**Implementation of scalp psoriasis steps:**

The project involves the early detection of scalp psoriasis using deep learning, particularly convolutional neural networks (CNNs). Initially, a CSV file containing image paths and labels is preprocessed to ensure paths are valid. The dataset is then split into training and validation sets, and image data generators are employed for data augmentation and normalization.

A CNN model is built and compiled with layers for convolution, pooling, flattening, and dense connections, optimized using the Adam optimizer and binary cross-entropy loss. The model is trained on the training set and validated using the validation set. After training, the model's performance is evaluated by plotting training and validation accuracy and loss over epochs.

A confusion matrix is generated from predictions on the validation set to visualize performance, and a heatmap is plotted to illustrate the distribution of true and predicted labels. This helps in understanding model accuracy and areas needing improvement. The entire process aims to classify images into healthy or unhealthy categories effectively.

**Convolutional Neural Network (CNN)**

**Overview**

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing data that has a grid-like topology, such as images. They are particularly effective for image recognition and classification tasks due to their ability to capture spatial hierarchies in data through convolutional layers.

**Key Components**

1. **Convolutional Layers**: These layers apply convolutional operations to the input, using a set of learnable filters to detect features such as edges, textures, and patterns. Each filter slides over the input image and produces a feature map, highlighting the presence of specific features at various locations.
2. **Activation Functions**: Typically, a ReLU (Rectified Linear Unit) activation function is applied after each convolutional layer to introduce non-linearity, allowing the network to learn more complex patterns.
3. **Pooling Layers**: These layers perform downsampling operations (e.g., max pooling), reducing the spatial dimensions of the feature maps. This helps in reducing computational complexity and controlling overfitting by summarizing the presence of features in a region.
4. **Fully Connected Layers**: After several convolutional and pooling layers, the high-level reasoning in the neural network is done via fully connected layers. These layers are similar to those in traditional neural networks and help in combining features learned by convolutional layers to make predictions.
5. **Dropout**: This technique is used to prevent overfitting by randomly setting a fraction of input units to zero at each update during training. It helps in making the model more robust and generalizable.
6. **Output Layer**: For binary classification, the output layer typically consists of a single neuron with a sigmoid activation function, producing a probability score between 0 and 1.

**Training Process**

1. **Data Preprocessing**: Images are normalized and augmented using techniques like rescaling, rotation, and flipping to enhance the diversity of the training set and improve model robustness.
2. **Compilation**: The model is compiled with an optimizer (e.g., Adam) and a loss function (e.g., binary cross-entropy) to guide the learning process.
3. **Training**: The model is trained on the training dataset, and its performance is monitored on a validation set. The training process involves forward propagation, backpropagation, and parameter updates to minimize the loss function.
4. **Evaluation**: After training, the model's performance is evaluated on unseen data to measure its accuracy and generalization capability. A confusion matrix and heatmap can be used to visualize the model's classification performance.

**Step by step implementation:**

 **Load and Preprocess Data**:

* Load the CSV file containing image paths and labels.
* Assign labels to images (1 for healthy, 0 for unhealthy).
* Correct image paths by replacing backslashes with forward slashes.

 **Split Dataset**:

* Split the dataset into training and validation sets using train\_test\_split.

 **Data Augmentation**:

* Use ImageDataGenerator to normalize and augment the image data for both training and validation sets.

 **Build CNN Model**:

* Construct a CNN model with layers: Conv2D, MaxPooling2D, Flatten, Dense, and Dropout.
* Use ReLU activations for hidden layers and sigmoid activation for the output layer.

 **Compile Model**:

* Compile the CNN model using the Adam optimizer and binary cross-entropy loss.
* Specify accuracy as a metric to monitor during training.

 **Train Model**:

* Train the model using the training data generator and validate it using the validation data generator.
* Monitor training and validation accuracy and loss over epochs.

 **Evaluate Model**:

* Evaluate the model's performance on the validation set using model.evaluate.
* Print validation accuracy.

 **Confusion Matrix and Heatmap**:

* Generate predictions on the validation set.
* Create a confusion matrix using confusion\_matrix.
* Plot a heatmap of the confusion matrix using Seaborn to visualize classification performance.

 **Classify New Images**:

* Define a function to classify new images using the trained model.
* Load and preprocess new images, make predictions, and print the classification results (healthy or unhealthy).

