Abrigo, Nathanael Chris O.
Salvador, Fabian III R.

Laboratory Exercise #3 - Hair Type Classification using Convolutional Neural Networks

I. Introduction

According to Rosebrook (2021), Image Classification is one of the primary tasks in computer vision, which aims to categorize or label the images based on their visual contents. This involves using machine learning algorithms, mainly Convolutional Neural Networks (CNNs), in order to extract relevant features from the inputted images and map them to specific categories. According to Sharma (2024), CNNs are commonly used for image classification as CNNs can learn hierarchical features like edges, textures, and shapes, enabling accurate object recognition in images. With the process of training through the use of labeled dataset, the model/s learn to recognize patterns to distinguish between different categories.

This task aims to classify between three different hair types namely: Wavy, Curly, and Straight Hair. The dataset provided include images of the mentioned hair types. However upon inspection, there are hair images that are inconsistent to its assigned type. For example, one of the wavy hair images looks curly. But the researchers are tasked to assume all data provided are correct.

II. Methodology

1. Installing & Importing

The researchers utilized the following libraries:

- tensorflow
- numpy
- pandas
- matplotlib

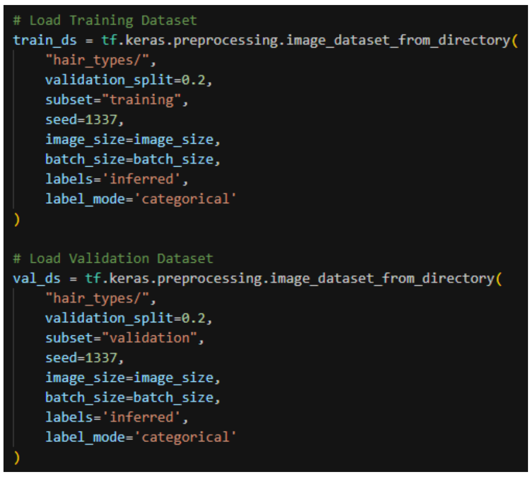
The tensorflow library is the most used for this project. This library allows the program to load the dataset, process the image, and create the CNN model for the classification of hair types. The other mentioned libraries which are numpy, pandas, and matplotlib were used to learn more about the given dataset. These libraries helped the researchers view and analyze the data.

2. Data Loading

For loading the dataset, the researchers first checked if the files contained in the mentioned directory indeed contained images specifically "bmp", "gif", "jpeg", and "png" image format. As these four image formats are accepted by tensorflow.

Afterwards, as mentioned, the researchers utilized the use of the tensorflow library to ensure that the data can be loaded before starting to experiment the given dataset.

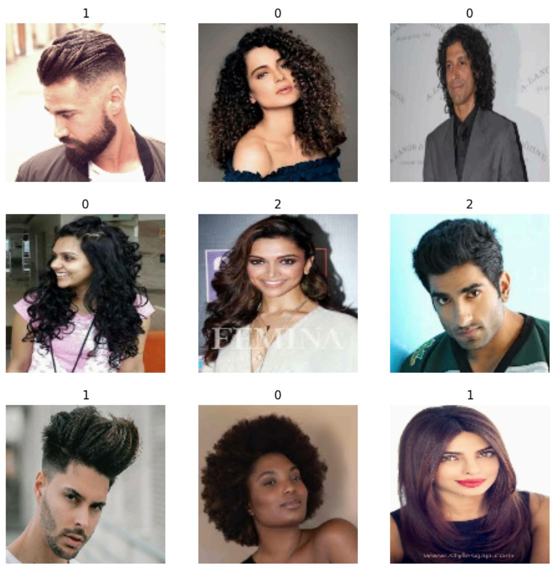
This line of code splits the dataset between the training and validation subset and applies some preprocessing for the images, specifically setting the image size and batch size for both subsets.



```
# Load Training Dataset
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "hair_types/",
    validation_split=0.2,
    subset="training",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
    labels='inferred',
    label_mode='categorical'
)

# Load Validation Dataset
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "hair_types/",
    validation_split=0.2,
    subset="validation",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
    labels='inferred',
    label_mode='categorical'
)
```

3. Data Visualization

For visualizing the data, as mentioned, the researchers utilized libraries such as matplotlib to view some images.



