Algorithm - Data Extraction and merging

Input : Kaggle API key

Output: Dataset D

Start

1. Initialize Kaggle API:

- Set Kaggle API key.

- Authenticate Kaggle API.

2. Download HAM10000 Dataset:

- Use Kaggle API to download HAM10000 dataset.

- Specify the download path.

3. Combine Image Folders:

- Combine two image folders into one.

- Specify the paths for the two folders and the combined folder.

- Store this as Dataset D

End

return D

Pre process:

Algorithm - Data Augumentation

Input : Dataset D

Output: Augumented Dataset Da

Start

1. Set Output Directory:

- Specify an output directory path for storing augmented images.

2. Create Output Directory:

- Create the output directory if it does not exist.

3. Set Original Images Directory:

- Specify the path to the original images directory.

4. Set Classes to Augment and Target Sample Count:

- Specify the classes to augment.

- Set the target sample count for augmentation.

5. Initialize Augmented Entries List:

- Create an empty list to store augmented entries.

6. Iterate Through Metadata :

- For each row in the metadata:

- Check if the class is in the classes to augment.

- Read the original image.

- Normalize pixels if the image is not None.

imagenorm = image/255.0

- Set augment count.

- For each augmentation:

- Rotate and flip the image.

- Rotation

If (x,y) are the original pixel coordinates and (x’, y’) are the rotated coordinates by an

Angle θ

x’ = x.cos(θ)-y.sin(θ)

y’ = x.sin(θ)+y.cos(θ)

- Flip

Flipping involves change of sign of the coordinates

Flipx(x,y) = (-x,y)

Flipy(x,y) = (x,-y)

- Save the augmented image.

- Create a new image ID.

Image\_idnew = Image\_idold + 1

- Create an augmented entry.

7. Create Augmented Metadata DataFrame :

- Create a DataFrame from the augmented entries.

8. Merge Metadata:

- Merge the original metadata with the augmented metadata into one DataFrame.

9. Store All Images:

- Store all the images, including augmented ones, in the specified output directory.

- Save it as Dataset Da

End

Return Da

Algorithm - Normalize and Resize Images

Input:

- List of image paths (image\_paths)

- Dataset Da

- Target size for resizing (target\_size)

Output:

- List of normalized and resized images (normalized\_resized\_images)

Start

1. Initialize an empty list to store normalized and resized images:

Normalized\_resized\_images = []

2. For each image path in the list image\_paths:

a. Read the image from the file path.

b. Convert the image to a format suitable for processing (e.g., RGB).

c. Resize the image to the target size:

resized\_image = cv2.resize(normalized\_image, target\_size)

d. Convert them into grayscale images:

*I*=0.299×*R*+0.587×*G*+0.114×*B*

e. Convert them into numpy arrays:  
 If I is the intensity of the pixels at (x,y) then

Img\_array[x,y] = I

d. Normalize the pixel values of the image:

- If the pixel values are in the range [0, 255], apply normalization:

normalized\_image = image/255.0

e. Append the normalized and resized image to the list normalized\_resized\_images.

End

Return normalized\_resized\_images

Disease Prediction:

Algortihm -Model Comparison:

Input:

- Dataset (Da) with labeled samples for training and evaluation.

- Models: ResNet-152, MobileNet, InceptionV3.

Output:

- Best performing model(modelbest)

Begin

1. Data Preprocessing:

- Split the dataset into training and evaluation sets.

2. Model Training:

- Train each model (ResNet-152, MobileNet, InceptionV3) on the training set using appropriate hyperparameters.

- Utilize a common evaluation metric (e.g., accuracy, F1-score) to assess the models' performance during training.Utilizing accuracy

-Utilizing F1 Score

3. Model Evaluation:

- Evaluate the trained models on the evaluation set using the chosen metrics.

- Calculate the performance metric for each model.

4. Model Selection:

- Identify the model with the highest performance based on the evaluation metric.

- If multiple metrics are considered, use a weighted combination or prioritize the most critical metric based on the application.

5. Result Output:

- Output the selected model as the best-performing model(modelbest) for the given dataset and evaluation metric.

return modelbest

Algorithm - Add Explainability module(CAM) to Best Model:

Input:

- Best performing model (modelbest)

- Image for CAM visualization

Output:

- CAM visualization overlaid on the original image

Begin:

1. Load Best Model:

- Load the saved best-performing model(modelbest).

2. Load Image for CAM Visualization:

- Load the image on which CAM will be visualized.

3. Preprocess Image:

- Preprocess the image to match the input requirements of (modelbest).

4. Get Relevant Layers:

- Identify the last convolutional layer and classifier layer indices in (modelbest).

5. Calculate Gradients:

- Using TensorFlow GradientTape, calculate gradients of the predicted class with respect to the output feature map.

6. Obtain CAM:

- Perform global average pooling to obtain the CAM.

7. Normalize CAM:

- Normalize the CAM to values between 0 and 1.

8. Heatmap Visualization:

- Create a heatmap using the CAM values.

9. Overlay CAM on Original Image:

- Superimpose the heatmap on the original image to create the CAM visualization.

10. Display Result:

- Display the original image and the CAM visualization.

End

Return visualization

Algorithm - Disease Prediction

Input

- image of the skin lesion

- Deployed model(modeldeployed)

Output

- Prediction of the disease along with the visualization of where it is

1. Load Deployed Model:

- Load the deployed mode(modeldeployed)l.

2. Load Image for Prediction and CAM Visualization:

- Load the image for which predictions and CAM will be visualized.

3. Preprocess Image:

- Preprocess the image to match the input requirements of

img\_array = Preprocess(img)

4. Make Prediction:

- Obtain model predictions for the input image:

predictions = modeldeployed(img\_array)

1. Obtain CAM:

- Obtain CAM visualization

6. Display Result:

- Display the original image, predictions, and the CAM visualization.

End

Return results