Lab Exercise #3: What’s Their Hair Type?

Aaryanah Micah Consorte, Kyle Gwyneth Morales | 3CSC

I. INTRODUCTION

Each person is inherently unique, and this individuality extends to their hair type. Humans exhibit a diverse range of hair types, including straight, curly, and wavy hair.

Hair classification holds significant importance, not only for accurate styling but also for personalized grooming and understanding individual hair needs (Adhikari Lifeline, 2023). By categorizing hair types, individuals can identify the specific care regimens required to maintain healthy hair. This knowledge goes beyond mere aesthetics, but also serves as a way for individuals to nurture their hair.

In this particular lab exercise, a hair type classification model will be built through the use of Convolutional Neural Network (CNN). With the provided dataset, the students will explore how CNN works and how different experiments affect the performance of the model — such as using preprocessing methods, changing the filter and kernel sizes, implementing dropout and early stopping, changing optimizer, applying max pooling, etc.

II. METHODOLOGY

Step #1: Preparing the Data Set

Aside from the actual code that performs image checking and checks if the files are image files, the students also manually checked the data set provided. This step is crucial as the data set will also be used to let the model learn. Any mistakes or anomalies in the dataset may give false or inaccurate classifications. Specifically, the following steps are taken:

2.01 Scanning the files

The manual exploration of the data set has led to discovering duplicate files and incorrectly classified images. This step urges the students to do some manual re-classification before loading the dataset.

2.02 Renaming the images

Most of the images have long file names and have inconsistent naming conventions. The students renamed all the images by setting new file names based on their root folders and with a counter. The renaming is important not just due to presentation purposes but also to assist students in deciding where to transfer the incorrectly classified images and where they are initially located. The image below shows the updated filenames with the naming convention:
“SubFolder_N”

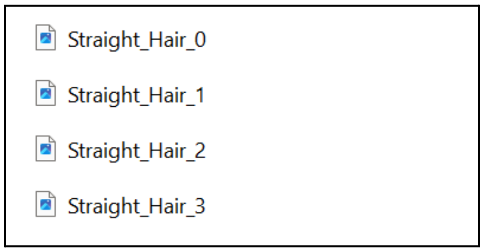

Straight_Hair_0
Straight_Hair_1
Straight_Hair_2
Straight_Hair_3

Figure 1. Naming Convention of Images

2.03 Manually re-classifying the images

Datasets are important as the CNN model depends on these during training. As mentioned during the scanning part, some images appear to be misclassified or are placed in an incorrect folder. This mistake can confuse the model or can be a hindrance in its learning process. As a solution, the students decided to first manually re-classify the images. This step is done by creating a new temporary folder called 'Incorrectly_Classified' and creating subfolders such as "move to curly folder", "move to straight folder", and "move to wavy folder" – which are also temporary folders and will be deleted once the images are moved to the correct folders.

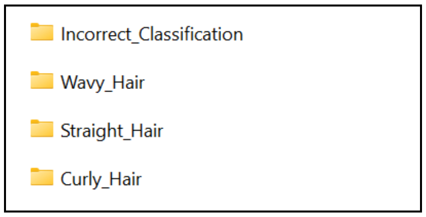


Figure 3. Temporary Folder Contents

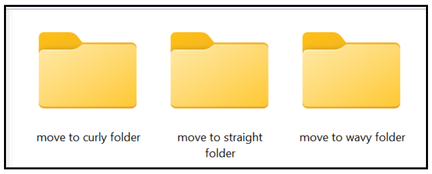


Figure 4. Temporary Sub-Folders Inside Incorrect_Classification

After this, the images are then moved to the correct folders and the temporary folders are deleted.

Before the validated dataset is loaded from the directory, the renaming of images is once again performed.

2.04 Removing Duplicates

To prevent potential data leakage due to duplicates, the students decided to delete every duplicate present in the dataset. This was done by using perceptual hashing. The students chose this over difference hashing due to its effectiveness in capturing nuanced similarities in images. This process of deletion is essential for enhancing the machine learning model's performance when encountering new data.

2.05 Deleting and Adding Images

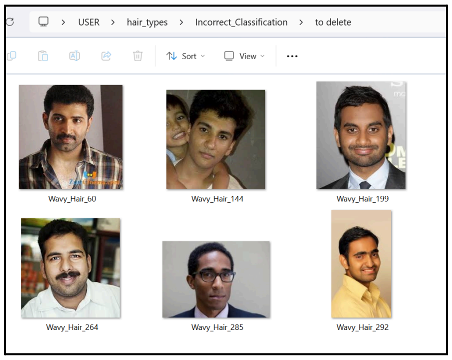


Figure 5. Images to Delete

Shown in Figure 5 are images with extremely short hair length. These images can introduce ambiguity into the classification process due to difficulties in identifying hair type, hence, they are deliberately excluded from the data set. This action is taken to ensure clarity in the classification task, thereby enhancing the effectiveness of the model's learning process.

Importantly, the decision to remove these images does not compromise the richness or diversity of the dataset. Instead, it serves to refine the dataset by eliminating potentially confusing instances, thereby optimizing the model's performance.

To maintain uniformity and balance within the dataset, and to compensate for deletion, new images are added to the dataset. This step also ensures that each hair type class is adequately represented with approximately 275 - 290 images.

After deleting and adding images, the accuracy of the model (both training and validation accuracy) slightly improved, indicating that this step is a good choice.

Step #2: Initial Run of the Model (Without Changes) and Analyzing the Metrics

In the initial model run, the students analyzed key metrics namely, training accuracy, training loss, validation accuracy, and validation loss to gauge the model's performance.

The primary goal is to achieve high validation accuracy and minimize losses, emphasizing the model's ability to generalize effectively. The selection of validation accuracy over training accuracy as the main basis is due to the former's nature of measuring the model's performance on unseen data. This ensures that it can reliably make predictions that are not on the training set. In addition, this helps in avoiding overfitting, where the model memorizes training data specifics rather than meaningful patterns.

Step #3: Conducting Multiple Experiments

In relation to the primary goal of minimizing the losses and achieving a high validation accuracy, the students has performed a total of 35 trials which include manipulation and/or application of the following:

- Filter Size
- Kernel Size
- L1 Regularization
- L2 Regularization
- Dropout Technique
- Early Stopping
- Max Pooling
- Batch Normalization
- Canny Edge Detection
- Sobel Edge Detection
- Learning Rate
- Optimizer Type
- Number of Epochs
- CNN Layers
- Image Size
- Regularization Strength

Having various experimentations gives valuable insights to the students and aids them in assessing the performance of the model in relation to the techniques/processes applied.

