PATTERN SENSE:Classification of poultry Diseases for Enhanced

* INTRODUCTION:
  + Project Title: transfer learning-based classification of poultry diseases for enhanced health management
  + Team Members:

1.Vinukonda karimulla( Gathering the complete project related Information)

2.Gorantla Dileep chowdary(Installation of the softwares related to front- End and Back-End )

3.Komera Himani( Importing the libraries and Building the Model)

4.Modepalli Jayanth(Running & Evaluating the Mode)

* PROJECT OVERVIEW:

**Skills Required:**

Python,Data Preprocessing Techniques,Tensorflow,HTML,CSS,JAVASCRIPT

**Project Description:**

**Scenario 1 - Description:**

*Poultry* farming is a vital sector in agriculture, but disease outbreaks can cause significant economic loss and threaten food supply chains. Traditionally, disease identification in birds relies on manual inspection, which is time-consuming and requires expert knowledge. Delays in diagnosis can lead to rapid spread and high mortality among poultry.

This project aims to solve this real-world problem by using Transfer Learning, a powerful technique in Deep Learning, to develop an intelligent system that can automatically detect and classify poultry diseases from images of infected birds. By leveraging pre-trained convolutional neural networks (CNNs), the system can achieve high accuracy even with limited data, enabling early detection and better disease management.

**✅ Scenario 2: Technical and Development-Focused Description**

**Scenario 2 - Description:**

This project focuses on implementing a deep learning-based poultry disease classification system using Transfer Learning. The system is built by fine-tuning a pre-trained CNN model (such as VGG16, ResNet50, or MobileNet) on a custom dataset of poultry images showing various diseases like Newcastle Disease, Fowl Pox, and Infectious Bronchitis.

The model is developed using Python, TensorFlow/Keras, and OpenCV for preprocessing and classification tasks. The dataset is augmented to improve model generalization, and performance is evaluated using accuracy, precision, recall, and confusion matrix metrics.

A Flask-based web interface is also provided where users can upload an image of a bird, and the system will predict the disease category. This smart solution can be used in farms or veterinary clinics to assist in early disease detection and improve overall poultry health management.

To accomplish this, we have to complete all the activities and tasks listed below

* Data Collection.
* Create a data frame with image paths, images, and labels
* Split into train\_df, test\_df, and valid\_df
* Data Pre-processing.
* Import the required library
* Apply data augmentation to balance train\_df
* Configure ImageDataGenerator class
* Apply ImageDataGenerator functionality to train\_df, valid\_df, and test\_df
* Model Building
* Pre-trained CNN model as a Feature Extractor
* Adding Dense Layer
* Configure the Learning Process
* Train the model
* Save the Model
* Test the model
* Application Building
* Create an HTML file
* Build Python Code

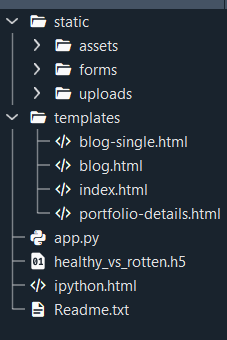
* PROJECT STRUCTURE:

* The Data folder contains the training and testing images for training our model.
* We are building a Flask Application that needs HTML pages stored in the templates.
* folder and a python scriptapp.pyfor server-side scripting
* we need the model that is saved and the saved model in this content is model\_cnn (2).h5
* templates folder contains home.html, predict.html & predictionpage.html pages.
  + ARCHITECTURE & SETUP INSTRUCTIONS:
* Data collection:

It is the most crucial aspect that makes algorithm training possible. There are many popular open sources for collecting the data.

Eg: kaggle.com, UCI repository, etc.

Create the Project folder which contains files as shown below



* We are building a Flask application with HTML pages stored in the templates folder and a Python script app.py for scripting.
* Healthy\_vs\_rotten.h5 is our saved model. Further, we will use this model for flask integration.

In this project, we have used ‘Common Fabric Pattern” data. This data is downloaded from kaggle.com

It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc. In this project, we have used ‘Common Fabric Pattern” data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

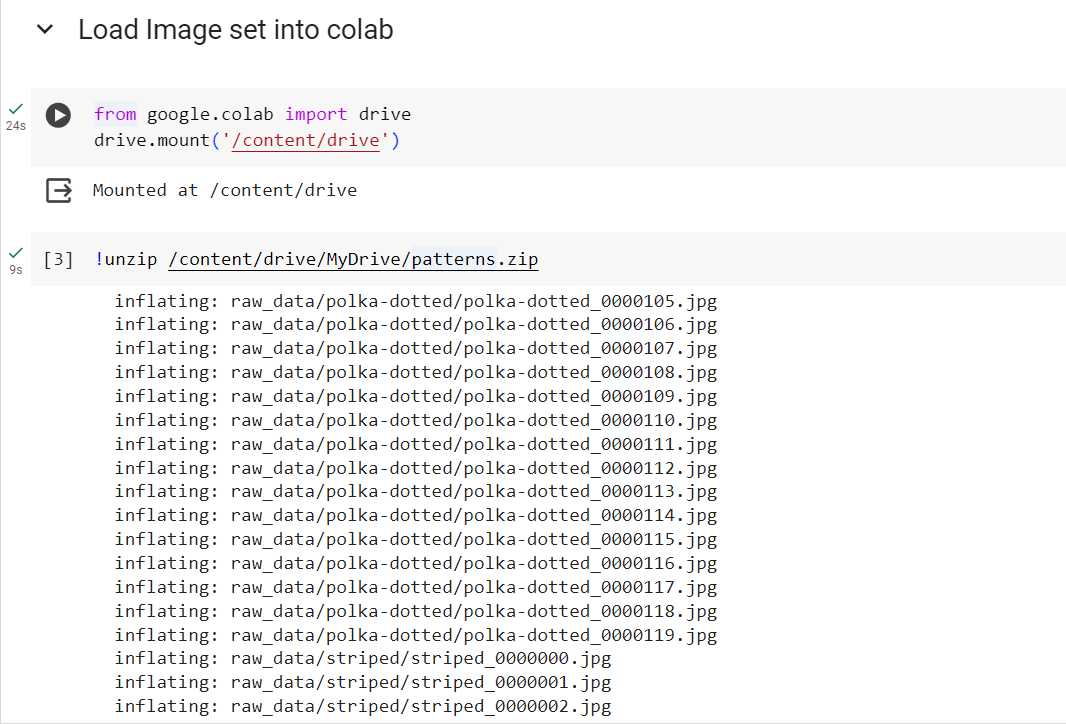
Link: <https://www.kaggle.com/datasets/nguyngiabol/dress-pattern-dataset>

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

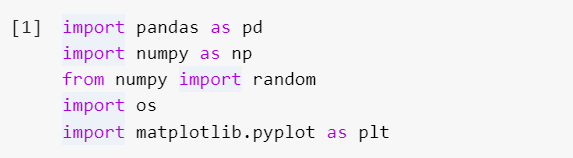
We are going to build our training model on Google Colab.

