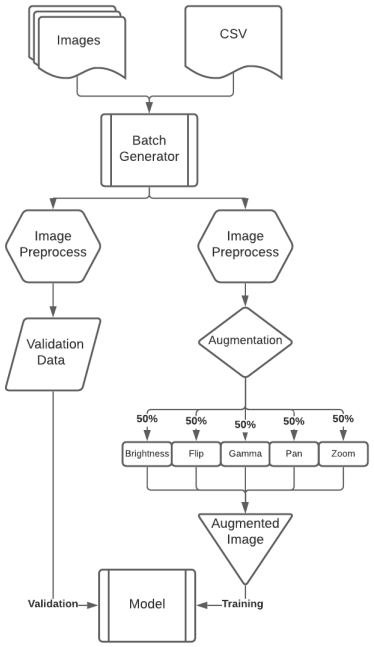

Fig. 16. Batch generation control flow.

IV. Machine Learning Method

For this task, a multi output Deep Convolutional Neural Network has been utilised as there are five classes, glasses, wrinkles, freckles, hair colour and hair top with multiple labels within each class. The plan for this is to reduce the number of models needed to be trained from five to one, with one model able to output predictions for five classes, this is achieved by utilising Tensorflow [9] Functional API over the Sequential methods to construct a model. This method allows greater flexibility and allows multiple outputs which is the aim for this solution. By building layers of a model, extraction of features occurs with convolutional layers that are shared by all outputs, after feature extraction the data is flattened into a one dimensional array that is fed into five different fully connected layers. Each of these layers is used to predict a single class with outputs that correspond to the number of outputs for each class.

As the dataset that is being used is bias for every class, as seen in Tables I,II,III and IV, a solution for this is to apply class weights during the training process. By implementing class weights, it lets the training process become more balanced as it assigns a weight to each class that is used to train. This is computed in the flow by using Scikit Learn [10] which has a function that takes the input of what needs to be balanced and outputs an array with the calculated bias.

Hair colour calculated bias				
brown	black	gray	blond	red
0.54741927	0.37204078	7.03594771	1.97704316	1.85443583
white	mixed	other	not visible	
5.6957672	2.49189815	9.2008547	0.38275556	

TABLE V
DATASET HAIR COLOUR CALCULATED BIAS

Hair top calculted bias			
Bald or shaved	Few hair	Thick hair	Not visible
5.32920792	0.75490884	0.51408787	1.84332192

TABLE VI
DATASET HAIR TOP CALCULATED BIAS

Freckle calculated bias		**Wrinkles calculated bias**	
Does not have	Has	Does not have	Has
0.61374002	2.69799499	0.67071651	1.96441606

TABLE VII
DATASET FRECKLE AND WRINKLES CALCULATED BIAS

Glasses calculated bias		
Does not wear	Normal	Sunglasses
0.3971592	2.82545932	7.80072464

TABLE VIII
DATASET GLASSES CALCULATED BIAS

The formula for this particular calculation is:

$$Wj = \frac{n\_samples}{(n\_classes * n\_samplesj)}$$

Where W is the weight j is the class, $n\_samples$ is the total length of the data, $n\_classes$ is the number of unique occurrences and $n\_samplesj$ number of rows in this class. This outputs a Numpy array with the weights for each class seen in Tables V,VI,VII and VIII. Adding these to a dictionary will allow this variable to be used during the fitting stage of the process which will apply weights to corresponding outputs and help with the bias seen in the dataset.

With the class weights calculated, we are able to start construction of the model, starting with the input layer, it needs to have the same dimensional shape as the image that is being

put through the network. For this model the input shape is 150 x 150 x 3 which is the same as the preprocessed image output. There are blocks of convolutional layers which are used to extract the features from the image, with thirty two feature maps at the start with batch normalisation layers after. These layers *"prevents small changes to the parameters from amplifying into larger and sub-optimal changes in activations in gradient"* [11] that are followed by down sampling the layers to reduce the computational power needed to train the model. In total there are ten of these convolutional, batch normalisation and max pooling layers that are used to extract the features and down size the information before it is fed through the fully connected dense layers.

