AI in Potato Agriculture

Step 1: Prototype Selection

Problem Statement

Traditional methods of potato farming involved manual disease detection and remedies, which were labor-intensive and often unreliable. Advancements in technology now allow for more accurate disease detection by analyzing potato leaves. This method uses deep learning techniques like Convolutional Neural Networks and machine learning techniques like K-means algorithms. These models are trained on extensive datasets of healthy and diseased potato leaves, and then deployed into a mobile or Web application. This app allows farmers to take a picture of a potato leaf, detecting any signs of disease, enabling timely interventions and supporting economic stability in farming communities.

Market/Customer/Business Need Assessment

Market/Customer/Business Need Assessment for Potato Disease Detection

Market Need:

• High disease prevalence of potato crops.

• Large scale of production in India.

• Inefficient traditional methods for disease detection.

Customer Need:

• Farmers need simple, accurate, and efficient disease monitoring tools.

• Agricultural consultants benefit from advanced tools for disease diagnosis and treatment recommendations.

• Agribusiness companies need to ensure crop quality and yield for profitability.

Business Need:

• Improved crop yields.

• Technological advancements in agriculture.

• Market expansion through mobile or web application development.

Target Specifications

Mobile or Web Application for Detecting Potato Diseases

• High Accuracy: Identifies common potato diseases with over 90% accuracy.

• Rapid Results: Provides real-time analysis within seconds of image capture.

• Ease of Use: Features an intuitive, user-friendly interface accessible to all technological skill levels.

• Scalability: Supports a large number of users and handles substantial volumes of image data efficiently.

• Offline Functionality: Usable without an internet connection for remote farmers.

• Affordability: Cost-effective solution with minimal usage and maintenance costs.

• Cross-Platform Compatibility: Works on various smartphone operating systems, including Android and iOS.

External Search

→ Dataset

→ CNN

Let's import the dataset and have a look at it!

Dependencies

In [1]:
```
import os
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.metrics import confusion_matrix, classification_report

import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array

print(tf.__version__)
############ settings ############
data_dir = r"..\data"

batch_size = 32
img_height = 128
img_width = 128
```