Upload the dataset into Google Drive and connect the Google Colab with drive using the below code



To build a DL model we have six classes in our dataset. But In the project dataset folder training and testing data are needed. So, in this case, we just have to assign a variable and pass the folder path to it.

Importing the libraries





The code performs data splitting using the train\_test\_split function from scikit-learn. Initially, it splits the original dataset df into training (train\_df) and test (dummy\_df) sets, allocating 80% of the data to training and 20% to testing, while ensuring class balance through stratification based on the 'label' column. Then, it further divides the test set dummy\_df into validation (valid\_df) and final test (test\_df) sets, each comprising 50% of the data. Finally, it prints the sizes of the three datasets and the value counts of the classes in the training set to verify the distribution.

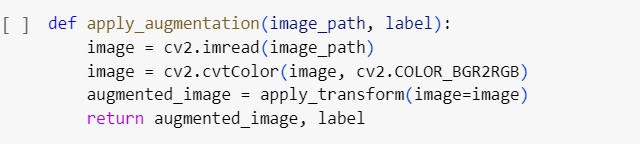
**Extracting labels from the files directory**

This code snippet retrieves a list of labels or classes from a directory. It uses the `os.listdir()` function to get the list of all items (files and directories) in the specified `directory`. The `labels` variable then stores this list, which typically represents the class names or categories used in a classification task. Each item in the `labels` list corresponds to a unique class or category in the dataset. We then sort the list in alphabetical order to ensure that our model gives the correct results.

Cv2 is a Computer Vision library which will be used for data image manipulations.

The provided Python function `apply\_transform(image)` performs image augmentation by applying random rotations (between -40 and 40 degrees), horizontal and vertical flips with a 50% probability, random adjustments to brightness and contrast, and random gamma correction. These transformations introduce variability to the input images, effectively increasing the diversity of the training dataset. This helps prevent overfitting and improves the model's ability to generalise unseen data by exposing it to a broader range of scenarios and variations.

Next we will create another augmentation function that calls the apply\_transform method.



The function `apply\_augmentation(image\_path, label)` reads an image from the specified

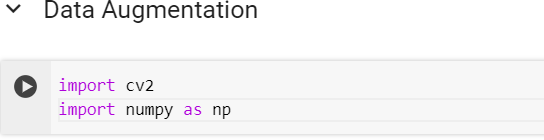
`image\_path` using OpenCV, converts it to RGB color space, and then applies augmentation transformations using the `apply\_transform()` function. Finally, it returns the augmented image along with the corresponding label. This function serves to augment a single image with transformations such as rotation, flipping, brightness and contrast adjustments, and gamma correction,thereby enhancing the diversity of the training dataset for improved model generalization and performance.

**Image Preprocessing**

In this milestone, we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although performing some geometric transformations of images like rotation, scaling, translation, etc.

**Importing the libraries**

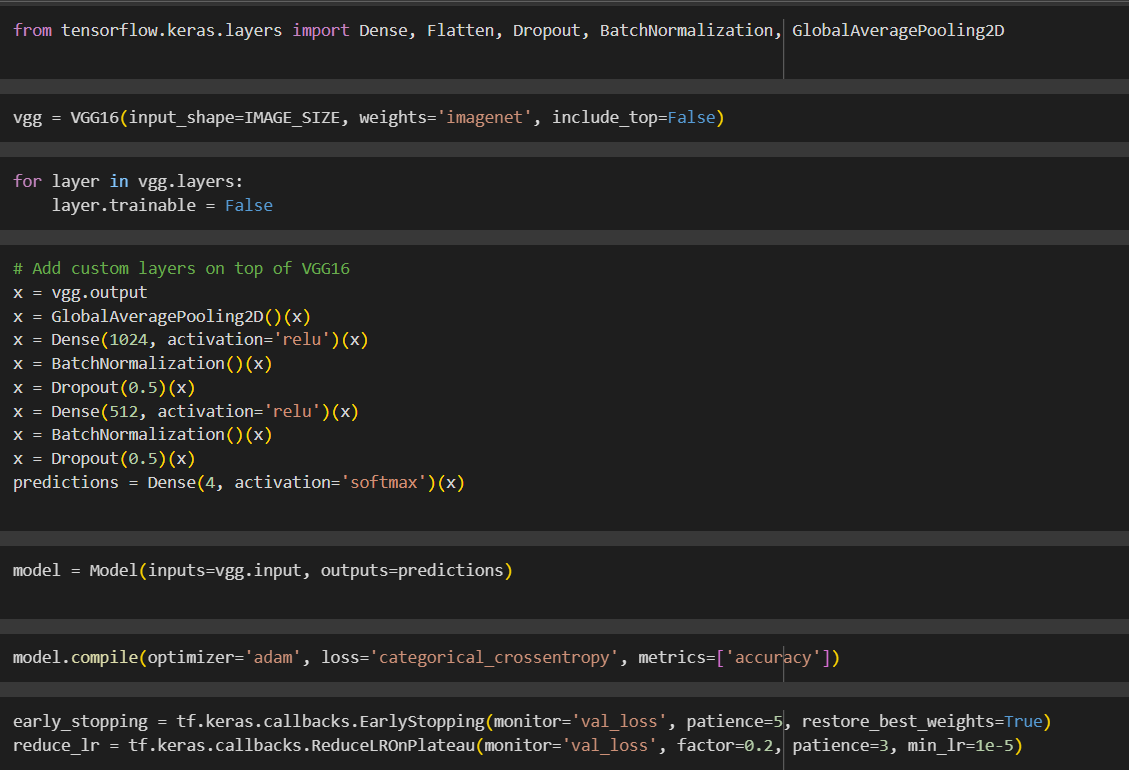
Import the necessary libraries as shown in the image



