University of Denver
Ritchie School of Engineering and Computer Science

Department of Computer Science
UNIVERSITY OF DENVER

COMP 4531 Deep Learning Final Project

Authors:
Kendrick Choong

I. Problem Statement

We've invented a lot of tools that are capable of taking images and storing them at a very rapid pace. However, easily categorizing and digitizing qualitative data in those images still remains a manual task. Take for example the documentation of individuals in mug shots. Law enforcement typically need to fill out descriptors based on the individual such as eye color or hair color.

With the use of Convolutional Neural Networks, we can easily automate the documentation of image descriptors reducing the workload on individuals. This automation would help in a wide variety of fields especially in my field of work of State benefits.

The goal of this project is to see if something simple like hair color can be accurately extracted from an image. Further application of this concept would be extracting multiple features to build a more complete profile of image's characteristics to digitize. This could then be fed into more data analysis pipelines down in the future.

II. Dataset

The dataset used was from the CelebFaces Attributes Dataset on Kaggle created to help test models using face detection (https://www.kaggle.com/datasets/jessicali9530/celeba-dataset). This data set contains 202,599 images of celebrities' faces, 10,177 unique identities, 40 binary attribute annotations per image, and 5 landmark locations.

The files contained in the 1 GB downloadable zip with their descriptions are listed below.

1. **img_align_celeba.zip**: All the face images, cropped and aligned
2. **list_eval_partition.csv**: Recommended partitioning of images into training, validation, testing sets.
3. **list_bbox_celeba.csv**: Bounding box information for each image.
4. **list_landmarks_align_celeba.csv**: Image landmarks and their respective coordinates (left eye, right eye, nose, left mouth, right mouth).
5. **list_attr_celeba.csv**: Attribute labels for each image. There are 40 attributes. "1" represents positive while "-1" represents negative.

III. Data Cleaning

The images of celebrities themselves were already in a somewhat clean format. The data itself were all face images that were cropped and aligned. As seen in the 3 images below, most of the images represented pictures that were taken as headshots zoomed into similar aspect ratios.

What needed additional cleaning were the attributes of the data. Within the attributes file there are variables pertaining to facial features such as “attractive”, “blurry”, “chubby”, “bald”, “blonde hair”, etc. Given the scope of the project was limited to hair color, other features of the data were dropped. The data also did not provide data on all types of hair colors so the categories for classification were limited to “Bald”, “Black”, “Blonde”, “Brown”, and “Gray”.

The attributes Excel file was created to have a binary encoding of whether or not an attribute was present for a particular image, so there were some images that didn’t have any hair color attribute present so they were omitted from the dataset. The final count of images ended up being 128,980. Images were also sorted into a new file directory structure so that training and validation generators could be created for the models.

IV. Model Inputs/Outputs

Variable	Type	Range	Encoding
Images	Numerical array representing the image.	0-255	N/A
Hair Color	Binary	Bald, Black, Blonde, Brown, and Gray.	One-hot encoded

Model inputs and outputs are the same as the problem type is an image classification problem. So the variables being passed in are labeled/unlabeled images and the output is the classification of the type of image that was passed.

V. Neural Network Implementation

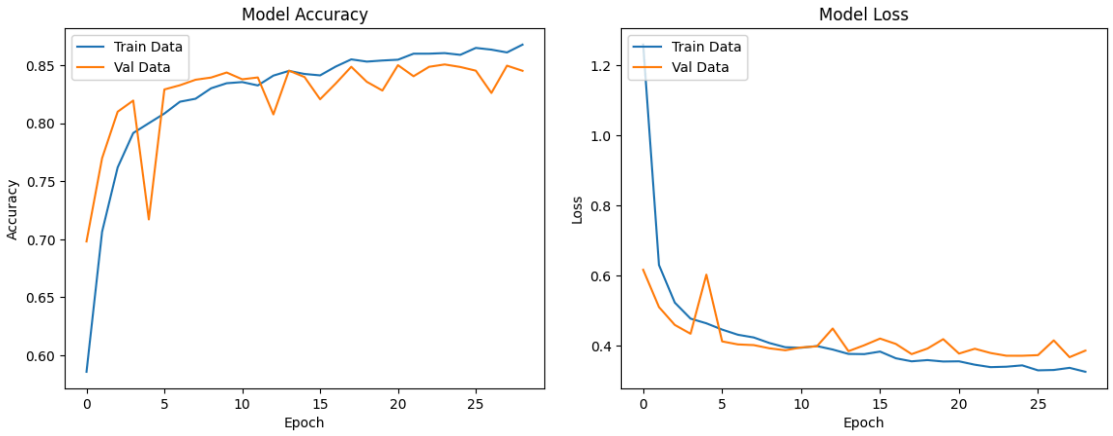
Given the intensity of training a CNN from scratch with such a big dataset of 100,000+ images, I chose to go a transfer learning route. For the two pretrained models, I chose to test the Inception V3 model and the VGG19 model.