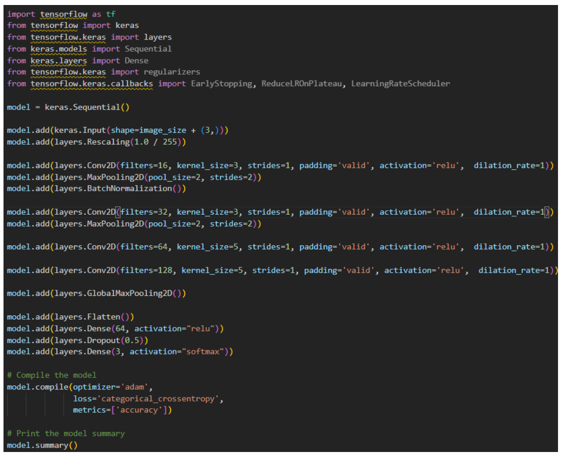
Here is one sample view of the images that the dataset contained. The code for this section displays nine images in random in order to view different images present in the dataset. This code is also used to display augmented images to see how the augmented images look like.

4. Model Architecture

The researchers build a model based on Convolutional Neural Networks (CNNs) for the task of classifying images, specifically hair types. The CNN model architecture the researchers manipulated consists of different layers which includes the input layer, convolutional layers, pooling layers, and dense layers.

The input layer of a CNN takes in the raw inputs. The Convolutional layers are responsible for extracting features from the inputted images through the use of filters, which detect patterns at different spatial scales. Pooling layers reduce the spatial dimensions of the feature maps, aiding in capturing the most important features while reducing computational complexity. Dense layers, also known as fully connected layers, further process the extracted features to produce the final classification output.

The researchers also applied the use of some regularization and normalization techniques in the model's architecture, specifically the use of dropout and batch normalization which is to prevent overfitting and improve generalization performance.



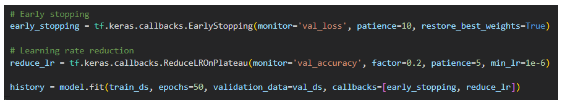
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras import regularizers
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, LearningRateScheduler

model = keras.Sequential()

model.add(keras.Input(shape=image_size + (3,)))
model.add(layers.Rescaling(1.0 / 255))

model.add(layers.Conv2D(filters=16, kernel_size=3, strides=1, padding='valid', activation='relu',  dilation_rate=1))
model.add(layers.MaxPooling2D(pool_size=2, strides=2))
model.add(layers.BatchNormalization())

model.add(layers.Conv2D(filters=32, kernel_size=3, strides=1, padding='valid', activation='relu',  dilation_rate=1))
model.add(layers.MaxPooling2D(pool_size=2, strides=2))

model.add(layers.Conv2D(filters=64, kernel_size=5, strides=1, padding='valid', activation='relu',  dilation_rate=1))

model.add(layers.Conv2D(filters=128, kernel_size=5, strides=1, padding='valid', activation='relu',  dilation_rate=1))

model.add(layers.GlobalMaxPooling2D())

model.add(layers.Flatten())
model.add(layers.Dense(64, activation="relu"))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(3, activation="softmax"))