As mentioned before, the convolutional layers are shared between all of the fully connected layers, with hair colour, hair top, freckles, wrinkles and glasses each having a fully connected layer which is named accordingly to match their purpose, each of these layers has a different architecture which have a different number of outputs to make predictions on the image. Matching the named output layer to the dictionary of weights lets the model use the calculated class weights to stop bias fitting, as there are many convolutional layers these dense layers are shallow in nature.

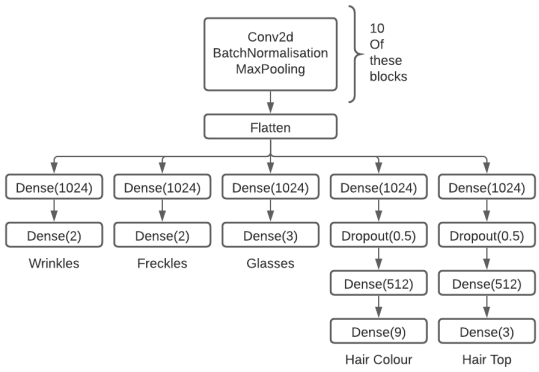

Fig. 17. Model dense layer architecture.

This architecture creates a model with $2,161,460$ total parameters, $2,160,052$ trainable parameters and $1,408$ non trainable parameters.

There are many hyper parameters that have great effects on how well a model will train and generalise to information that it has not seen, for instance, the size of the batch that is being fed into the algorithm has an effect on the training time and the ability to generalise well. Although it my be counter intuitive, Kandel [5] recommends a batch size of 32 or 64 after conducting experiments on how different batch sizes effect the models, by sticking to this recommendation a batch size of 32 is used throughout the training process.

The last output layers max the activation function of soft-max, as this function calculates the probability for each class as the output rather than a binary 0 or 1 answer that would prove useless for predicting hair colour that has nine outputs to select from. The loss fucntion for each of the outputs are categorical crossentropy losses as there are two or more outputs for each class, this is a measure of the label values and predicted predicted values and where the confidence level is high and the predicted value is wrong, the model is heavily penalised to try and get the model to learn the correct output for the input that it receives.

Each layer needs its own activation function and Rectified Linear Unit (ReLU) is used though out this model

$$y = max(0, x).$$

ReLU outputs a 0 for any negative value that it receives or for a positive value it returns this value back, and is computationally efficient and simple to implement in this model. For optimisation the Adam [12] optimiser is used due to its efficient use of computational power and the ability to handle high dimensional problem spaces.

V. Evaluation

A. Model 24

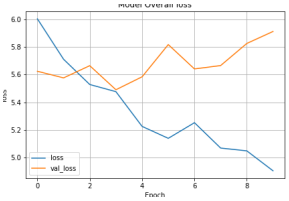

Fig. 18. Model 24 loss plot

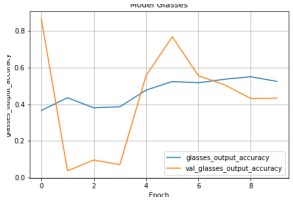

Fig. 19. Model 24 Glasses plot

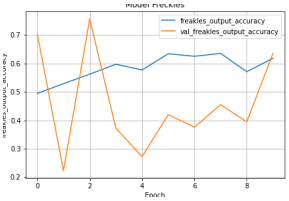

Fig. 20. Model 24 Freckles plot

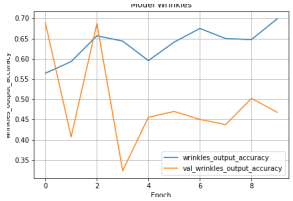

Fig. 21. Model 24 Wrinkles plot

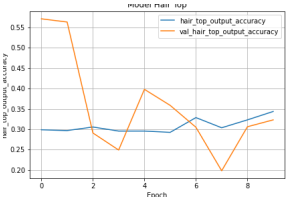

Fig. 22. Model 24 Hair Top plot

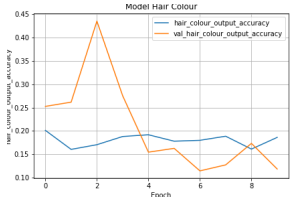

Fig. 23. Model 24 Hair Colour plot

Model 24 Evaluation Scores		
Test	Loss	Accuracy
Test loss	5.540980060895284	
Wrinkles	0.6791828	0.6025641
Freckles	0.5697809	0.71794873
Glasses	0.9509699	0.6025641
Hair Color	2.0279467	0.20512821
Hair Top	1.3130995	0.34615386