In testing both models, the Inception V3 outperformed the VGG 19 model using a batch size of 32, 30 epochs, and 81 steps per epoch. The Inception V3 had a validation accuracy of 77.3% whereas the VGG 19 model had a validation accuracy of 73.7%. Given that the Inception V3 model was the best of the two, I chose to use the Inception model as my base model for transfer learning and to iterate off of it.

VI. Model Refinement

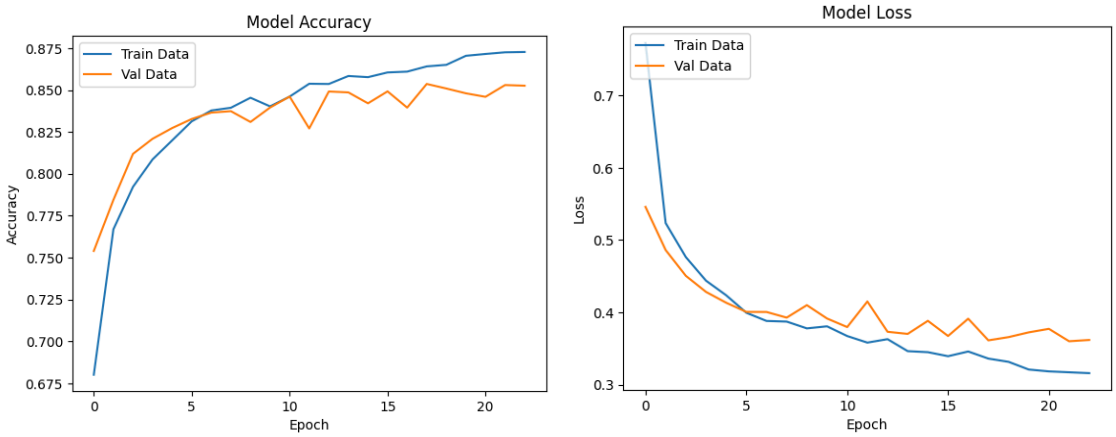
Once I decided which model to use as my baseline, I focused on adding varying the batch size of the model from 32. I used the steps 64, 128, and 255 to try and increase the training accuracy. When the model was run at different batch sizes, the accuracy increased to 84.0%, 80.9%, and 84.8%.

With the model at a batch size of 255 there was an increase of about 7% from the baseline batch size 32 Inception model to the 255 batch size Inception model. But at this model I noticed a bit of overfitting between the training and validation data as seen in the two figures below.

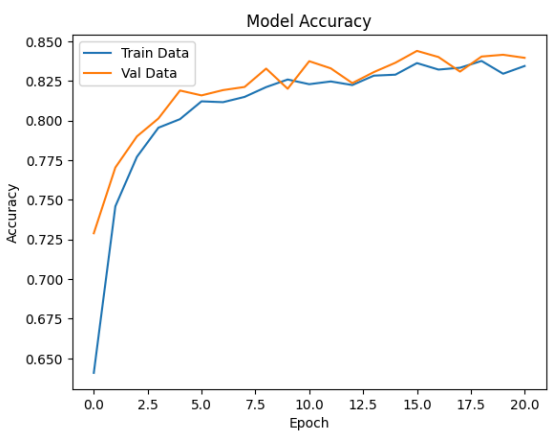


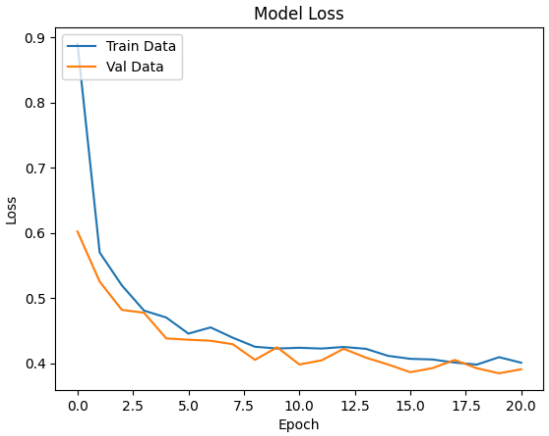
This prompted me to add a dropout layer to the CNN to see if I could get additional increases from the validation set. After adding a dropout layer at a dropout rate of 0.2, the performance of the model decreased a little bit to 84.52%. In the graphs below we see the general shape of the accuracy and loss over each epochs remain the same showing me the dropout had little impact on the model and perhaps adding in more dropout layers throughout the model or having a higher dropout rate could helped with the divergence of the two datasets.