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Print the model summary
model.summary()
```

5. Model Training

Model training is a crucial phase in machine learning where a model learns patterns and relationships within data to make accurate predictions or classifications.

Early Stopping is a preventative measure against overfitting and excessive computational expenditure. It scrutinizes the validation loss and halts training if there's no improvement for 10 consecutive epochs, preserving the best weights.

Learning Rate Reduction dynamically adjusts the learning rate based on validation accuracy performance. If validation accuracy is stagnant over 5 consecutive epochs, the learning rate is reduced by a factor of 0.2, preventing it from going below the minimum threshold of 1e-6.



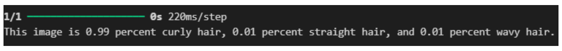
```python
# Early stopping
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Learning rate reduction
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy', factor=0.2, patience=5, min_lr=1e-6)

history = model.fit(train_ds, epochs=50, validation_data=val_ds, callbacks=[early_stopping, reduce_lr])
```

These techniques synergistically optimize the training process, ensuring model convergence while guarding against overfitting, thus enhancing the efficacy and efficiency of the CNN training regime.

6. Evaluation

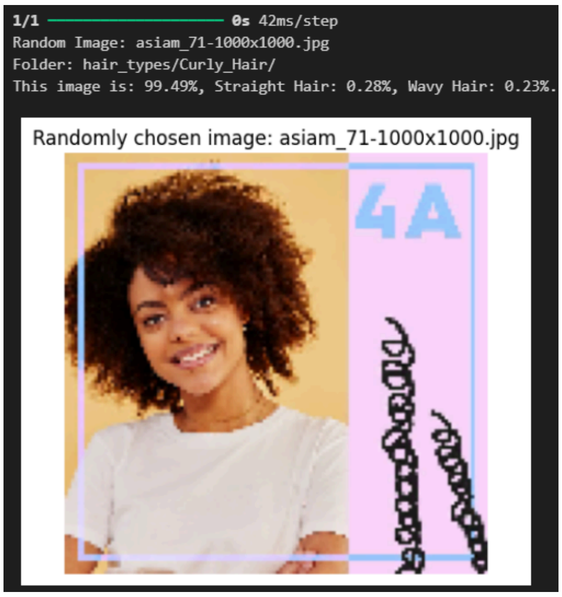
In the model evaluation phase, several key steps were undertaken to assess the performance of the trained model.

Firstly, the model's capability to predict the hair type of a selected image was evaluated, providing insight into its accuracy and ability to classify specific examples.



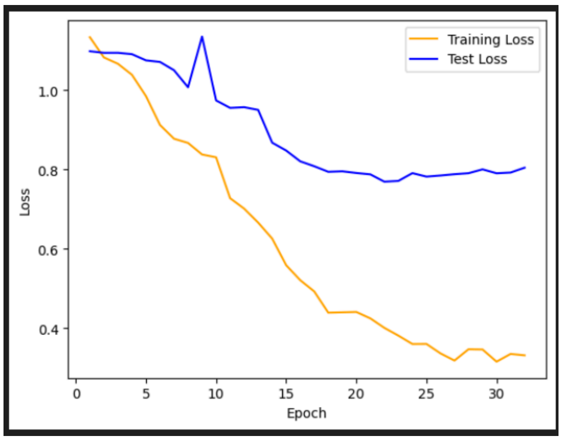
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 220ms/step
This image is 0.99 percent curly hair, 0.01 percent straight hair, and 0.01 percent wavy hair.
```

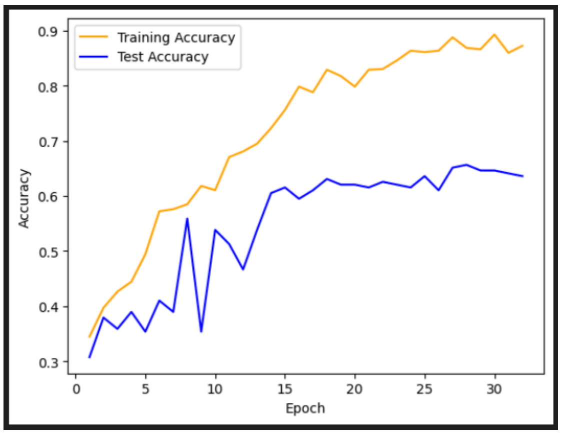
Additionally, the model's generalization performance was tested by predicting the hair type of a randomly chosen image, allowing for assessment of its effectiveness across varied input data.