TABLE IX
MODEL 24 EVALUATION SCORE

Model 25 Evaluation Scores		
Test	Loss	Accuracy
Test loss	5.8993152777353925	
Wrinkles	0.75669074	0.53846157
Freckles	0.91791576	0.64102566
Glasses	1.0505298	0.55128205
Hair Color	2.0233727	0.17948718
Hair Top	1.150806	0.37179488

TABLE X
MODEL 25 EVALUATION SCORE

Model 24 has been trained over a total of 10 epocs with 100 steps per epoc, and obtained decent results for three of the five outputs that were needed. Seen in Table IX we can see that this particular model fared well when distinguishing between wrinkles, freckles and glasses. However getting a correct result for hair colour or hair top this model performed extremely poorly. This could be due to the shallow nature of the dense layers that are used to output the result or even possibly node death due to the nature of a ReLU activation function always giving the output of zero. From the loss and accuracy plots there is little stability during training which makes this models performance poor.

B. Model 25

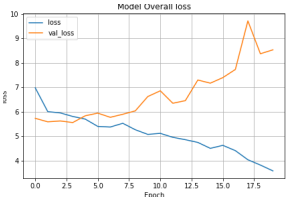

Fig. 24. Model 25 loss plot

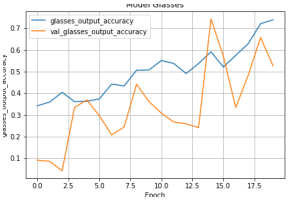

Fig. 25. Model 25 Glasses plot

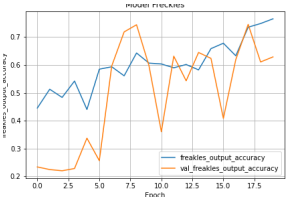

Fig. 26. Model 25 Freckles plot

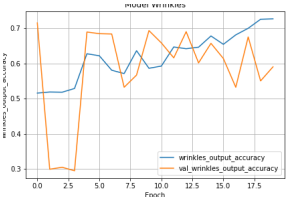

Fig. 27. Model 25 Wrinkles plot

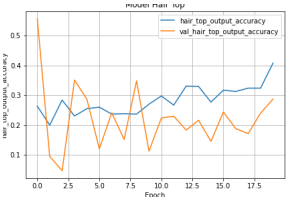

Fig. 28. Model 25 Hair Top plot

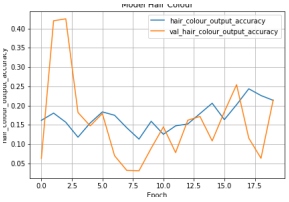

Fig. 29. Model 25 Hair Colour plot

Unlike model 24, this model was also train for 20 epocs with 100 steps per epoc and still shows very poor results for both hair colour and hair top classifications. With only a 20.5% accuracy for hair colour and 34.6% accuracy for hair top during evaluation.

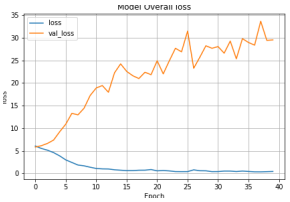

Fig. 30. Model 25 loss plot. Longer training session.

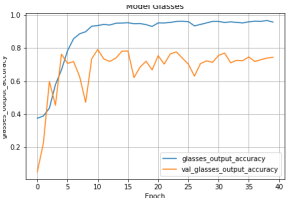

Fig. 31. Model 25 Glasses plot Longer training session.