Step #4: Identifying which techniques improves the model's performance

Several techniques were employed to optimize the model's performance. This includes using L2 regularization with a strength of 0.2 to prevent overfitting, reducing dense layers for a streamlined architecture, and setting the filter sizes of the three convolutional layers to 16, 32, and 64 respectively to capture complex features. Consistently using a 3x3 kernel size in convolutional layers aids in feature extraction. The Adam optimizer with a learning rate of 1e-3 facilitates effective weight updates. Batch normalization stabilizes training, while early stopping and restoring of weights prevents overfitting. Dropout with a rate of 0.5 reduces overfitting during training. Max pooling focuses on important features by reducing dimensionality. Skipping preprocessing steps like normalization and edge detection allows the model to learn directly from raw data. Lastly, the image dimension is changed to 224x224 to enhance its resolution and prepare for better classification. These techniques collectively optimize performance and accuracy, adapting to specific task requirements and data characteristics.

Below shows the exact process of the model which performs best among other trials:

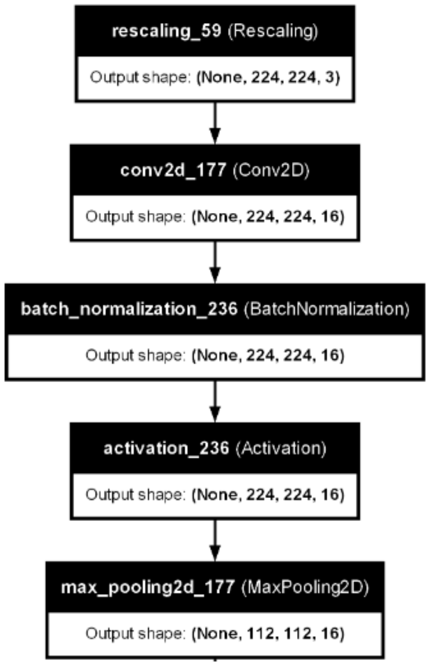

rescaling_59 (Rescaling)
Output shape: (None, 224, 224, 3)
conv2d_177 (Conv2D)
Output shape: (None, 224, 224, 16)
batch_normalization_236 (BatchNormalization)
Output shape: (None, 224, 224, 16)
activation_236 (Activation)
Output shape: (None, 224, 224, 16)
max_pooling2d_177 (MaxPooling2D)
Output shape: (None, 112, 112, 16)

Figure 6. CNN Model Process Part 1

The input image is first resized to a shape of 224x224 pixels with 3 color channels (RGB). A convolutional layer applies 16 filters to the input image, resulting in an output with spatial dimensions of 224x224 and 16 channels. Batch normalization is then applied to normalize the activations of the previous layer. An activation function is applied element-wise to the output of the batch normalization layer to introduce non-linearity. Subsequently, max pooling with a pool size of 2x2 is performed, reducing the spatial dimensions from 224x224 to 112x112 while retaining the same number of channels (16).

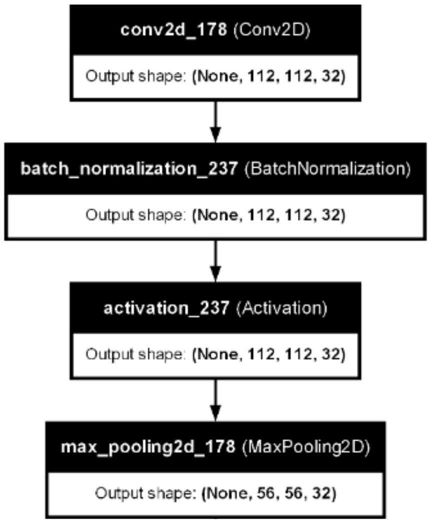

conv2d_178 (Conv2D)
Output shape: (None, 112, 112, 32)
batch_normalization_237 (BatchNormalization)
Output shape: (None, 112, 112, 32)
activation_237 (Activation)
Output shape: (None, 112, 112, 32)
max_pooling2d_178 (MaxPooling2D)
Output shape: (None, 56, 56, 32)

Figure 7. CNN Model Process Part 2

Then, another convolutional layer with 32 filters is applied, resulting in an output with spatial dimensions of 112x112, and 32 channels. Similar to the previous layer, batch normalization, an activation function, and max pooling follow, reducing the spatial dimensions to 56x56 while keeping the channel count at 32.

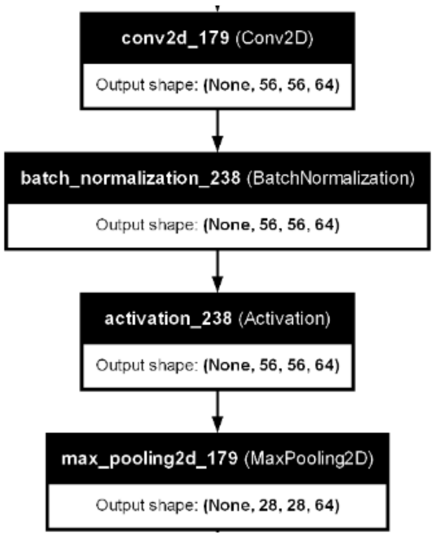

conv2d_179 (Conv2D)
Output shape: (None, 56, 56, 64)
batch_normalization_238 (BatchNormalization)
Output shape: (None, 56, 56, 64)
activation_238 (Activation)
Output shape: (None, 56, 56, 64)
max_pooling2d_179 (MaxPooling2D)
Output shape: (None, 28, 28, 64)

Figure 8. CNN Model Process Part 3

Similar process goes on in another convolutional layer but this time applying 64 filters, and reduces the dimension at the end to 28x28 with 64 channels.

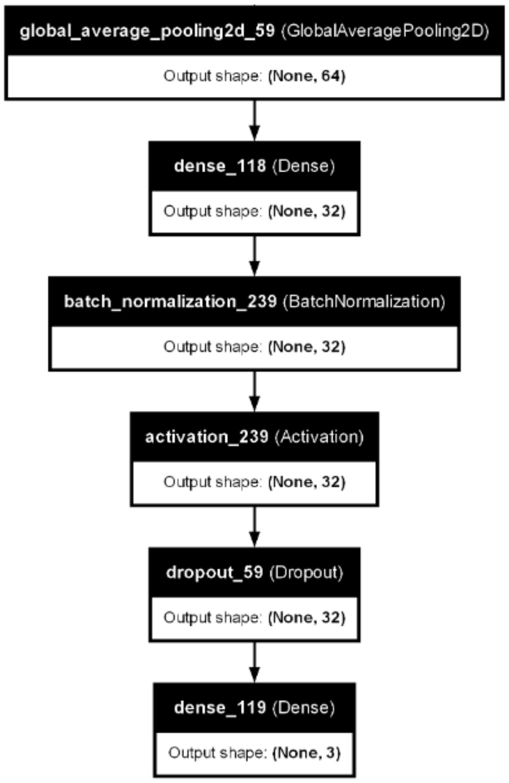


Figure 9. CNN Model Process Part 4

A global average pooling layer computes the average of each feature map, resulting in an output with 32 channels. Batch normalization and an activation function are then applied, maintaining the output shape. Subsequently, dropout is performed to regularize the network, and finally, a dense layer with 3 output units is applied, producing an output with 3 classes.