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
Random Image: asiam_71-1000x1000.jpg
Folder: hair_types/Curly_Hair/
This image is: 99.49%, Straight Hair: 0.28%, Wavy Hair: 0.23%.
```

Furthermore, visualizations such as line graphs depicting the training and validation loss as well as accuracy metrics were generated. These graphs provided a comprehensive overview of the model's learning process, highlighting trends in performance over epochs and helping to identify potential areas for improvement.





By combining quantitative predictions with visual representations of performance metrics, the model evaluation phase facilitated a thorough assessment of the CNN model's capabilities in hair type classification.

III. Experiments

1. Convolutional Layers

The researchers manipulated several architectural aspects, including the number of layers, the filters within those layers, kernel sizes, strides, and padding. By systematically exploring different configurations, they aimed to identify the architecture that strikes the optimal balance ultimately improving the model's performance.

Firstly, The researchers varied the number of convolutional layers and its filters to find the ideal complexity of the model. Experimented with configurations such as (16, 32, 64), (32, 64, 128), and (32, 64, 128, 256), altering the network and the number of filters generated at each layer.

Moreover, the researchers adjusted the kernel size across different convolutional layers, testing variations such as (7, 5, 7), (3, 5, 5, 3), and (3, 3, 3). The kernel size determines the receptive field of the filters and directly impacts the level of detail captured by the network. By experimenting with different kernel sizes, the researchers sought to understand how variations in kernel size affect the model's ability to capture intricate features present in images of varying complexities.

Additionally, they explored different values for the number of strides (1, 2) and padding techniques ('same' and 'valid'). Strides determine the step size of the filter's movement during convolution. Padding techniques, such as 'same' and 'valid', control how the input is padded before applying the convolution operation, influencing the spatial dimensions of the output.

2. Max Pooling

The researchers explored the optimal placement of Max Pooling. They varied the placement of Max Pooling considering configurations where these were added after each convolutional layer. By assessing classification performance across these configurations, the researchers aimed to identify the arrangement that strikes a balance.

3. Normalization

The researchers conducted experiments to determine the most effective normalization technique and its optimal placement. They compared the performance of two normalization methods: L2 normalization and Batch Normalization. L2 normalization involves scaling the activations of neurons to have a unit L2 norm, while Batch Normalization normalizes the activations of each layer by adjusting them to have a mean of zero and standard deviation of one, applied over mini-batches of data. The researchers varied the application of these normalization techniques across convolutional layers of the network.

4. Dense Layers

The researchers explored the configuration of dense layers within their model. They adjusted the size of the dense layers, experimented with flatten, and varied the incorporation of dropout regularization. By modifying the size of dense layers, the researchers aimed to fine-tune the model's capacity to learn complex patterns and relationships in the extracted features. Additionally, the inclusion of flatten layers facilitated the transition from the convolutional layers to the dense layers, reshaping the feature maps into a more suitable format. Furthermore, the integration of dropout regularization mitigated overfitting by randomly deactivating a fraction of neurons during training, promoting the learning of more robust and generalizable features.

5. Data Augmentation

The researchers incorporated the use of data augmentation. This is to add diversity in the present dataset by applying transformations which may help in introducing variability in the designed model. The researchers experimented with different variations that can be done with the images. The researchers decided to augment the images by flipping, rotating, adjusting brightness and the contrast of the images.

6. Regularization

The researchers experimented using regularization techniques to attempt to improve the metrics the model can provide. As mentioned, the researchers attempted inserting dropout within the model to attempt to lessen the overfitting present. The researchers also experimented inputting an early stopping and learning rate scheduler for the callbacks of the model. The early stopping stops the training if it is no longer improving. The learning rate scheduler to adjust the learning rate of the model dynamically while training.

7. Fine-Tuned Models

The researchers tried using a pretrained model specifically VGG16 which is one of the CNN architecture models. It is surprising to see a very high accuracy result with at least 0.9. However, the researchers are tasked to attempt to create a model that does not rely on pretrained models to further understand how CNNs works.

IV. Results

With the final results of