**Model Building:**

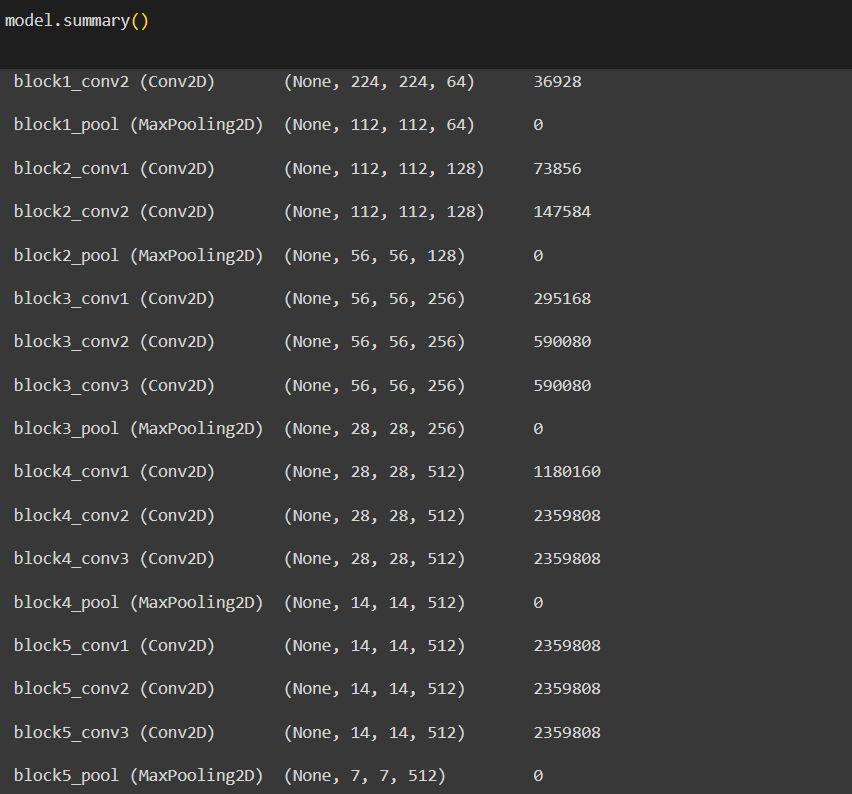
* Training the model in multiple algorithms  :

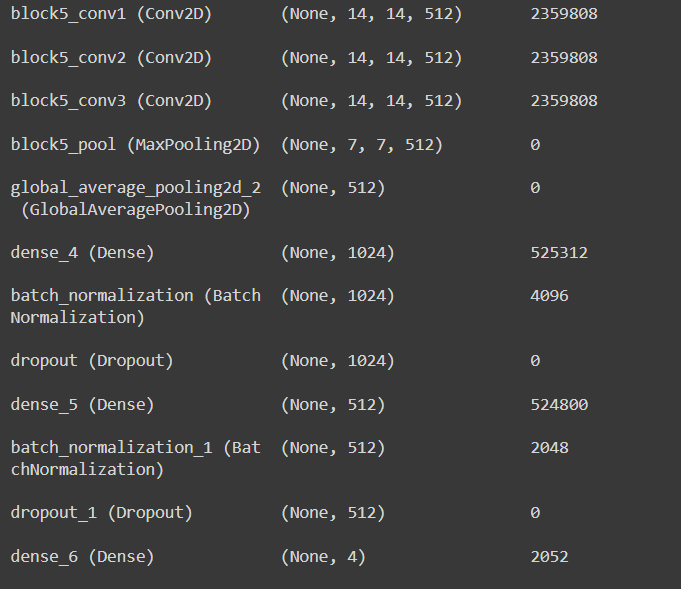
 Activity 2.1:VGG16

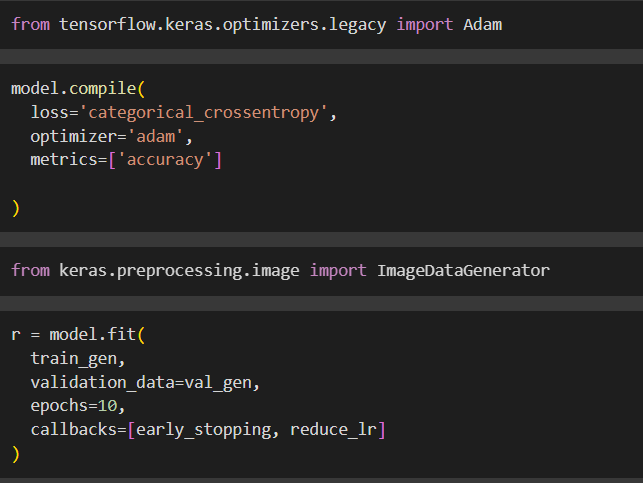


This short Python code snippet builds an image classifier with Keras. It cleverly reuses a pre-trained VGG16 model for its powerful image recognition abilities. Here's the key idea:

* The code loads the VGG16 model, but skips its final classification layers (keeping its feature extraction power).
* It then freezes the pre-trained part to focus training on new custom layers added on top.
* These custom layers likely handle the specific classification task you have in mind.
* Finally, it compiles the whole model for training, setting up how to improve and assess its performance.