Step #5: Trying more predictions

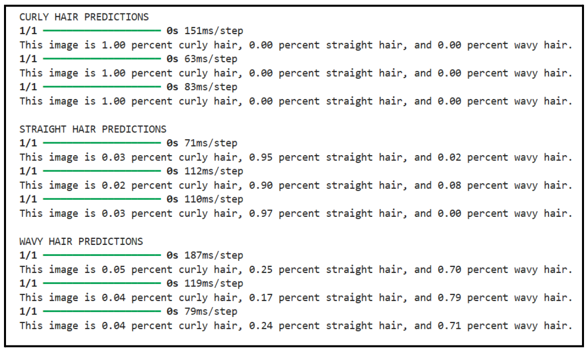
After conducting trials and finding out that the model performs well with the techniques mentioned in the 4th step, the students tried multiple predictions on different hair types. This step validates that the specific combinations of techniques/processes worked not just with predicting images with actual curly hair.

The students tried testing the model's predictions on a total of three (3) curly hair, three (3) straight hair, and three (3) wavy hair.

Below is the result of the predictions:

Table 1. Best Model's Predictions

Best Model		
Image Name	Actual	Predicted
Curly_Hair_1	curly	100% curly
Curly_Hair_150	curly	100% curly
Curly_Hair_300	curly	100% curly
Straight_Hair_1	straight	95% straight
Straight_Hair_150	straight	90% straight
Straight_Hair_300	straight	97% straight
Wavy_Hair_1	wavy	70% wavy
Wavy_Hair_150	wavy	79% wavy
Wavy_Hair_1300	wavy	71% wavy


```
CURLY HAIR PREDICTIONS
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 151ms/step
This image is 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step
This image is 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
This image is 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.

STRAIGHT HAIR PREDICTIONS
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step
This image is 0.03 percent curly hair, 0.95 percent straight hair, and 0.02 percent wavy hair.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step
This image is 0.02 percent curly hair, 0.90 percent straight hair, and 0.08 percent wavy hair.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
This image is 0.03 percent curly hair, 0.97 percent straight hair, and 0.00 percent wavy hair.

WAVY HAIR PREDICTIONS
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 187ms/step
This image is 0.05 percent curly hair, 0.25 percent straight hair, and 0.70 percent wavy hair.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 119ms/step
This image is 0.04 percent curly hair, 0.17 percent straight hair, and 0.79 percent wavy hair.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step
This image is 0.04 percent curly hair, 0.24 percent straight hair, and 0.71 percent wavy hair.
```

Figure 10. Best Model's Predictions

When making predictions, the students have observed that the model excels at predicting curly hair, performs moderately well with straight hair, and struggles the most with wavy hair. This could be attributed to the challenging nuances and structures present in images of wavy and straight hair strands, in contrast to the clearer and more defined characteristics of curly hair.

III. EXPERIMENTS

Table 2. Trials Conducted

TRIAL 0: Initial Result from Given Code		
0	**Epoch 49/50** accuracy: 0.5356 loss: 0.9319 val_accuracy: 0.4721 val_loss: 1.0109 **Epoch 50** accuracy: 0.5356 loss: 0.9057 val_accuracy: 0.4619 val_loss: 1.0115 **Prediction** 0.90 percent curly hair, 0.06 percent straight hair, and 0.04 percent wavy hair.	The initial result without changes
TRIAL 1: After modifying the data set		
1	**Epoch 49/50** accuracy: 0.5885 loss: 0.9034 val_accuracy: 0.5174 val_loss: 0.9542 **Epoch 50/50** accuracy: 0.4880 loss: 0.9734 val_accuracy: 0.5075 val_loss: 0.9740 **Prediction** 0.69 percent curly hair, 0.21 percent straight hair, and 0.11 percent wavy hair.	The val_accuracy increases BUT prediction gives a lower percentage of being curly.

TRIAL 2: Increasing the Epoch Epoch = 60		
2	**Epoch 59/60** accuracy: 0.5904 loss: 0.8477 val_accuracy: 0.4925 val_loss: 0.9770 **Epoch 60/60** accuracy: 0.5795 loss: 0.8627 val_accuracy: 0.4925 **Prediction** 0.81 percent curly hair, 0.18 percent straight hair, and 0.01 percent wavy hair.	The val_accuracy from trial 1 decreases BUT prediction gives a higher percentage of being curly.
TRIAL 3: Increasing Filter Size, Decreasing Kernel Size, Adding Dropout		
3	**Epoch 49/50** accuracy: 0.5381 loss: 0.9220 val_accuracy: 0.4826 val_loss: 0.9456 **Epoch 50/50** accuracy: 0.5431 loss: 0.9267 val_accuracy: 0.4776 val_loss: 0.944 **Prediction** 0.95 percent curly hair, 0.03 percent straight hair, and 0.02 percent wavy hair.	Does not make the validation accuracy higher BUT percentage of being curly in prediction increases.

TRIAL 4: Applying Batch Normalization, Applying Max Pooling, Decreasing Filter Size, Adding an activation function		
4	**Epoch 49/50** accuracy: 0.8592 loss: 0.3823 val_accuracy: 0.5572 val_loss: 1.2026 **Epoch 50/50** accuracy: 0.8557 loss: 0.3623 val_accuracy: 0.5025 val_loss: 1.473 **Prediction** 0.99 percent curly hair, 0.01 percent straight hair, and 0.00 percent wavy hair.	The accuracy is the highest among other trials. The val_accuracy also increases from Trial 3. It also gives a prediction that the hair is almost 100% curly. However, accuracy and val_accuracy are not balanced.

TRIAL 5: Trying RMSProp Optimizer		
5	**Epoch 49/50** accuracy: 0.8178 loss: 0.4714 val_accuracy: 0.5025 val_loss: 2.2902 **Epoch 50/50** accuracy: 0.8387 loss: 0.4044 val_accuracy: 0.5672 val_loss: 1.1253 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	The accuracy slightly decreases while there's only a minimal change with the val_accuracy. However, the loss are higher compared to trial 4. It gives a 100% prediction that hair is curly, which means the model is confident.