```
accuracy: 0.8730
loss: 0.3233
val_accuracy: 0.6359
val_loss: 0.8043
```

This metrics provides insights into the effectiveness of the model to classify images of different hair types. The achieved accuracy of 87.30% indicates that the model correctly classified approximately 87.30% of the images in the training set. This demonstrates a reasonable level of proficiency of the model in terms of learning the patterns and features associated with the various hair types present in the training data.

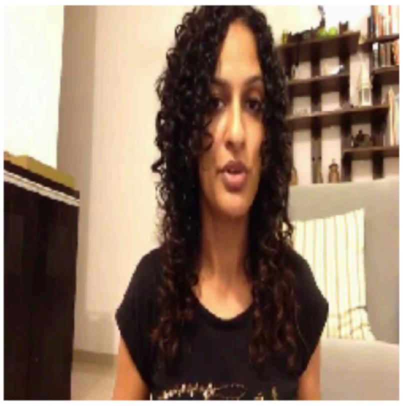
However, with the validation accuracy of 63.59% and a validation loss of 0.8043. The validation accuracy represents the model's performance on the validation dataset that it hasn't seen during training, which indicates the lacking ability of the model to classify unseen data. The validation loss represents how less accurate the model's classifications are.

While the validation accuracy is lower than the training accuracy, this is expected as the model may encounter new variations or complexities in the validation dataset that it didn't encounter during training.

V. Conclusion & Recommendations

With these final metrics results, the model shows its shortcomings when it comes to classifying the images in the validation data. These results indicate that overfitting is present in the model's performance as the difference of accuracy results between training and validation dataset is far.

One possible reason as to why overfitting is still present despite trying different methods to prevent it like regularization techniques is that there are images in the dataset not classified correctly.



Here is one image which is not classified correctly. This image is located inside the Wavy_Hair folder which means this image is categorized as wavy. However upon closer view, this image is more suited in the Curly_Hair folder. The researchers did not relocate this image as it is assumed that all data provided are correct.

Given that the dataset may not be entirely accurate, mitigating overfitting presents a significant challenge in model training. Overfitting occurs when a model learns to perform exceptionally well on the training data but fails to generalize effectively to unseen data. Inaccuracies or inconsistencies within the dataset can emphasize this issue, as the model may learn patterns or noise specific to the training data.

The researchers also recognized the importance of further standardizing the dataset to mitigate potential biases and improve the performance of the model.
One approach involves standardizing the color of hair across images to prevent the model from relying solely on hair color as a distinguishing feature, which could lead to biased classifications. By ensuring consistent hair color representation, the model can focus more on other distinguishing characteristics of hair types. Additionally, dataset cleaning procedures, such as isolating the hair region from the rest of the image, could be employed to remove outside factors that could introduce bias, such as ethnicity-related associations with hair texture. For instance, some ethnicities may have a higher prevalence of curly hair compared to others with predominantly straight hair. By isolating the hair region, the model can learn to classify hair types based on its intrinsic features rather than external factors like ethnicity. These

standardization and data cleaning techniques aimed to enhance the model's ability to generalize across diverse hair types and minimize the risk of biased classifications based on outside factors.

VI. References

Rosebrock, A. (2021, April 17). *Image Classification Basics*. PyImageSearch. https://pyimagesearch.com/2021/04/17/image-classification-basics/

Sharma, D. (2024, January 12). *Image classification using CNN | Step-wise tutorial*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/01/image-classification-using-convolutional-neural-networks-a-step-by-step-guide/#:~:text=Image%20classification%20using%20CNN%20involves,trainable%20parameters%20become%20extremely%20large.&text=We%20use%20filters%20when%20using%20CNNs.