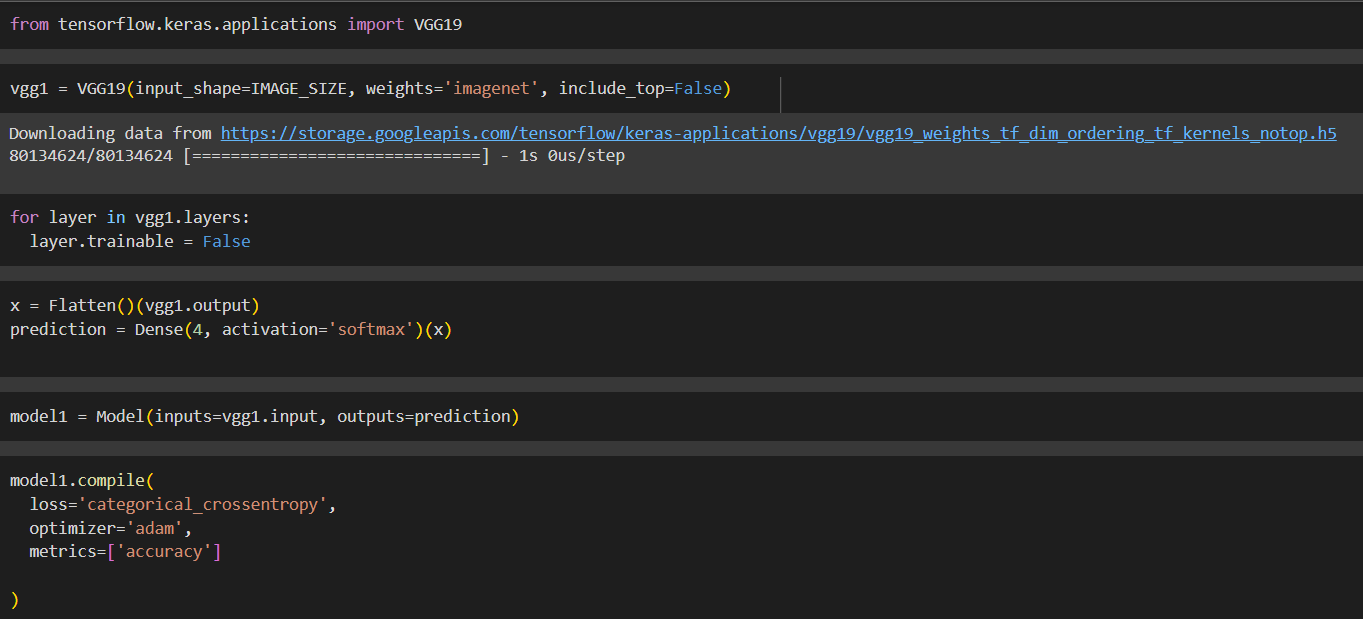




The text shows epochs, which are iterations over the training data. It also shows loss, which is how well the model is performing on the training data, and accuracy, which is how often the model makes correct predictions.

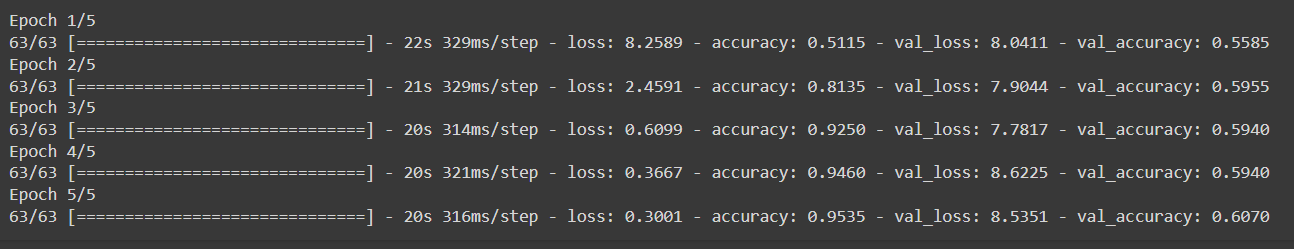
In the output you provided, it appears the model is improving over time as the loss is decreasing and the accuracy is increasing.

Activity 2.2:VGG19



* The first lines import libraries including TensorFlow and Keras.
* It appears to be defining a model with layers including Flatten and Dense which are commonly used in CNN architectures.
* The code then defines a process to compile the model, specifying an optimizer, loss function and metrics.

Overall, the code snippet seems to be training a CNN model on some data. However, without more context it’s difficult to say exactly what the model is being trained for.