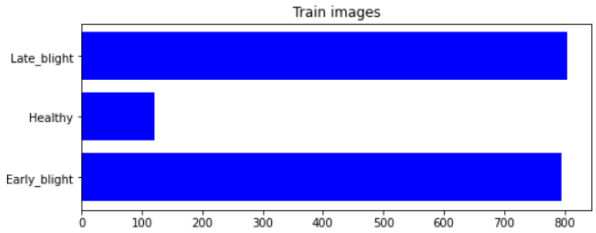
```
2.6.0
```

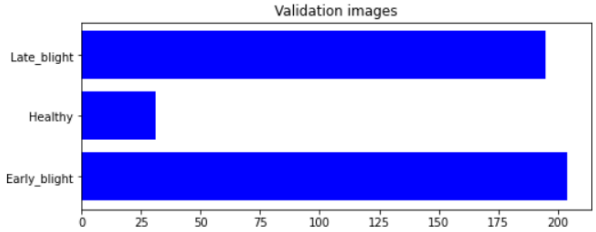
Data preprocessing

- Loading and splitting data
- Train:80% and Validation:20%

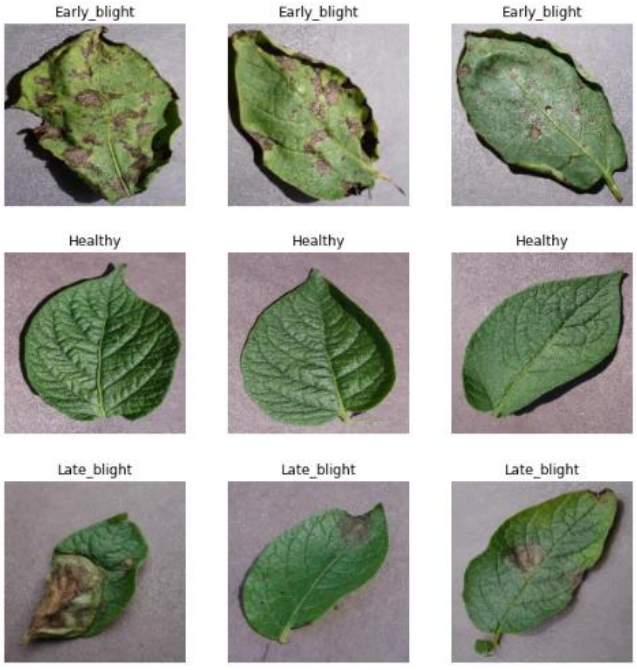
In [2]:
```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size)
```

```
Found 2152 files belonging to 3 classes.
Using 1722 files for training.
Found 2152 files belonging to 3 classes.
Using 430 files for validation.
```





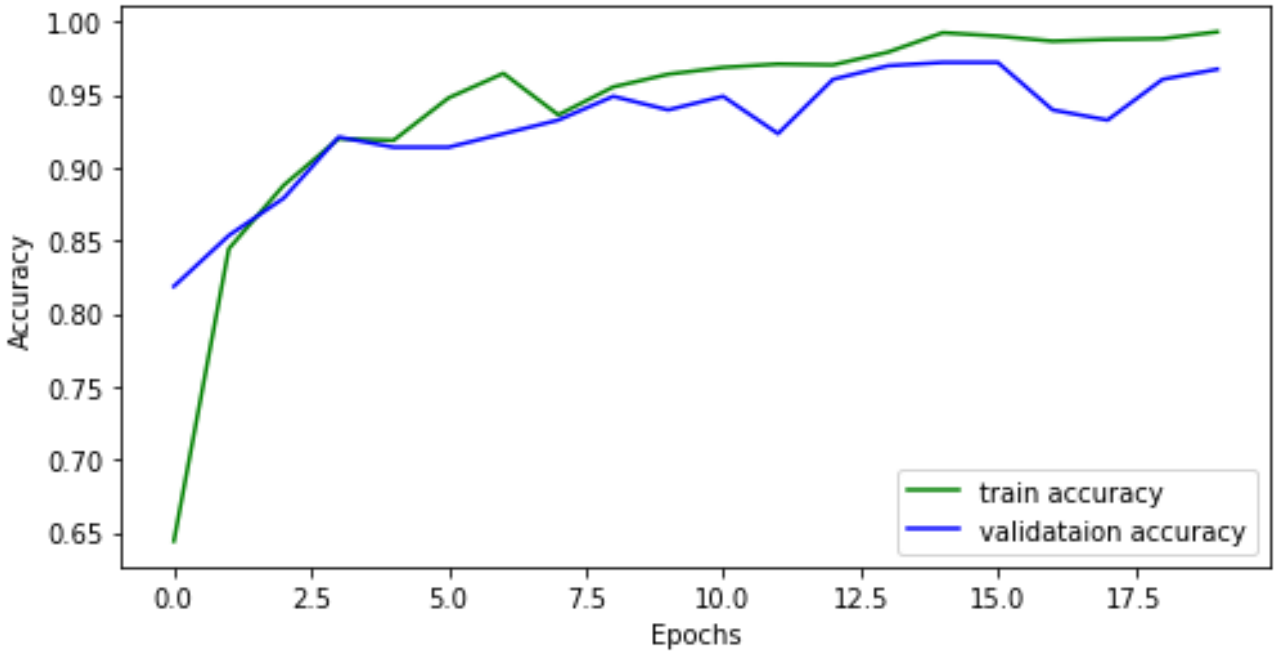
In [5]:
```
## Sample images
class_names = train_ds.class_names

for class_name in class_names:
    imgs = os.listdir(os.path.join(data_dir, class_name))[:3]
    plt.figure(figsize=(10, 10))
    for i, img in enumerate(imgs):
        ax = plt.subplot(3, 3, i+1)
        plt.imshow(plt.imread(os.path.join(data_dir, class_name, img)))
        plt.title(class_name)
        plt.axis('off')
```