TRIAL 6: Adding Random Zoom		
6	**Epoch 49/50** accuracy: 0.6224 loss: 0.8279 val_accuracy: 0.4826 val_loss: 1.0178 **Epoch 50/50** accuracy: 0.6420 loss: 0.7875 val_accuracy: 0.5821 val_loss: 0.9136 **Prediction** 0.98 percent curly hair, 0.00 percent straight hair, and 0.01 percent wavy hair.	The model showed lower training accuracy and higher training loss compared to the final validation accuracy and validation loss at the last epoch. Despite the lower training metrics, the model demonstrated improved validation performance.

TRIAL 7: Adding Random Flip		
7	**Epoch 49/50** accuracy: 0.5607 loss: 0.8777 val_accuracy: 0.3831 val_loss: 1.3511 **Epoch 50/50** accuracy: 0.6446 loss: 0.7964 val_accuracy: 0.5572 val_loss: 0.9037 **Prediction** 0.99 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	The model demonstrated higher training accuracy but also higher training loss compared to the final validation accuracy and validation loss at the last epoch. Despite achieving higher training metrics, the model's performance on the validation dataset was lower, indicating potential overfitting.

TRIAL 8: Trying Flatten with validation split = 0.25		
8	**Epoch 49/50** accuracy: 0.7207 loss: 0.6041 val_accuracy: 0.5777 val_loss: 1.0438 **Epoch 50/50** accuracy: 0.7578 loss: 0.5819 val_accuracy: 0.5857 val_loss: 0.9808 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	Despite achieving higher training metrics than the previous trials, the use of Flatten was not recommended for the task, as it does not retain spatial information crucial for hair texture classification.

TRIAL 9: Setting validation split to 0.2		
9	**Epoch 49/50** accuracy: 0.6879 loss: 0.7391 val_accuracy: 0.5522 val_loss: 1.0499 **Epoch 50/50** accuracy: 0.7074 loss: 0.7140 val_accuracy: 0.6119 val_loss: 0.8758 **Prediction** 0.97 percent curly hair, 0.02 percent straight hair, and 0.01 percent wavy hair.	Although this trial has lower training metrics, the model showed improved validation accuracy and lower validation loss, indicating better generalization performance.

TRIAL 10: Adding early stopping		
10	**Epoch 33/50** accuracy: 0.6308 loss: 0.7425 val_accuracy: 0.5572 val_loss: 0.9326 **Epoch 34/50** accuracy: 0.6941 loss: 0.6882 val_accuracy: 0.5771 val_loss: 0.9000 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	Despite the decrease in training metrics, the validation accuracy remained relatively stable, and the validation loss slightly increased. This suggests that early stopping effectively halted training when the validation metric (loss) no longer improved, thus preventing the model from overfitting to the training data.

TRIAL 11: Setting dropout to 0.2		
11	**Epoch 39/50** accuracy: 0.7537 loss: 0.5623 val_accuracy: 0.5224 val_loss: 1.1489 **Epoch 40/50** accuracy: 0.7556 loss: 0.5414 val_accuracy: 0.5771 val_loss: 1.3019 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	The model showed improved training metrics compared to the previous dropout rate of 0.5. However, the validation accuracy remained the same, while the validation loss increased significantly compared to the previous trial. The observed decrease in validation loss indicates potential overfitting with the dropout rate of 0.2.

TRIAL 12: Setting dropout to 0.3		
12	**Epoch 19/50** accuracy: 0.6700 loss: 0.7329 val_accuracy: 0.5622 val_loss: 0.9266 **Epoch 20/50** accuracy: 0.6783 loss: 0.7096 val_accuracy: 0.5920 val_loss: 0.9249 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	The overall performance with a dropout rate of 0.3 was worse than that with a dropout rate of 0.5, as indicated by lower training metrics and higher validation loss.

TRIAL 13: Setting dropout to 0.4		
13	**Epoch 19/50** accuracy: 0.6332 loss: 0.7829 val_accuracy: 0.5274 val_loss: 1.0513 **Epoch 20/50** accuracy: 0.6583 loss: 0.7624 val_accuracy: 0.5821 val_loss: 0.9352 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	Similar to the previous trial, the performance with a dropout rate of 0.4 was worse than that with a dropout rate of 0.5. This suggests that a dropout rate of 0.4 may not be optimal for this certain scenario.

TRIAL 14: Setting dropout to 0.6		
14	**Epoch 19/50** accuracy: 0.5800 loss: 0.8666 val_accuracy: 0.4677 val_loss: 1.1280 **Epoch 20/50** accuracy: 0.6246 loss: 0.8337 val_accuracy: 0.5672 val_loss: 0.9317 **Prediction** 0.73 percent curly hair, 0.16 percent straight hair, and 0.11 percent wavy hair.	The performance with a dropout rate of 0.6 was significantly worse than that with a dropout rate of 0.5, as indicated by lower metrics and poor prediction performance. This suggests that a dropout rate of 0.6 may be too high, leading to increased underfitting and poorer model generalization.

TRIAL 15: Lowering the Learning Rate learning rate = 1e-4, dropout = 0.5		
15	**Epoch 9/50** accuracy: 0.4630 loss: 1.3268 val_accuracy: 0.3831 val_loss: 1.0574 **Epoch 10/50** accuracy: 0.4276 loss: 1.3495 val_accuracy: 0.3930 val_loss: 1.0480 **Prediction** 0.23 percent curly hair, 0.27 percent straight hair, and 0.50 percent wavy hair.	The accuracies are lower than the previous trials. The prediction is also wrong as it only predicts that hair is 23% curly.

TRIAL 16: Highering the Learning Rate learning rate = 1e-2		
16	**Epoch 9/50** accuracy: 0.5249 loss: 0.9866 val_accuracy: 0.4129 val_loss: 1.2463 **Epoch 10/50** accuracy: 0.5821 loss: 0.9104 val_accuracy: 0.4627 val_loss: 1.0618 **Prediction** 0.40 percent curly hair, 0.25 percent straight hair, and 0.34 percent wavy hair.	The accuracies increase from previous trials and the model gives a higher percentage of curly prediction but still worse than the original learning rate.

TRIAL 17: Applying Canny Edge Detection		
17	**Epoch 10/50** accuracy: 0.4340 loss: 1.0811 val_accuracy: 0.3950 val_loss: 15.7512 **Epoch 11/50** accuracy: 0.4500 loss: 1.0608 val_accuracy: 0.3850 val_loss: 28.3905 **Prediction** 0.00 percent curly hair, 0.00 percent straight hair, and 1.00 percent wavy hair.	The metrics are worse and the model gives wrong prediction.