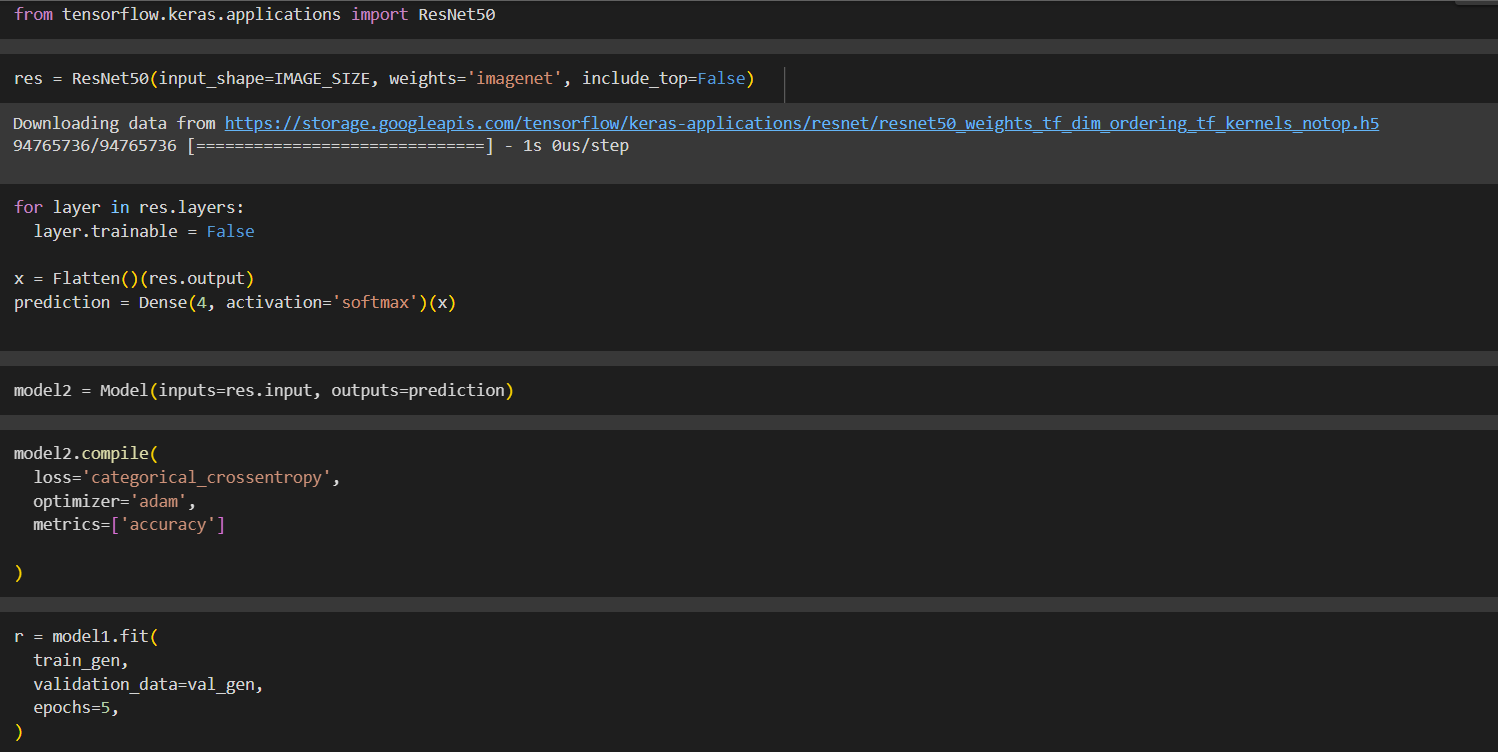


The image you sent shows the results of training a machine learning model over several epochs. Each epoch represents one pass through the training data.

* Loss: The training loss is decreasing over time, which indicates the model is learning to fit the training data better.
* Accuracy: The training accuracy is increasing over time, which indicates the model is making better predictions on the training data.

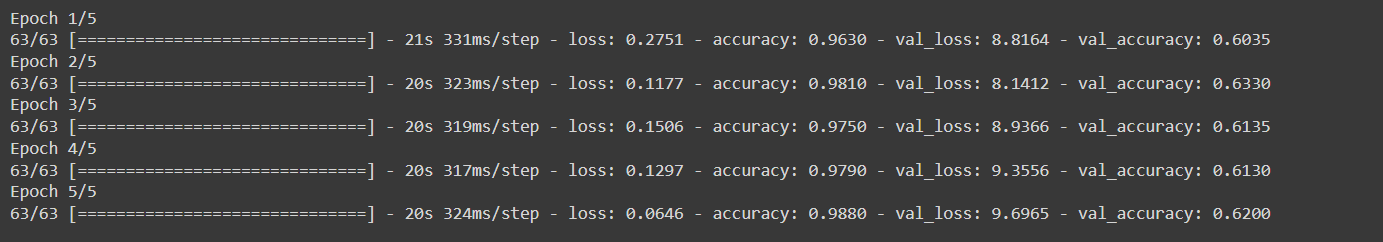
In machine learning, the goal is to train a model that generalizes well to unseen data. While the model's performance is improving on the training data, it is important to evaluate its performance on a separate validation set to assess itsgeneralizability.

Activity 2.3:ResNet50:



the code appears to be training a convolutional neural network (CNN) model for image classification using Keras. Here's a two-line explanation:

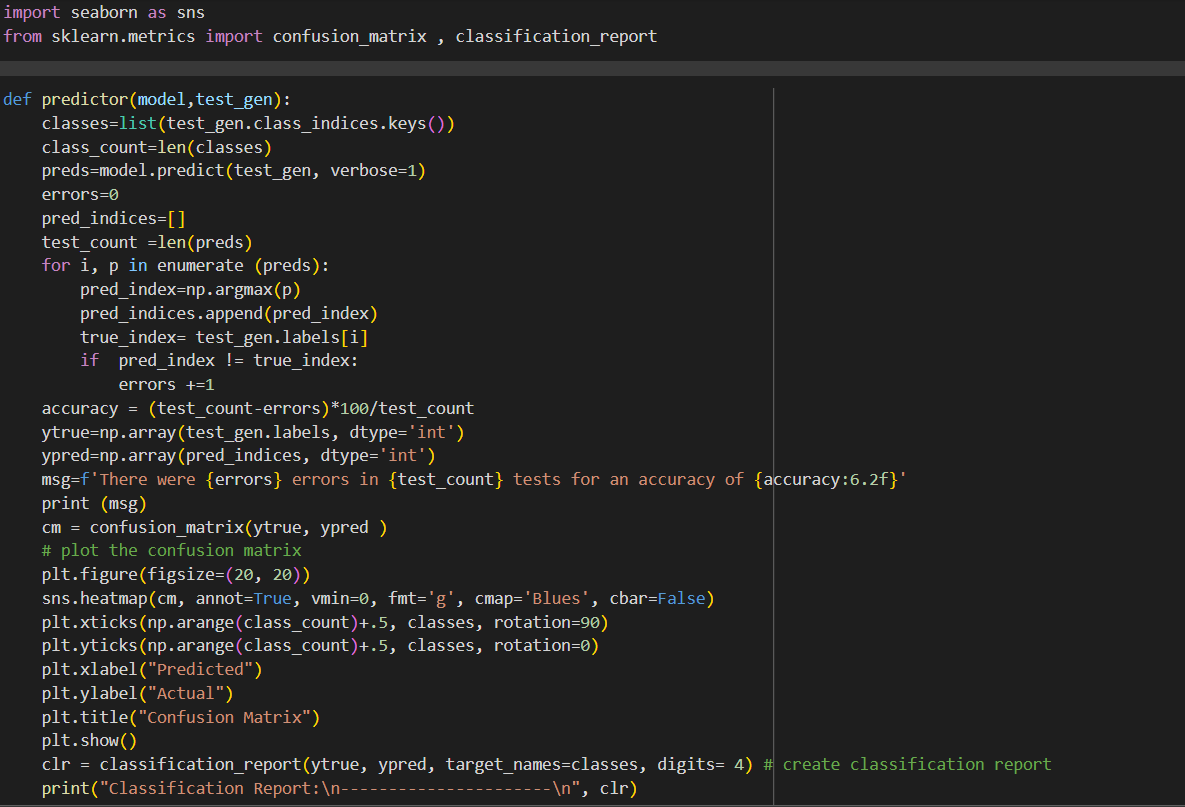
* The model is trained over multiple epochs (iterations) on training data.
* Loss (model's performance) decreases and accuracy (correct predictions) increases over epochs, suggesting the model is learning.



It displays results over five epochs, which are iterations over the training data.

* Loss: The model's training loss is decreasing (0.2751 to 0.0646) signifying the model is improving on the training data.
* Accuracy: Conversely, the training accuracy is increasing (0.9630 to 0.9880) indicating the model is making better predictions on the training data.