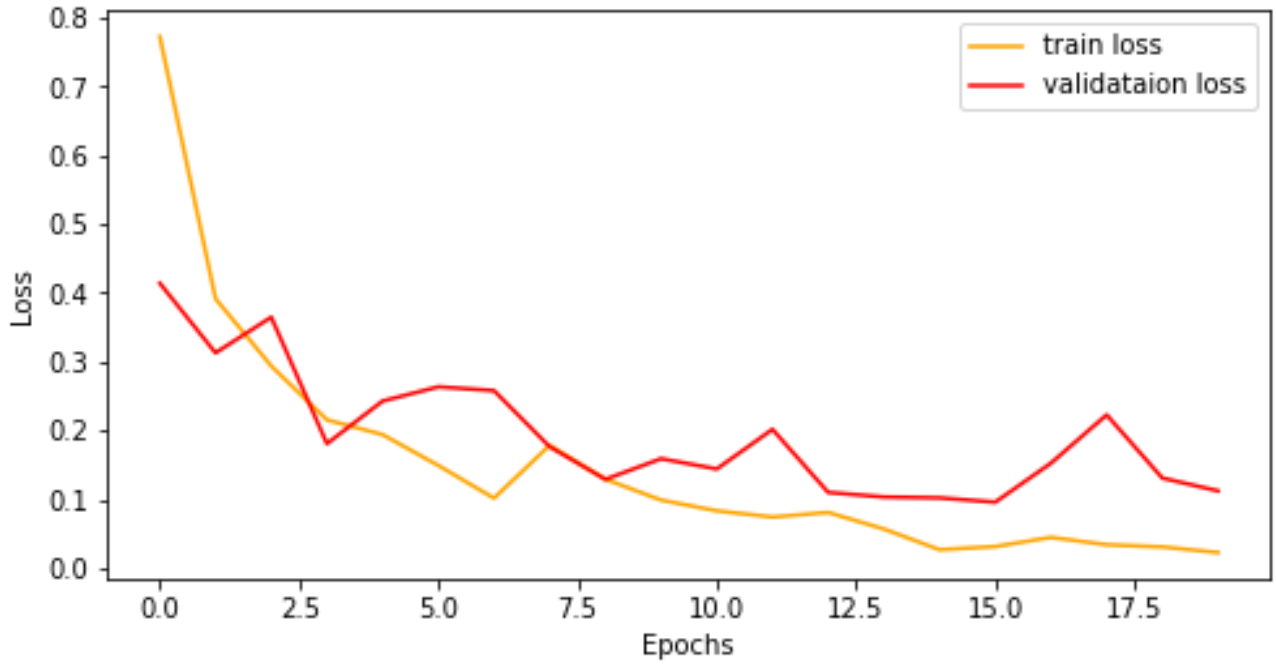


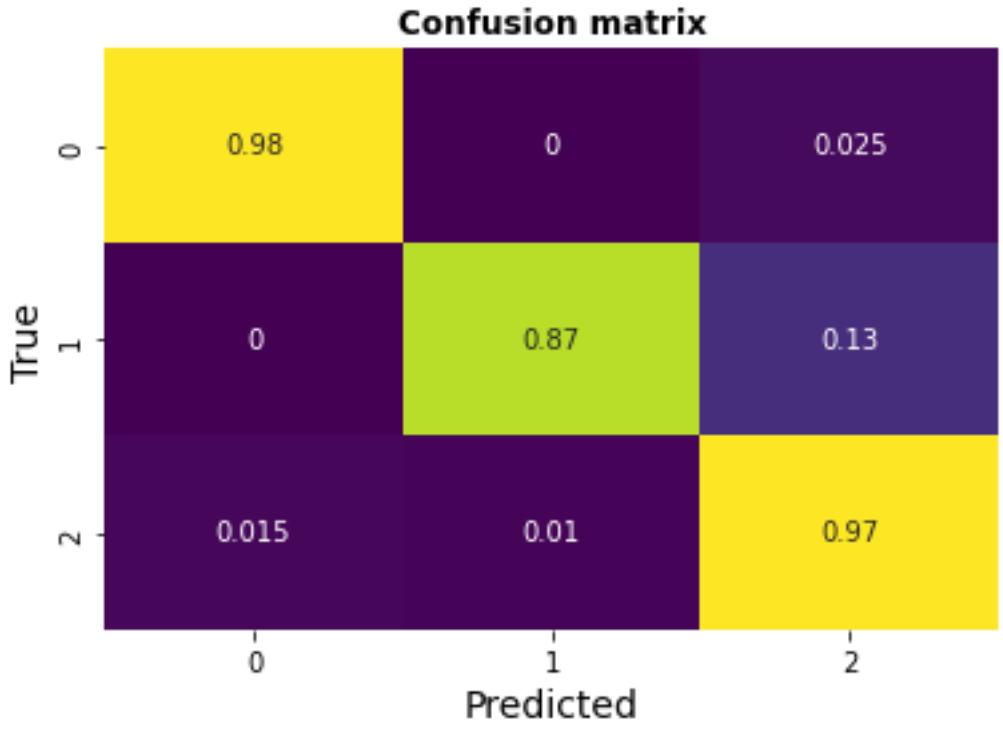
Model Training

In [6]:
```
# Configure the dataset for performance

AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

In [7]:
```
# Model architecture

num_classes = 3

model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(img_height, img_width, 3)),
    tf.keras.layers.experimental.preprocessing.Rescaling(1./255),

    tf.keras.layers.Conv2D(16, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(),

    tf.keras.layers.Conv2D(32, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(),

    tf.keras.layers.Conv2D(64, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(num_classes, activation='softmax')

])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

earlystop_callback = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy',
                                                      min_delta=0.0001,
                                                      patience=5)

history = model.fit(train_ds,
                    validation_data=val_ds,
                    epochs=20,
                    callbacks=[earlystop_callback])
```








```
In [25]: print(classification_report(correct_labels, predicted_labels))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       204
           1       0.93      0.87      0.90        31
           2       0.95      0.97      0.96       195

