How CNN Works for Disease Prediction Using Rice Leaf Images

1. Input Data (Training Dataset):

- **Leaf Images**: The first step is to gather a large number of labeled leaf images, where each image corresponds to a specific disease (we currently considering "Bright," "Blast," "Brown Spot").
- **Preprocessing**: Before feeding these images into the model, preprocessing steps like resizing, normalizing, and sometimes augmenting the images are performed to make sure the model receives the data in a suitable format for training.
    - **Resizing**: All images are resized to the same dimensions (e.g., 224x224 pixels or 512*512 pixel).
    - **Normalization**: Pixel values of images are usually scaled to a range, typically between 0 and 1, to improve model convergence.(conversion of grey scale)
    - **2. Feature Extraction:**

Feature extraction is a key part of any machine learning model. The goal here is to identify features that can help distinguish between the different classes (e.g., disease types). In image classification:

- **Color**: color can be very between different disease.
- **Texture**: diseases might cause certain texture patterns on the leaf surface.
- **Shape**: The shape of spots or lesions (e.g., round, irregular) can be a distinguishing feature.
- **Size of the Spots**: This could be another feature where the model learns the typical size of the spots for each disease type.

3. Model Architecture (Convolutional Neural Networks - CNNs):

CNNs are specifically designed to handle image data. Inside a CNN, there are several key layers that work together to learn from the data:

- **Convolutional Layer**:
    - This is the core building block of a CNN. It applies convolution operations to the image to extract features.
    - A filter (also known as a kernel) slides over the image and calculates the dot product between the filter and the local region of the image, creating feature maps that highlight important details (edges, textures, etc.).
    - Multiple filters can be used at each layer to detect different features.
- **Activation Function (ReLU)**:

    - After convolution, the ReLU (Rectified Linear Unit) activation function is applied. This introduces non-linearity into the model, allowing it to learn more complex patterns.
    - It replaces all negative pixel values with zero, which helps introduce sparsity (reducing the amount of unnecessary information).

- **Pooling Layer**:
    - Pooling is used to reduce the spatial dimensions of the image (width and height), which helps reduce computational cost.
    - **Max pooling** is often used, where the maximum value from a small region is taken to summarize that part of the image.
    - This allows the model to focus on the most important information in the image.

- **Fully Connected Layer**:
    - After the convolutional and pooling layers, the image data is flattened into a one-dimensional vector and passed to fully connected layers (also called dense layers).
    - These layers learn to combine the features extracted earlier to make predictions.
    - The output of this layer will have a probability distribution over the different classes (diseases).

- **Softmax Layer**:
    - The final layer in a classification task is typically the **softmax layer**. This converts the output into probabilities for each class, allowing the model to decide which disease is most likely.
    - For example, if the model predicts a leaf disease, it might give the following probabilities: 0.8 for "Bright," 0.1 for "Blatt," and 0.1 for "Brown Spot."
    - The class with the highest probability is selected as the final prediction.

4. Training the Model:

The training process involves showing the model labeled images (i.e., images with known disease types) and adjusting the model's weights so that its predictions become more accurate over time. This is done by:

- **Loss Function**: The loss function quantifies how far the model's predictions are from the actual labels (ground truth).
    - Example: If the model predicts 80% for "Bright" but the actual label is "Blast," the loss function will compute a high error.

- **Optimization Algorithm**: To minimize the loss function, an optimization algorithm like **Stochastic Gradient Descent (SGD)** or **Adam** is used. It adjusts the model's weights iteratively by calculating the gradients .

Training Steps:

- Forward pass: The image is passed through the network, and the output is computed.
- Loss calculation: The error (difference between predicted and actual labels) is computed.
- Backpropagation: The gradients are calculated by backpropagating the error through the network.
- Update weights: The optimizer adjusts the weights using the gradients to reduce the error in future predictions.

5. Model Testing:

This helps assess how well the model generalizes to unseen data.

- **Accuracy**: The most common metric for classification tasks. It is the ratio of correctly predicted instances to total instances.
- **Confusion Matrix**: It helps to understand how well the model is distinguishing between classes by showing the true positives, false positives, true negatives, and false negatives.

6.Deployment & Real-Time Inference:

- Once the model is trained and tested, it can be deployed on a microcontroller, such as a Raspberry Pi Pico, to operate autonomously in real-time applications. A user can capture a photo of a plant leaf using a camera system connected to the microcontroller. The photo is then sent to a mobile application, where the model performs inference (prediction) to identify the disease. Based on the prediction, the system provides actionable solutions, such as recommending the application of fungicides or adjustments in fertilization. Since the model has been trained to recognize patterns of various diseases, it can make fast, automated decisions based on new leaf images.