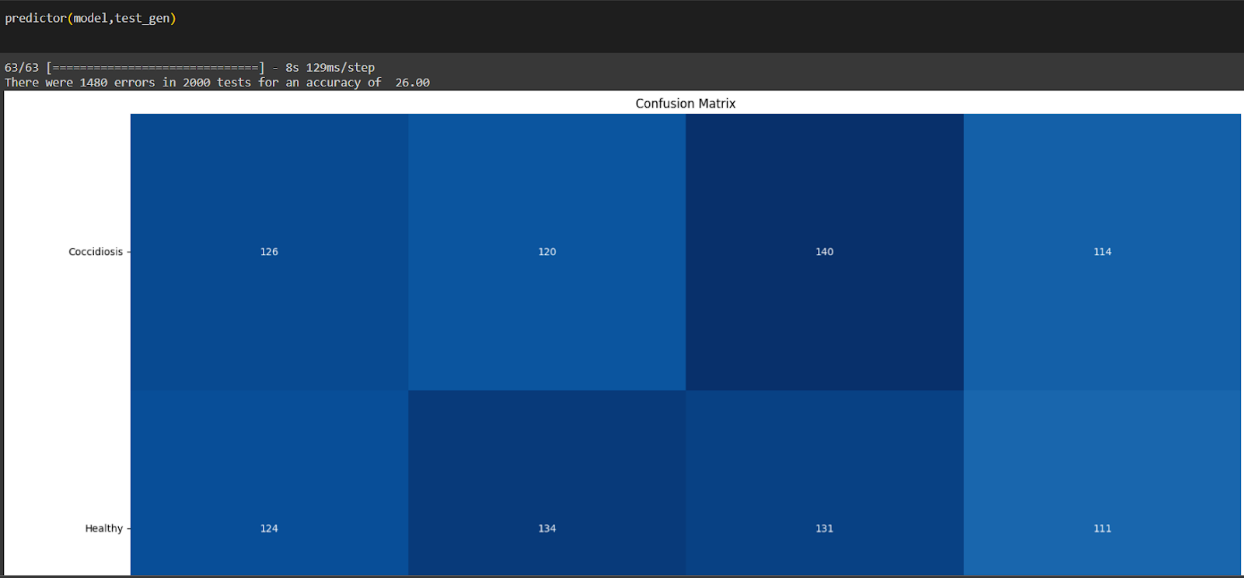
It's important to note that while this suggests the model is learning, its generalizability to unseen data needs to be assessed on a separate validation set.

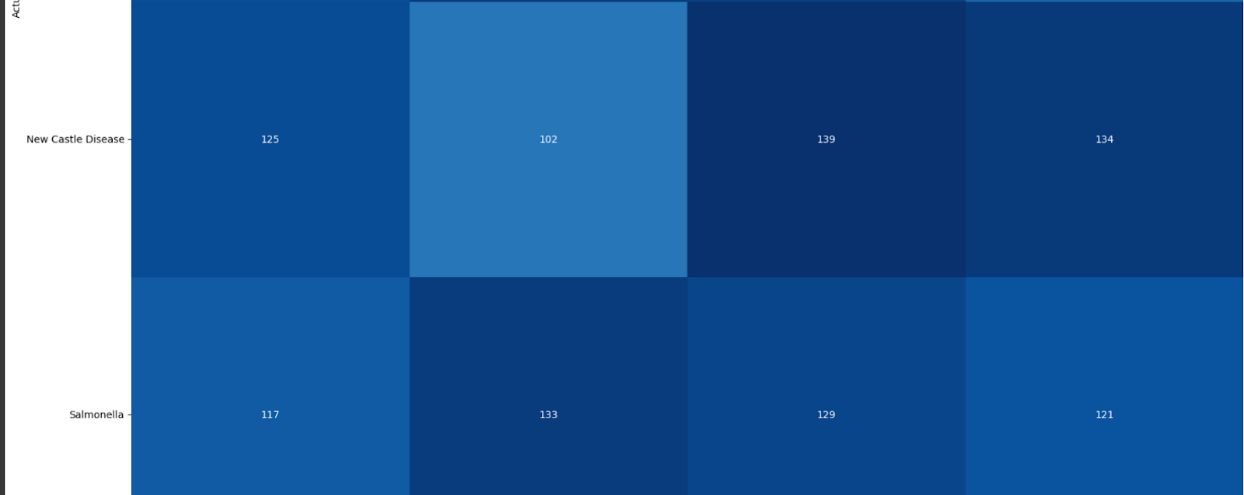


This Python code (Keras library) evaluates a pre-trained image classifier model. It likely:

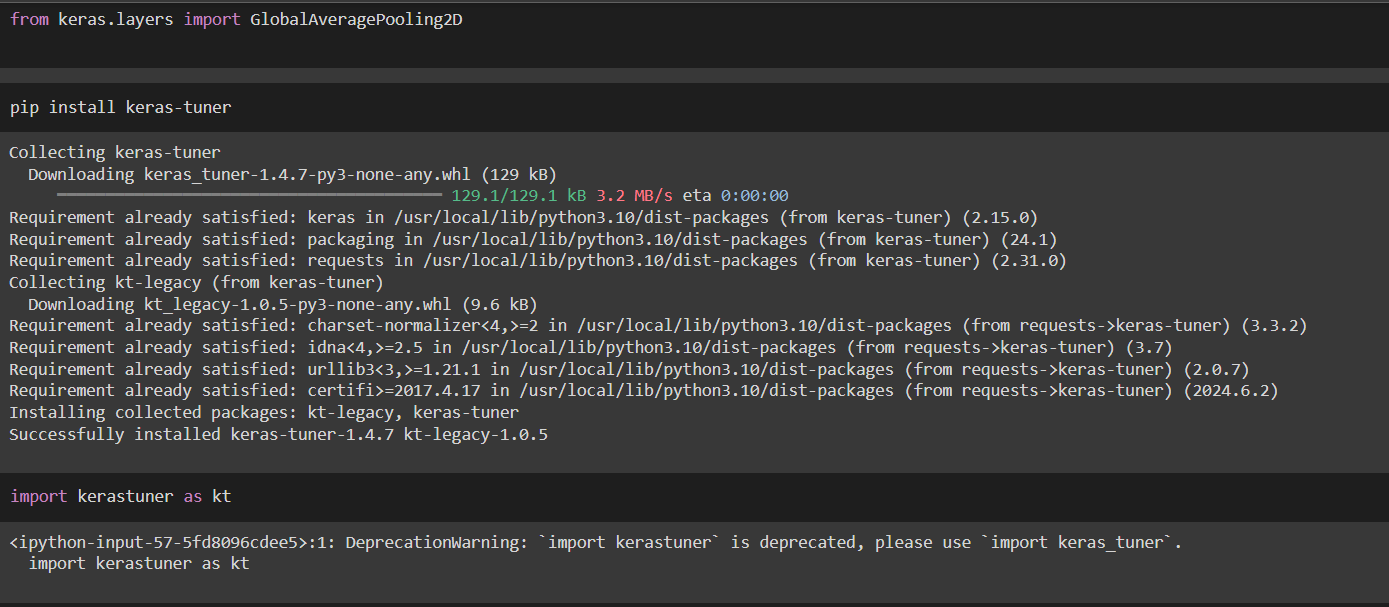
1. Imports libraries for machine learning and visualization.
2. Loads a pre-trained CNN model (e.g., VGG16) known for image recognition.
3. Prepares new image data for evaluation (resizing, formatting).
4. Feeds the data through the model and generates a confusion matrix.