    accuracy                           0.97       430
   macro avg       0.96      0.94      0.95       430
weighted avg       0.97      0.97      0.97       430
```

Applicable Regulations

Regulatory Compliance for Potato Disease Detection App in India

Data Privacy and Protection:
• Personal Data Protection Bill (PDPB): App must secure explicit user consent for data collection and ensure transparency and security.

Agricultural Regulations:
• Insecticides Act, 1968: Recommendations for disease treatments must comply with approved pesticide use.
• Seeds Act, 1966: App should help farmers identify disease-free seeds and offer best practices for seed treatment and storage.

Environmental Regulations:
• Environment (Protection) Act, 1986: Promote sustainable disease management practices.
• Water and Air Pollution Acts: Prevent environmental pollution.

Health and Safety Regulations:
• FSSAI: Ensure disease treatments align with food safety standards.
• Occupational Safety Code, 2020: Provide safe usage guidelines for pesticides and treatments.

Intellectual Property Rights:
• Indian Patent Act, 1970: Ensure technologies do not infringe on existing patents.
• Copyright Act, 1957: Ensure all app content is original or properly licensed.

Electronic and Information Technology Regulations:
• Information Technology Act, 2000: Ensure secure data transmission and storage.
• Digital India Initiative: Promote digital literacy and reliable tools for farmers.

Applicable Constraints

Developing a Mobile or Web Application for Potato Disease Detection in India

Space Constraints:
• Efficient storage for high-resolution images, user data, and machine learning models.
• Network optimization for offline functionality in rural areas.

Budget Constraints:
• Development Costs: Investment in software development, machine learning model training, and maintenance.
• Operational Costs: Budget for cloud services, server maintenance, cybersecurity, and user support.
• Hardware: Compatibility with a wide range of smartphones, including low-cost devices.
• Marketing and Training: Allocate resources for app promotion and farmer training programs.

Expertise Constraints:
• Technical Expertise: Skilled professionals in software development, machine learning, and agricultural science.
• User Training: Design user-friendly interfaces and conduct training sessions.
• Support and Maintenance: Provide ongoing technical support, updates, and model retraining.

Business Opportunity

Smartphone or web Application for Detecting Potato Diseases in India's Agriculture

• Fills a crucial need in India's agriculture industry.

• Offers significant commercial opportunity.

• Improves crop health management, positively impacting India's GDP.

• Provides fast, accurate disease diagnosis and treatment suggestions.

• Monetization ideas include premium features, subscription models, and alliances with agricultural groups.

• Potential expansion to include more crops and geographical areas.

• Promotes agricultural sustainability and food security.

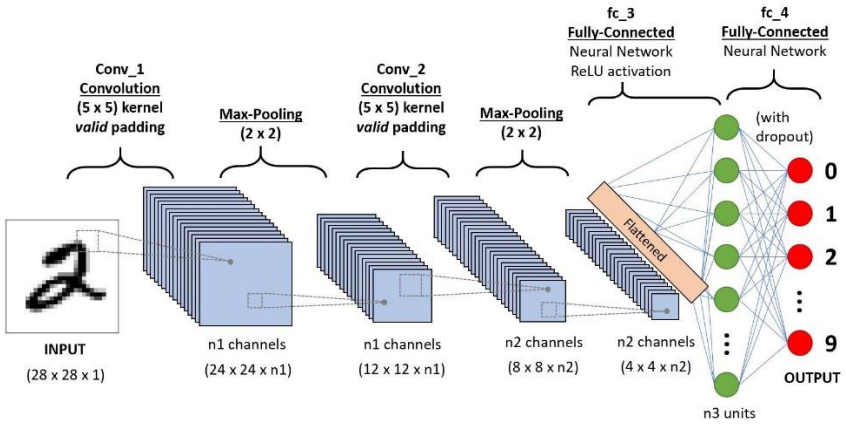
Concept Generation

For successful implementation, the proposed service will require the following algorithms, tools and experts.

Algorithms:

CNN

CNNs for Classification and Recognition of Images
• Deep learning algorithms for categorization and recognition of images.
• Use backpropagation to discover the spatial hierarchies of features.
• Employ fully connected layers, pooling, and several convolutional layers.
• Good at spotting features and patterns in photos.
• Robust feature extraction; • Able to handle substantial amounts of picture data.



Tools:

- → Python: It's a programming language that will be used for building the service.
- → Pandas: Pandas is a library mainly used for handling, manipulating and transforming data.
- → Scikit-learn: It is the gold standard library for machine learning which comes with plenty of algorithms to perform different tasks such as regression, classification etc.
- → Matplotlib and Seaborn: Both of these libraries are used for visualization purposes.

Team:

• **Project Manager**: Oversees project milestones and budget allocation.
• **Machine Learning Engineers**: Develop and refine disease detection models.
• **Mobile and WebApp Developers**: Responsible for app creation and ML model integration.
• **Backend Developers**: Handle data management and server-side functionalities.
• **UI/UX Designers**: Design intuitive interfaces and ensure user-friendly experiences.
• **Agricultural Experts**: Provide domain-specific insights and validate model accuracy.
• **Quality Assurance Testers**: Conduct thorough testing to ensure app functionality and user satisfaction.

Links:

Github