TRIAL 18: Applying Sobel Edge Detection		
18	**Epoch 13/50** accuracy: 0.4228 loss: 1.0917 val_accuracy: 0.2900 val_loss: 13.0150 **Epoch 14/50** accuracy: 0.4451 loss: 1.0784 val_accuracy: 0.2800 val_loss: 23.9703 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	The metrics improved from the previous trial but the val_accuracy is still very low. However, the model once again gives a prediction of 100% curly hair.

TRIAL 19: Setting validation split to 0.3 and patience to 5		
19	**Epoch 5/50** accuracy: 0.5293 loss: 1.0121 val_accuracy: 0.3654 val_loss: 1.3046 **Epoch 6/50** accuracy: 0.5250 loss: 0.9919 val_accuracy: 0.3621 val_loss: 1.2991 **Prediction** 0.15 percent curly hair, 0.31 percent straight hair, and 0.54 percent wavy hair.	The training performance after 6 epochs showed limited improvement, with a training accuracy and training loss, leading to poor prediction performance. The model's performance suggests that further training epochs could potentially lead to improvements in both training and validation metrics.

TRIAL 20: Setting patience to 10		
20	**Epoch 11/50** accuracy: 0.6092 loss: 0.8605 val_accuracy: 0.4186 val_loss: 1.1087 **Epoch 12/50** accuracy: 0.6279 loss: 0.8281 val_accuracy: 0.4120 val_loss: 1.5650 **Prediction** 0.15 percent curly hair, 0.31 percent straight hair, and 0.54 percent wavy hair.	Despite getting higher training accuracy and lower training loss compared to the previous epoch, the validation performance deteriorated, suggesting potential overfitting.

TRIAL 21: Setting patience to 15		
21	**Epoch 38/50** accuracy: 0.6969 loss: 0.6533 val_accuracy: 0.5847 val_loss: 0.8965 **Epoch 39/50** accuracy: 0.7251 loss: 0.6111 val_accuracy: 0.5581 val_loss: 1.1334	The model's performance improved overall, with better metrics on training and validation datasets. The early stopping mechanism with a patience of 15 epochs effectively prevented overfitting and allowed the model to perform better on unseen data.

TRIAL 22: Setting validation split to 0.25		
22	**Epoch 40/50** accuracy: 0.6863 loss: 0.7460 val_accuracy: 0.5415 val_loss: 1.1542	Compared to the previous trial with a validation split of 0.3, this trial showed slightly lower training accuracy, higher training loss, lower validation accuracy, and higher validation loss. The decrease in model performance suggests that setting the validation split to 0.25 was not a good call.

TRIAL 23: Using Adam, setting dropout to 0.5, and setting patience to 15		
23	**Epoch 34/50** accuracy: 0.7006 - loss: 0.6725 - val_accuracy: 0.4700 - val_loss: 1.3727 **Epoch 35/50** accuracy: 0.6590 loss: 0.7103 val_accuracy: 0.6000 val_loss: 0.8580 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	The model's performance suggests that it is generalizing well on the validation dataset, as indicated by the relatively close values between training accuracy and validation accuracy. Additionally, the model's prediction for the given image was correct.

TRIAL 24: Applying L1 Regularization		
24	**Epoch 14/50** accuracy: 0.6367 loss: 1.5725 val_accuracy: 0.5450 val_loss: 1.6084 **Epoch 15/50** accuracy: 0.6184 loss: 1.5411 val_accuracy: 0.5600 val_loss: 1.6033 **Prediction** 0.92 percent curly hair, 0.05 percent straight hair, and 0.03 percent wavy hair.	The accuracy and val_accuracy values are close together (balance). Prediction's curly percentage is lower than the previous trial.
TRIAL 25: Applying L2 Regularization		
25	**Epoch 14/50** accuracy: 0.6095 loss: 0.9872 val_accuracy: 0.6150 val_loss: 0.9955 **Prediction** 0.99 percent curly hair, 0.01 percent straight hair, and 0.00 percent wavy hair.	The val_accuracy is quite lower than the previous trial but the percentage of curly hair is higher and almost close to 100.
TRIAL 26: No regularization technique removing the fully connected layer with 66 neurons		
26	**Epoch 14/50** accuracy: 0.5325 loss: 1.0985 val_accuracy: 0.5650 val_loss: 0.9486 **Epoch 15/50** accuracy: 0.5187 loss: 1.0686 val_accuracy: 0.5300 val_loss: 0.9722 **Prediction** 0.97 percent curly hair, 0.03 percent straight hair, and 0.01 percent wavy hair.	The val_accuracy is now higher than accuracy. The percentage of curly hair in prediction is also high.

TRIAL 27: Adding more CNN Layers		
27	**Epoch 14/50** accuracy: 0.5326 loss: 0.9734 val_accuracy: 0.4200 val_loss: 1.2510 **Epoch 15/50** accuracy: 0.5153 loss: 0.9902 val_accuracy: 0.4250 val_loss: 1.1450 **Prediction** 0.28 percent curly hair, 0.26 percent straight hair, and 0.46 percent wavy hair.	The val_accuracy decreases and the model makes a wrong prediction.
TRIAL 28: Removed image augmentation, added best model, setting patience to 20, and setting validation split to 0.25		
28	**Epoch 25/50** accuracy: 0.7233 loss: 0.6569 val_accuracy: 0.5566 val_loss: 0.9216 **Prediction** 0.97 percent curly hair, 0.03 percent straight hair, and 0.00 percent wavy hair.	The model's performance indicates higher training accuracy, lower training loss, higher validation accuracy, and lower validation loss compared to previous trials. Additionally, the best model was saved based on the highest validation accuracy achieved during training.

TRIAL 29: 4 CNN Layers, Filter = 16 -> 32 -> 64 -> 64		
29	**Epoch 8/50** accuracy: 0.6432 loss: 0.7434 val_accuracy: 0.4588 val_loss: 1.0133 **Epoch 9/50** accuracy: 0.6895 loss: 0.6733 val_accuracy: 0.5706 val_loss: 0.8949 **Prediction** 0.82 percent curly hair, 0.09 percent straight hair, and 0.08 percent wavy hair.	Model has worse predictions but better validation metrics.
TRIAL 30: 5 CNN Layers, Filter = 16 -> 32 -> 32 -> 64 -> 64, Pool size = (2,2), (2,2), (2,2), (1,1), (1,1), Kernel = all 5, Patience = 15		
30	**Epoch 17/50** accuracy: 0.7545 loss: 0.6131 val_accuracy: 0.3824 val_loss: 2.0431 **Epoch 18/50** accuracy: 0.7593 loss: 0.6092 val_accuracy: 0.5647 val_loss: 1.0801 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	Model has worse metrics but better predictions.