The confusion matrix shows how well the model classifies the images (ideally high values on the diagonal).





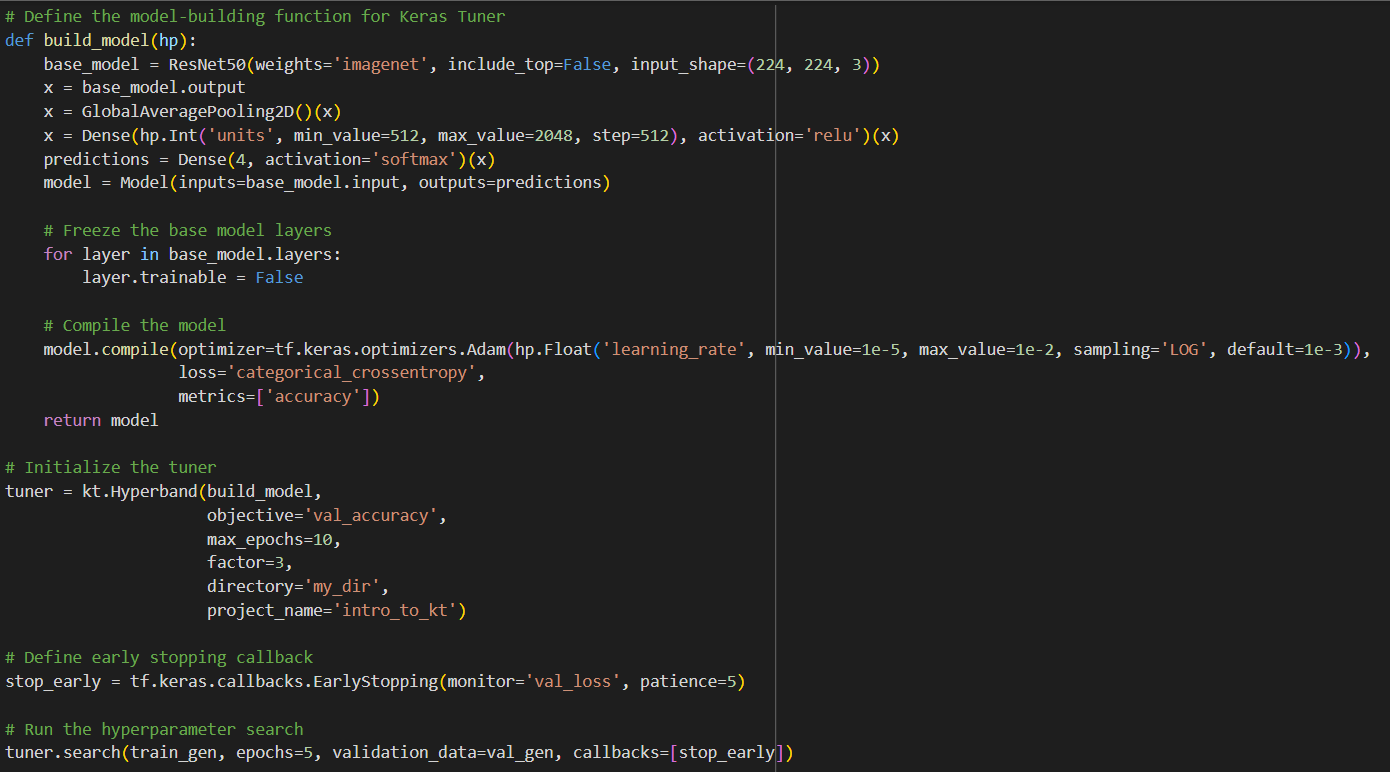


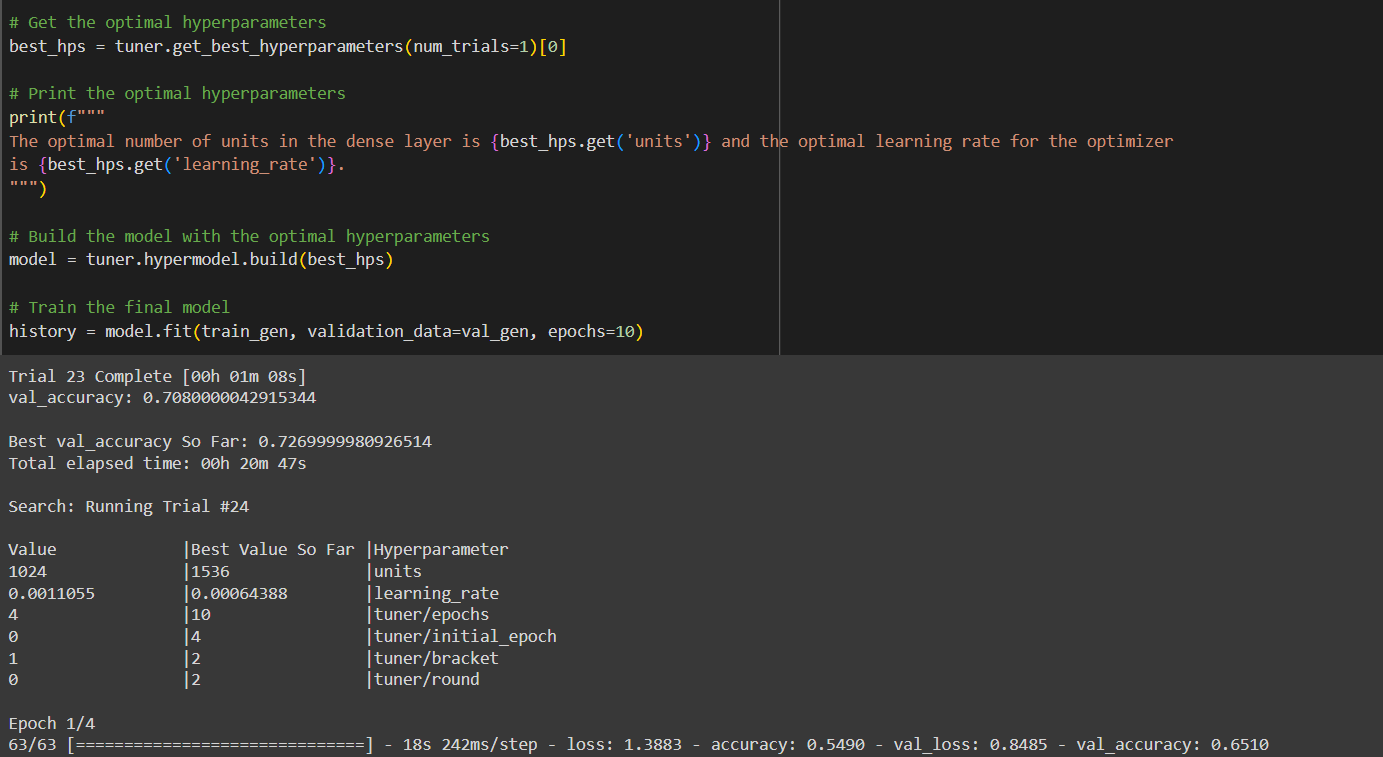


The code snippet appears to be setting up a convolutional neural network (CNN) for image classification using Keras. It likely involves:

1. Data Augmentation: Importing libraries (ImageDataGenerator) to perform transformations like rotation or flipping images. This helps the model learn from variations and generalize better.
2. Data Generators: Creating generators (train\_datagen and val\_datagen) to load and pre-process training and validation data efficiently during training.

Overall, this code prepares the data for training a CNN model on image classification tasks.





? The code defines a function named build\_model that creates a CNN model using Keras.

? It then uses this function along with Hyperband, a hyperparameter tuning technique, to find the optimal configuration (e.g., number of layers, learning rate) for the model that achieves the best validation accuracy.

? It finds the best hyperparameters (like learning rate) from past trials using tuner.get\_best\_hyperparameters().

? These optimal settings are used to build a new model with tuner.hypermodel.build.

? Finally, the model is trained on training data (train\_gen) while monitoring performance on validation data (val\_gen).

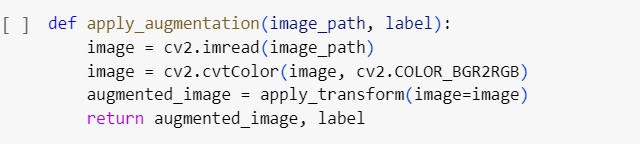
**Defining the augmentation function**

We will create a function that will take an image and augment it and return it back to the calling statement.



The provided Python function `apply\_transform(image)` performs image augmentation by applying random rotations (between -40 and 40 degrees), horizontal and vertical flips with a 50% probability, random adjustments to brightness and contrast, and random gamma correction. These transformations introduce variability to the input images, effectively increasing the diversity of the training dataset. This helps prevent overfitting and improves the model's ability to generalise unseen data by exposing it to a broader range of scenarios and variations.

Next we will create another augmentation function that calls the apply\_transform method.



The function `apply\_augmentation(image\_path, label)` reads an image from the specified `image\_path` using OpenCV, converts it to RGB color space, and then applies augmentation transformations using the