I then tried to see if I could get more raw performance from the model by changing the optimizer to “Adam”, but I kept the dropout layer with the assumption the switch of optimizers could introduce more overfitting. Changing the optimizer did end up improving the performance of the training data with a small improvement (accuracy of 85.26%) to the validation data but there was a lot more overfitting as seen in the graphs below.



Given how much this model was overfitting with the new optimizer, I tried adding in data augmentation to help combat the overfitting of the training data by adding in image distortions. This model performed worse than even the Inception model with a batch size of 64. So the distortions added could’ve been a little too strong impacting the overall accuracy, but it did solve the overfitting issue as seen in the graphs below.





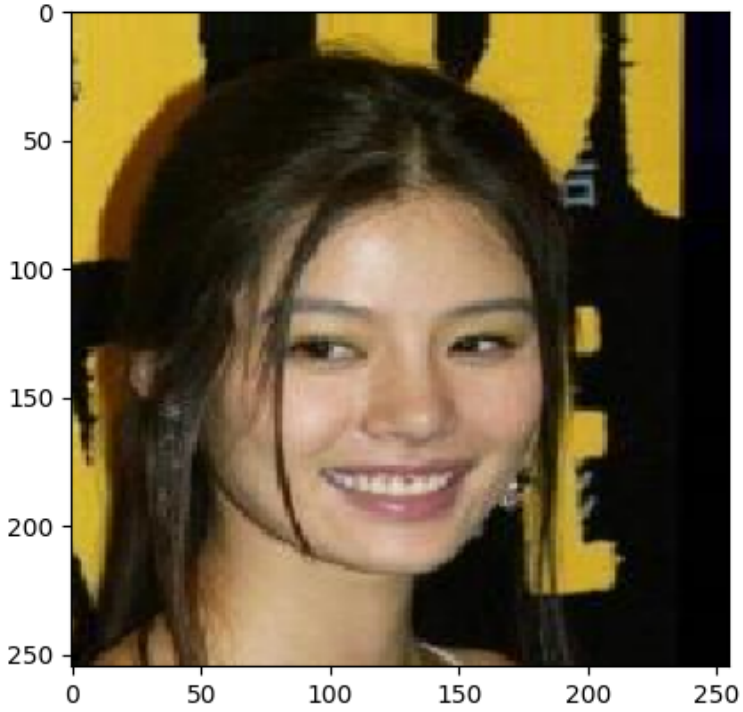
VII. Results & Limitations

The best model ended up being the Inception with the "Adam" optimizer with a validation accuracy of 85.26%. A table of all the model's performances can be seen below.

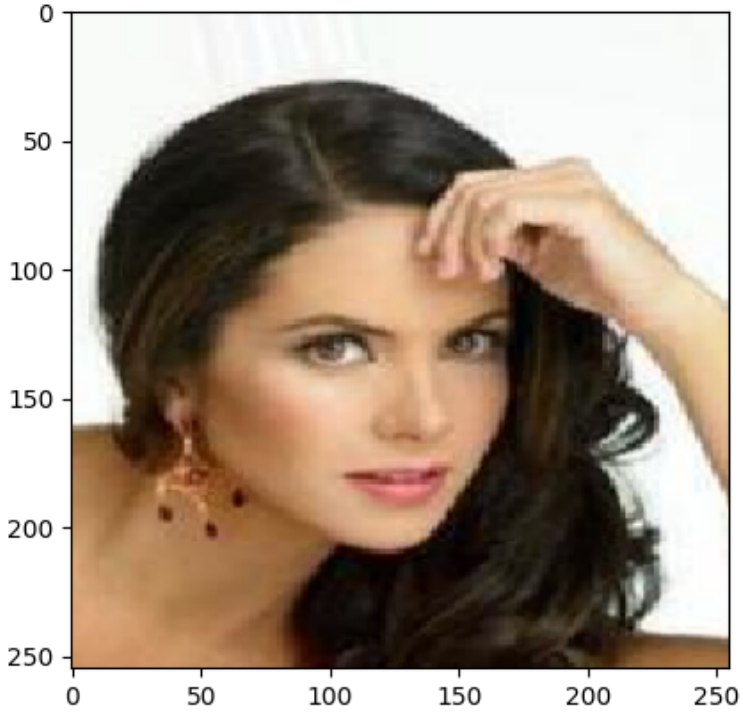
Model	Validation Accuracy
Baseline Inception V3	77.2622
Baseline VGG19	73.7497
Inception w/ BS_64	83.9846
Inception w/ BS_128	80.9607
Inception w/ BS_256	84.8724
Inception w/ BS_256 & Dropout	84.5235
Inception w/ Adam	85.2563
Inception w/ Data Augmentation	83.9498

So then taking the first 6 records of the validation dataset and running predictions of the hair color versus what they actually were, I found that it accurately predicted 3/6 images. But what's interesting is the images themselves and what the actual hair color was. Below is a representation of the images with their predictions versus actual labels.