 **Visualize Training Performance**:

* Plot training and validation accuracy and loss curves to analyze model performance over epochs.

**Why we used cnn**

We used a Convolutional Neural Network (CNN) in this project because CNNs are highly effective for image data processing and classification tasks. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input images. They consist of convolutional layers that apply filters to the input images, capturing important features such as edges, textures, and shapes, which are crucial for distinguishing between healthy and unhealthy scalp images.

CNNs also benefit from locality and translation invariance, meaning they can detect features regardless of their location in the image, making the model robust to variations in input. The use of shared weights in convolutional layers significantly reduces the number of parameters, leading to efficient training and lower computational complexity compared to fully connected networks.

Additionally, CNNs can be built with deep architectures, allowing them to learn complex and abstract representations of the input data. This deep learning capability is essential for accurately classifying medical images, where subtle differences between healthy and unhealthy conditions need to be identified. The proven performance of CNNs in numerous image classification tasks makes them a reliable choice for the early detection of scalp psoriasis.

How we used cnn:

In this project, we used a Convolutional Neural Network (CNN) to classify images of scalps as either healthy or unhealthy. We built a CNN model with multiple layers, including convolutional layers to automatically extract features, pooling layers to reduce dimensionality, and fully connected layers for classification. We trained the model using labeled scalp images, where the first 474 images were labeled as healthy and the remaining 291 as unhealthy. The model was trained to minimize binary cross-entropy loss using the Adam optimizer. After training, we evaluated the model's performance using validation data, generating a confusion matrix and a heatmap to visualize classification accuracy. Finally, the trained model was used to classify new images.

Operation performed on images and why:

In this project, several operations were performed on images to enhance model performance and reliability:

1. **Rescaling**: Images were rescaled to normalize pixel values to a range of 0 to 1, aiding in faster convergence during training.
2. **Data Augmentation**: Techniques like rotation, flipping, and zooming were applied to artificially increase the dataset size, helping the model generalize better.
3. **Convolution**: Convolutional layers were used to extract meaningful features such as edges and textures from the images.
4. **Pooling**: Pooling layers reduced the dimensionality of feature maps, retaining essential information while minimizing computational load.
5. **Flattening**: The feature maps were flattened to convert them into a one-dimensional array, suitable for the fully connected layers.
6. **Fully Connected Layers**: These layers were used to interpret the extracted features and make final classification decisions.
7. **Activation Functions**: ReLU was used for non-linearity, and sigmoid for the output to produce probabilities for binary classification.

These operations were crucial for effectively training the CNN to distinguish between healthy and unhealthy scalp images.

**Information about model we have used**

This code defines a Convolutional Neural Network (CNN) model using TensorFlow and Keras. Here’s a short and meaningful explanation of how it performs operations on images and why this model is used:

1. **Conv2D Layers**: These layers apply 32, 64, and 128 filters (3x3 in size) to the input image, detecting features like edges and textures. ReLU activation introduces non-linearity, enabling the model to learn complex patterns.
2. **MaxPooling2D Layers**: These layers downsample the feature maps by taking the maximum value in each 2x2 window, reducing the spatial dimensions and computational load while retaining essential features.
3. **Flatten Layer**: Converts the 2D feature maps into a 1D array, preparing them for the fully connected layers.
4. **Dense Layers**: The first dense layer with 512 neurons and ReLU activation learns high-level features, while the final dense layer with a single neuron and sigmoid activation outputs a probability for binary classification (healthy or unhealthy).
5. **Dropout Layer**: This layer randomly sets 50% of its inputs to zero during training, preventing overfitting and improving model generalization.

This model is used because it efficiently extracts hierarchical features from images, reducing dimensionality and computational complexity, making it suitable for image classification tasks like distinguishing between healthy and unhealthy scalp images.

**Conclusion:**

This project aims to develop a Convolutional Neural Network (CNN) to classify scalp images as healthy or unhealthy for early detection of scalp psoriasis. The dataset includes 765 images with labels indicating health status. Data preprocessing involved normalizing image paths and labels, followed by splitting the data into training and validation sets. ImageDataGenerator was used for data augmentation and normalization. A CNN model was constructed with convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification. The model was compiled with the Adam optimizer and binary cross-entropy loss. Training involved monitoring accuracy and loss on the validation set. Performance evaluation included generating a confusion matrix and heatmap to visualize classification results. The final model can classify new scalp images, helping in the early detection of psoriasis.