TRIAL 31: Changing Filter, Kernel, and Pool Sizes, Setting Patience to 20, Removing pre-processing methods
Filter = 16 -> 32 -> 64
Kernel = all 3
Pool = 2,2		

31	**Epoch 17/50** accuracy: 0.6864 loss: 0.7564 val_accuracy: 0.5896 val_loss: 0.8951 **Prediction** 0.98 percent curly hair, 0.01 percent straight hair, and 0.01 percent wavy hair.	The val_accuracy increases
TRIAL 32: Added L2 Regularization		
32	**Epoch 33/50** accuracy: 0.8480 loss: 0.3679 val_accuracy: 0.5802 val_loss: 1.6559 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	Higher accuracy but slightly lower val_accuracy
TRIAL 33: Decreasing the dense layer		
33	**Epoch 18/50:** accuracy: 0.6474 loss: 0.7603 val_accuracy: 0.6038 val_loss: 0.9043 **Prediction** 0.99 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	Higher val_accuracy and lower val_loss

Trial 34: Changing image size to (224, 224)		
34	**Epoch 38/50** accuracy: 0.7343 loss: 0.5972 val_accuracy: 0.6353 val_loss: 0.7725 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair	Higher val_accuracy and lowest val_loss
Trial 35: Changing regularization strength to 0.2		
35	**Epoch 47/100** accuracy: 0.6801 loss: 0.7465 val_accuracy: 0.6471 val_loss: 0.8898 **Epoch 79/100** accuracy: 0.7651 loss: 0.6092 val_accuracy: 0.6529 val_loss: 0.8 **Prediction** 1.00 percent curly hair, 0.00 percent straight hair, and 0.00 percent wavy hair.	Highest val_accuracy.

IV. INSIGHTS ON EXPERIMENTS

4.01 Effects of Changing Kernel Size and Filter Size

In CNNs, kernels are small matrices used for convolution operations, extracting features by sliding across input data and performing element-wise multiplications. Filters, also called convolutional filters or feature detectors, comprise collections of kernels applied to input data, producing feature maps that highlight specific patterns. These components are crucial in tasks like image recognition and object detection, allowing the network to learn hierarchical representations of complex data (Simonyan, K., & Zisserman, A. (2014).

In Trial 20, the students have modified the filter sizes starting from 16 to 32 to 64. This allows the model to capture more complex patterns as the depth increases. As per the kernel, a consistent 3x3 kernel/filter is used. Using 3x3 filters in convolutional neural networks helps extract various useful features and achieves top accuracy in tasks like image classification and object recognition (ArunaDevi Karuppasamy et al., 2019).

These changes as shown in the experiments table have increased both the training and validation accuracies of the model, proving that changing kernel and filter sizes is a good choice.

4.02 Effects of Applying Batch Normalization

Batch normalization is a technique in neural networks that helps keep the outputs between layers consistent. It's like giving each layer a fresh start with the data, making it easier for the network to understand (Deepchecks, 2022).

This technique acts as a form of regularization by adding noise to the network, which can help prevent it from memorizing the training data too closely. By performing normalization during training, it reduces the likelihood of large weight updates that can lead to overfitting (Ioffe, S., & Szegedy, 2015).

As shown in Trial 4, the trial that uses batch normalization generates the highest accuracy and also a high val_accuracy, indicating a good model performance.

4.03 Effects of Implementing Dropout

On the initial trials of this exercise, the testing accuracy is higher than the validation accuracy.

This case infers that there may be a case of overfitting. Dropout is a technique that is usually used to address overfitting.

Lower dropout rates such as 0.2, 0.3, and 0.4 generally exhibit higher validation accuracy but come with higher validation losses. This indicates potential overfitting or suboptimal generalization. Higher dropout rates like 0.6 result in lower validation accuracy and varied validation losses, suggesting underfitting. Dropout = 0.5 appears to have a better balance, providing good validation accuracy and a manageable validation loss, leading to the selection of this dropout rate as the constant throughout the following trials.

4.04 Effects of Max Pooling

Max pooling is a common feature extraction technique in CNN that reduces spatial dimensions by selecting the maximum value within defined windows or regions. This process preserves crucial features like edges and textures (Zhao & Zhang, 2024).

Implementing max pooling can be deemed as necessary when there is overfitting as it contributes to the regularization of data.

At Trial 4, after implementing max pooling, there has been a slight increase in training accuracy and validation accuracy, and a decrease in loss. This implies that the effects of max pooling in the performance of the model is positive and it contributes to better accuracy despite only having a small increase in accuracy values.

4.05 Effects of Different Optimizers

Optimizers are used in updating the weights and minimizing loss functions. As part of experimentation and trying to achieve better validation accuracy, different optimizers were tried. The initial optimizer used is Adam. RMSProp (Root Mean Square Propagation) is also tested, however, Adam optimizer still performs better, gathering the lowest losses.

Despite that, the best optimizer depends on the model itself. Some important factors to consider are model complexity, data characteristics, and optimization requirements. Experimentation with different optimizers, considering their strengths and weaknesses, is crucial for achieving optimal performance. For instance, while Adam typically performs well in training deep networks, RMSProp offers a balanced adaptiveness and simplicity. Ultimately, empirical evaluation guides the choice of optimizer to maximize performance for a given task.

4.06 Effects of Pre-Processing Methods

Pre-processing images for a classification model typically involves normalization, resizing, and augmentation (Deva, 2024). Normalization involves scaling pixel values to a common range (e.g., [0, 1] or [-1, 1]), ensuring numerical stability and preventing input bias. Resizing images to a uniform size ensures consistent input dimensions, facilitating batch processing and efficient training of the model. Augmentation techniques, such as random transformations (e.g., rotation, flipping, scaling), exposes the model to diverse variations of the input data.

4.06.1 Random Contrast

To enhance model adaptability to varying lighting conditions, random adjustments to contrast were applied, introducing variability by scaling pixel intensity values within specified ranges. This technique helps the model recognize features under different contrast levels and makes it more resilient to fluctuations in image quality and lighting conditions. However, despite the promising potential of this preprocessing technique, it did not yield satisfactory results during the trials, indicating that further optimization or alternative approaches may be needed for a better model performance.

4.06.2 Random Flip

Random flipping involves horizontally mirroring images during training to increase dataset diversity and exposes the model to varied orientations of the same image. In Trial 7, where random flipping was applied as an image augmentation technique, the model demonstrated improved training accuracy but lower validation accuracy, suggesting potential overfitting.