`apply\_transform()` function. Finally, it returns the augmented image along with the corresponding label. This function serves to augment a single image with transformations such as rotation, flipping, brightness and contrast adjustments, and gamma correction.

**Importing the libraries**

Import the necessary libraries as shown in the image



.

Keras.ImageDataGenerator is a powerful tool used for data augmentation in computer vision tasks, particularly image classification.



**Apply ImageDataGenerator functionality to train\_df, valid\_df, test\_df**



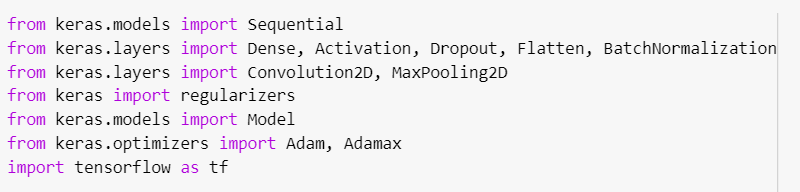
The provided code segment utilizes the flow\_from\_dataframe() method from a data augmentation generator (gen) to generate batches of augmented image data for training, validation, and testing purposes in a deep learning pipeline.

For the train\_gen, balanced\_df DataFrame is used as the source of training data. The 'path' column specifies the paths to the images, while the 'label' column indicates the corresponding labels. Images are resized to a target size of (255,255) pixels. The seed parameter ensures reproducibility by setting the random seed for shuffling. class\_mode is set to 'categorical' as the labels are one-hot encoded. color\_mode is specified as 'rgb' to indicate that images are in RGB color space. The shuffle parameter is set to True to shuffle the data after each epoch, enhancing randomness during training. Finally, batch\_size is set to 32, determining the number of samples per batch during training.

Similarly, for the val\_gen and test\_gen, the valid\_df and test\_df DataFrames are used as the sources of validation and testing data, respectively. Parameters such as shuffle and batch\_size are adjusted accordingly, with shuffle set to False for validation and testing data to ensure consistency during evaluation.

**Importing the libraries**

The following libraries will be required



The keras library is used to import all the layers that will be needed to create the CNN.

**Creating and Compiling the model**



* + By providing the all the textile related information in one-platform all the small textile Producers can be benefitted more.
  + Can be add tie-up with foreign traders for the export and import the textiles which

Is not only benefit to the textile industries but also increase our Indian economy growth.

\*THE END\*

vv