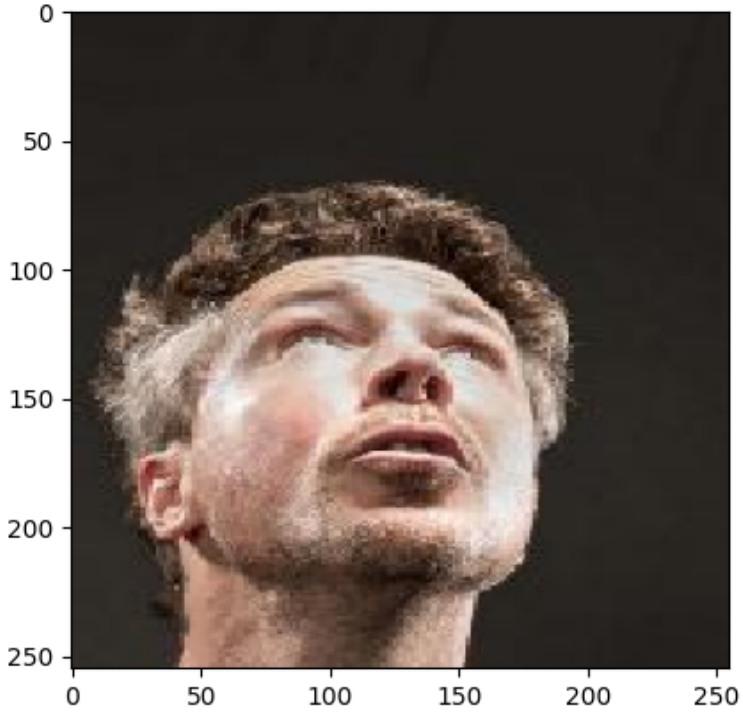
Prediction: Black, Actual: Brown



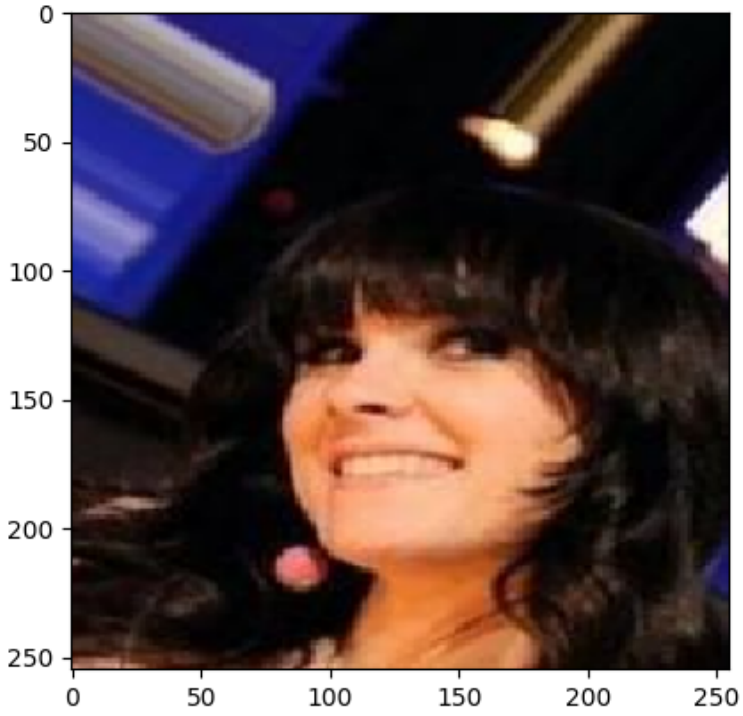
Prediction: Brown, Actual: Black



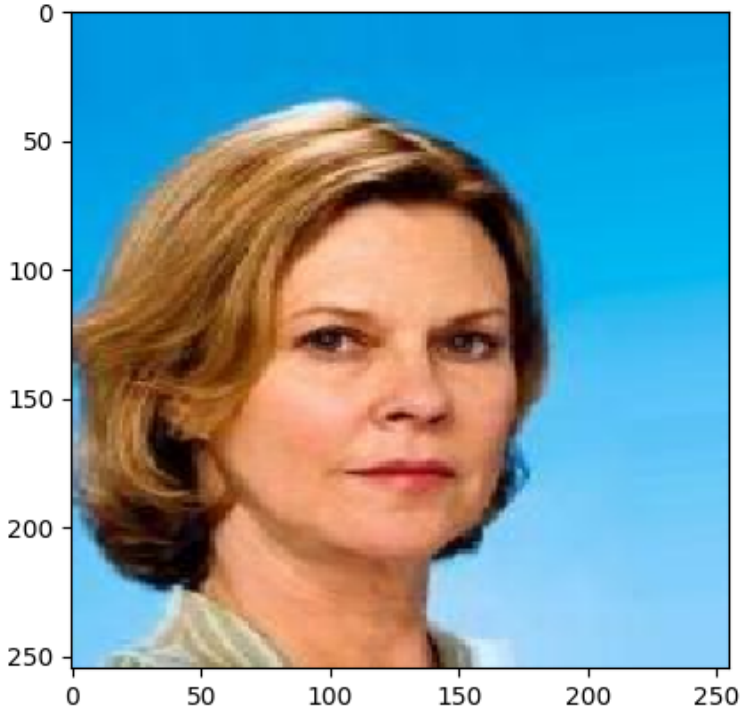
Prediction: Brown, Actual: Brown



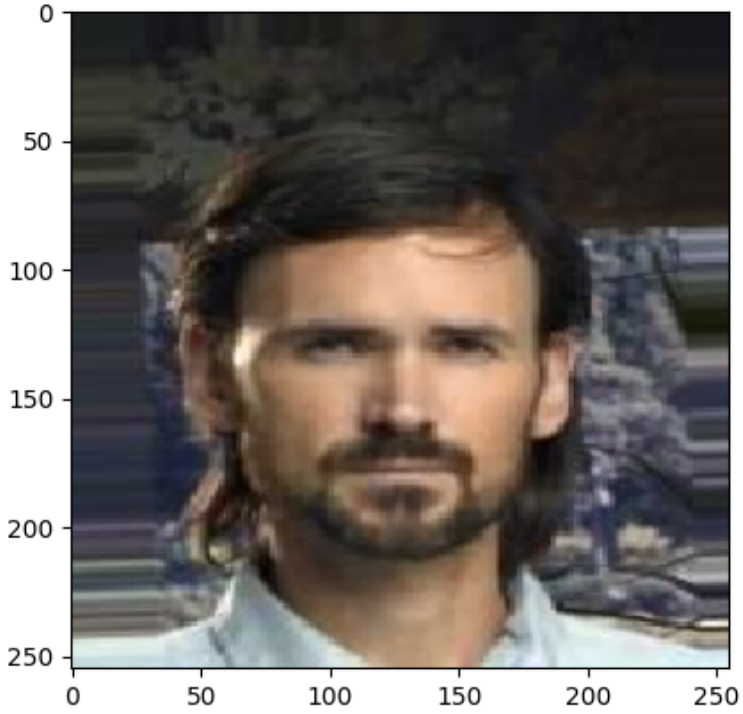
Predict: Black, Actual: Black



Prediction: Black, Actual: Brown



Prediction: Black, Actual: Black



Looking at these images there are some images that are pretty clear cut in what the hair color is. In the last image, the hair looks to be a black color. But with image 5, I would say their hand is

more blonde than it is brown. It could even be classified as a dirty blonde. There are even instances like in the first image where I would agree with the prediction over what the actual label says. Image one to me looks like their hair is black instead of brown.

To me this signals that the data labeling itself could be wrong which would in turn be a detriment to the model's performance. This could explain more about why the model was struggling to achieve accuracy percentages above 85%. It could learn how to classify a certain hair color one way but then get thrown off with the addition of the same hair color labeled incorrectly.

VIII. Conclusion

Overall I was able to create a model that was semi accurate in identifying the hair color of an image. While there were limitations imposed by the dataset and labeling itself, the core concept of image classification was achieved to a reasonable percentage.

A good accuracy would be very industry dependent, for certain use cases some businesses might find value in even being able to accurately identify elements of an image 85% of the time. Pair this with some additional feedback loops and validations, and you would be able to create a process that can digitize information from images that are likely to be accurate and then have the less likely images go through human intervention for proper documentation.

If anything, this has highlighted one of the biggest challenges of the data science field which is obtaining data and wrangling so that it's clean/usable. The model is only as good as what we feed it so it's paramount to try and develop data collection tools that are near perfect in labeling what we want. And even then taking the time with some human intervention to validate and improve the dataset for training is important even if it is costly.