4.06.3 Random Zoom

Random zooming involves applying random scaling transformations to images during training, helping the model to handle variations in image scale. In Trial 6, which involved random zoom as an image augmentation technique, the model demonstrated improved validation accuracy (0.5821) and lower validation loss (0.9136) compared to the training accuracy (0.6420) and training loss (0.7875) at the final epoch. However, despite the improvement in validation performance, there was a discrepancy between training and validation accuracies, suggesting potential underfitting or the need for further optimization of the augmentation parameters.

4.06.4 Flatten

In Trial 8, where the Flatten layer was used, the model exhibited improved training accuracy (0.7578) but higher training loss (0.5819) compared to the final validation accuracy (0.5857) and validation loss (0.9808) at the last epoch. The Flatten layer was noted as not being suitable for retaining spatial information, such as the distribution of hair texture across the image, which is crucial for hair texture classification. Thus, the students decided to use GlobalAveragePooling2D instead, as it preserves spatial information more effectively. Despite achieving higher accuracy on the training set, the model's performance on the validation set suggests that using Flatten may have hindered its ability to generalize effectively to new data, particularly in tasks requiring spatial awareness and feature localization like hair texture classification.

4.07 Effects of Early Stopping

Early stopping is a regularization technique that is used to prevent overfitting during model training by stopping the training process when a specified metric stops improving. In this case, the students chose validation loss. This was chosen due to it being a good reflection of the model’s performance on new and unseen data. This ensures that the model doesn’t have excessive training, leading to overfitting.

4.08 Effects of Changing Validation Split

Validation split plays an important role in machine learning by facilitating unbiased model evaluation. Trials with different validation split sizes, like trial 22 (validation split = 0.3) and trial 26 (validation split = 0.25), demonstrate the impact on model performance. Trial 22 started off having promising metrics, however, it exhibited a decline in validation performance. Trial 26, on the other hand, consistently had poor performance on both training and validation stages.

4.09 Effects of Changing Patience

The "patience" parameter in early stopping determines the number of epochs without improvement in a monitored metric before training is stopped. A higher patience value allows more time for potential improvements but risks longer training times or overfitting, while a lower value can lead to faster training times but may stop it prematurely. Finding the optimal value for this parameter requires monitoring the metrics and finding the balance between training time and improvements in model performance (Brownlee, 2018).

4.10 Effects of Changing Learning Rate

Changing the learning rate of optimizers can help in avoiding overfitting and capturing more important patterns. High learning rate means data is updated in large amounts, while in low learning rate, parameters are updated at small amounts only (Smith, & Johnson, 2020).

Ideally, the objective is to have an ideal learning rate that avoids both overfitting and underfitting. At the 15th trial, the students tried a lower training rate, which resulted in an incorrect prediction. At Trial 16, an increased learning rate was tested, and while the metrics improved, using the optimal learning rate, which was somewhere between low and high, resulted in higher accuracies.

4.11 Effects of Using Canny and Sobel Edge Detection

The students also tried the use of edge detection such as Canny and Sobel edge detection. Edge detection is the process of finding an outline within the image. Initially, it was thought that applying edge detection tools will be beneficial in classifying the hair images. However, applying it appeared to not be effective in detecting the hair strands.

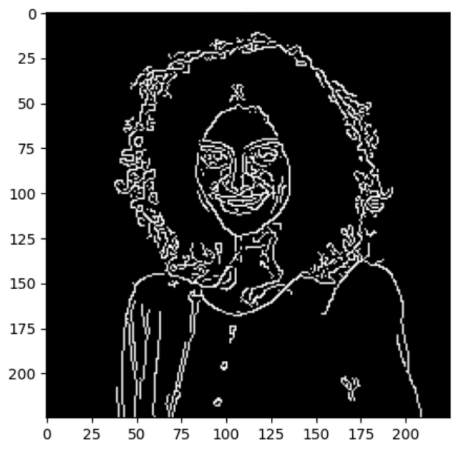


Figure 11. Canny Edge Detection

As observed from the image above, edges are successfully detected from the original image. However, it does not clearly show the hair patterns that make it curly.

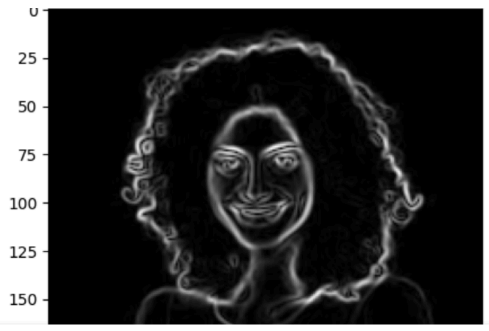


Figure 12.1 Sobel Edge Detection (1)

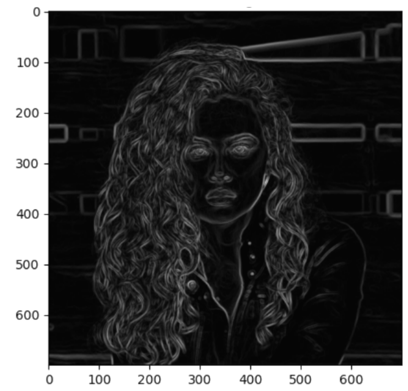


Figure 12.2 Sobel Edge Detection (2)

In comparison to Canny Edge detection, the image generated using Sobel detector seemed to be clearer and more successful in showing the hair strands and hair formation.

However, analyzing the model's metrics and its predictions reveals that employing edge detection tools proves ineffective. In Trial X, it's evident that utilizing Canny Edge Detection leads to an incorrect prediction. Despite achieving a correct prediction (100% curly) with Sobel Edge Detection, the model's testing and validation accuracies remain notably low, accompanied by high losses. These findings suggest that the model's performance deteriorates when edge detection techniques are implemented.

The inefficacy of edge detection tools in hair type classification may stem from image characteristics. While they can identify edges aiding in discerning hair type (curly, straight, or wavy), their applicability across all images is not given. Variations in image angles, image lighting, hair colors, or the presence of extraneous elements detract from their utility. These factors may contribute to the ineffectiveness of edge detection tools in the experimental trials. It is recommended to conduct further research aimed at enhancing the capabilities of edge detection tools to more effectively capture hair details within images.

4.12 Effects of Adding Regularization and Modifying its Strength

Given that the early trials have higher validation accuracy than testing accuracy, overfitting may still persist even after implementing dropout. In an attempt to provide a solution to that, L1 and L2 Regularization are applied.

Based on experimentation, L2 regularization results in better model performance and a higher percentage of correct prediction in comparison to L1 regularization.

During Trial 32, an enhancement in validation accuracy to 0.58 was noted upon implementing L2 regularization with a strength of 0.01. Subsequently, at Trial 35, elevating the regularization strength to 0.2 resulted in a further improvement to 0.65.