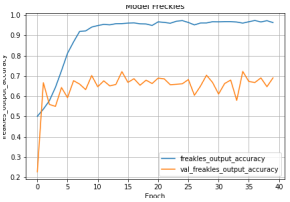

Fig. 32. Model 25 Freckles plot Longer training session.

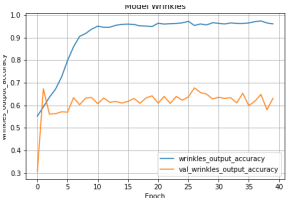

Fig. 33. Model 25 Wrinkles plot Longer training session.

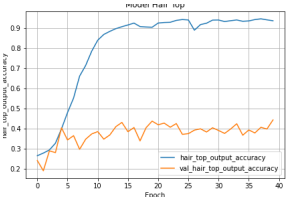

Fig. 34. Model 25 Hair Top plot Longer training session.

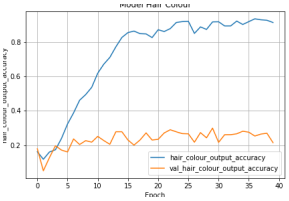

Fig. 35. Model 25 Hair Colour plot Longer training session.

Model 25 Evaluation Scores Longer Training		
Test	Loss	Accuracy
Test loss	14.3141108751297	
Wrinkles	3.4861119 4	0.6282051
Freckles	1.7069165	0.67948717
Glasses	1.1878738	0.974359
Hair Color	5.2519994	0.20512821
Hair Top	2.6812086	0.37179488

TABLE XI
MODEL 25 EVALUATION SCORE LONGER TRAINING SESSION

Increasing the number of epocs to 40 and increasing the steps to 250 improved all the scores except for hair top, wrinkles improved by 9%, freckles improved by 3.8% glasses which shows a 42.3% and hair colour improved by 2.6%improvement during evaluation which is a step in the right direction. The plots for loss and accuracy are also more stable, but still not optimal and show signs of over fitting on the data being used to train and rather then generalising it has learned the noise of the training data.

The high number of trainable parameters is likely the issue behind these results as too many parameters makes training a model very difficult and should be kept to as munimal as possible.

VI. Future Works

A better approach to this problem could have segmentation of the image, this would have helped identify parts of the image that it needs to classify correctly. Although implementing the Tensorflow API model was a challenge in itself, it was extremely satisfying to get the entire pipeline working correctly using self annotated data, implementing the pipeline in a manner that correctly works and getting a model to train on the data correctly. This process has shown that even Tensorflow has issues as the environment that was present on my own personal machine that was used to train and test models, it had Tensorflow 2.2. When I tried to implement class weights to try and remove the bias it would not work, but reading into the issue I found that it was not the way the it had been implemented, but that it would only work with Tensorflow 2.1 and multiple outputs.

Augmenting images too much is also something that has been taken away from this process, as too much augmentation will take away the true values that are needed to get a good result, but some augmentation does work well for small datasets.

These learning will also be used going forward as I enjoy this problem space of computer vision and have plans to use the techniques that have been learned here for the Masters Project, but it has also shown that there is more that needs to be learned before I really master Computer Vision.

Acknowledgment

Thank you to JJ Rose-Agoro, where we worked together well to produce the start of the data pipeline running, with bouncing ideas off one another it would have been a struggle to get this project completed.

Thank you to Dr. Dimitrios Kollias for not only delivering the content on this course, but for answering my questions when I needed them most and for finding me solutions to Google Colabs timeout issues when I need to switch over environments.

References

[1] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[2] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.

[3] T. A. Ileni, D. L. Borza, and A. S. Darabant, "Fast in-the-wild hair segmentation and color classification." in *VISIGRAPP (4: VISAPP)*, 2019, pp. 59–66.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[5] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Express*, vol. 6, no. 4, pp. 312–315, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959519303455

[6] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[7] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernández, F.-M. De Rainville, C.-H. Weng, A. Ayala-Acevedo, R. Meudec, M. Laporte *et al.*, "imgaug," https://github.com/aleju/imgaug, 2020.

[8] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[9] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[10] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

Appendix

List of Figures

List of Tables