One possible reason why L2 regularization worked better is because it effectively penalizes large weights in the model and helps to smooth out the model's decision boundaries, making it less sensitive to noisy or irrelevant features in the data. Moreover, a higher regularization strength works due to its ability to strike a balance between reducing overfitting and preserving important features in the model.

4.13 Effects of Restoring Best Model’s Weights

The “best model” feature involved setting the weights back to the epoch with the highest validation accuracy. This feature, implemented in trial 28, led to notable improvements in validation accuracy and prediction performance, despite some variations in training and validation metrics. The model’s ability to predict hair type percentages, specifically curly hair, showed more accurate and consistent performance across the trials compared to previous iterations without this feature.

4.13 Effects of Increasing Image Dimensions

The choice of image size plays a critical role in the performance of Convolutional Neural Network (CNN) models for image classification (Semma et al., 2021). This was observed during the students' trials when the validation accuracy increased between trials 33 and 34. Additionally, besides achieving better classification, a significant difference in training time was also noted. Specifically, the training time increased from approximately 30-60 ms/step to around 300-450 ms/step.

4.14 Other Notable Observations

Two notable observations from the experiments are the performance variability seen across trials and the varying levels of prediction confidence. The performance variability highlights the sensitivity of neural networks to initializations, data splits, and parameter settings, resulting in fluctuations in accuracy, loss, and validation metrics throughout training epochs. This shows the non-deterministic nature of training deep learning models.

Additionally, the experiments demonstrate that higher validation accuracy does not necessarily translate to confident predictions, particularly in tasks like hair type recognition. Different model configurations yield varying levels of prediction confidence, emphasizing the importance of not solely relying on validation metrics but also assessing the model's certainty in its predictions for real-world applications.

V. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

The students have focused on increasing the validation accuracy and lowering the losses. According to research, having a high training accuracy is good since it means that the model is learning well, however, when it is higher than the validation accuracy, it means the model is overfitting. On the other hand, higher value of losses indicates that there's a huge difference between the target and actual values. The ideal value of loss is close to zero to imply better model performance.

As an attempt to address overfitting, the students perform data augmentation, regularization, and batch normalization, which are all suggested techniques to not overfit the data. While different modifications are applied, it has still become a challenge for students to increase the validation accuracy.

In addition to that, several experimentations are also done such as trying different optimizers, testing different learning rates, applying dropout technique, using L1 and L2 regularizations, adding CNN layer, implementing Canny and Sobel edge detectors, changing validation split, changing kernel and filter sizes, and applying early stopping. However, the performance of the model is not improving much, during the early trials and validation accuracy mostly ranges around 0.5 to 0.6.

One possible reason as to why the students find it challenging to increase the validation accuracy is the data set size. Despite adding more images to the data set, having an approximate of 800 samples may still not work best with the model given that CNN is typically designed for high dimensional inputs.

Some notable modifications that have increased the model's validation accuracy at Trial 30-35 are increasing the image size to 224x224, changing Kernel to increasing sizes starting at 16, and using L2 regularization at 0.2 strength. The **highest validation accuracy that the model has achieved across all trials is <u>0.6529</u> at Trial 35**, with a corresponding validation loss of 0.8. Upon reaching the highest validation accuracy, the weights from the epoch associated with this highest validation accuracy are restored to ensure that it maintains its optimal performance. The purpose of restoring the weights from the best epoch is to preserve the model's learned representations and parameter configurations that led to its highest validation accuracy. By reverting to these weights, the model can continue to make accurate predictions.

With regards to predictions, the model excels in identifying curly hair, typically labeling selected images with full confidence (100%). In categorizing straight hair, the prediction for some images ranges from 90% to 97% straight hair. Among the three hair types, the model struggles most with predicting wavy hair, often assigning confidence scores of only 70% to 79%. This difficulty arises from the ambiguity in wavy hair, which sometimes falls between the distinctly straight and curly categories. Also, straight hairs may exhibit small curves that are not curvy enough to qualify as wavy. The nuanced distinctions can contribute to lower prediction percentages for straight and wavy hair compared to curly hair.

To develop a model that performs better in classifying hair types, it is recommended to use more data samples for each class. Other pre-processing methods can also be experimented, specifically the use of methods that isolates the hair from the rest of the image and an improved edge detection tool that can capture all hair patterns and strands.

Moreover, classifying hair types may be a complex task as some curl patterns and hair textures are confusing. Sometimes, hair styling or hair length affects the appearance of the hair in the images. While CNN may be a good baseline for this task, exploring other architectures may be beneficial if the goal is deeply focused on improving the model's ability to perform hair type classification. This lab exercise, however, remains centered on grasping CNN fundamentals, and the students have undertaken numerous trials, experimenting with different techniques and processes — which all fosters a deeper understanding of CNN principles and the associated underlying concepts.

VI. REFERENCES

Adhikari Lifeline. (2023, December 28). DIFFERENT HAIR TYPES THAT YOU MUST KNOW |. Adhikarilifeline.com. https://www.adhikarilifeline.com/different-hair-types/#:~:text=Determining%20your%20hair%20type%20is,fine%2C%20medium%2C%20or%20thick.

ArunaDevi Karuppasamy, Abdelhamid Abdessalem, Rachid Hedjam, & Zidoum, H. M. (2019). Investigating Existing Techniques for Designing Convolutional Neural Network Filters. https://doi.org/10.1109/icccnt45670.2019.8944816

Brownlee, J. (2018, December 9). Use Early Stopping to Halt the Training of Neural Networks At the Right Time. Machine Learning Mastery. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

Deva, J. (2024). Impact of the Preprocessing Steps in Deep Learning-Based Image Classifications. National Academy Science Letters. https://doi.org/10.1007/s40009-023-01372-2

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456).

Khanna, S. (2023, November 16). A Comprehensive Guide to Train-Test-Validation Split in 2023. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2023/11/train-test-validation-split/

Semma, A., Lazrak, S., Hannad, Y., Boukhani, M., & Kettani, Y. E. (2021). WRITER IDENTIFICATION: THE EFFECT OF IMAGE RESIZING ON CNN PERFORMANCE. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLVI-4/W5-2021, 501–507. https://doi.org/10.5194/isprs-archives-xlvi-4-w5-2021-501-2021

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Smith, J., & Johnson, A. (2020). The impact of learning rate on overfitting in machine learning algorithms. Journal of Machine Learning Research, 21(3), 456-472.

Zhao, L., & Zhang, Z. (2024). A improved pooling method for convolutional neural networks. Scientific Reports, 14(1). https://doi.org/10.1038/s41598-024-51258-6