

**UET Mardan**

**Lab Project**

Submitted to: Engr.Zafar Ali Shah

Submitted by: Adil Hayat

Reg No: 21MDSWE173

Subject: Machine Learning lab

Department: Computer Software Engineering

**Step-by-Step Guide for Implementing Machine Learning Models on the Plant Village Dataset**

**Step 1: Upload the Dataset**

* Load the **Plant Village Dataset**, which contains images of tomato leaves for disease classification.
* The dataset includes healthy leaves and leaves affected by various diseases.

**Step 2: Explore the Data**

* **Total Images:** The dataset contains 4,872 images.
* **Number of Classes:** The dataset is categorized into the following classes:
  + Tomato\_healthy
  + Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite
  + Tomato\_Target\_Spot
  + Tomato\_Septoria\_leaf\_spot
  + Tomato\_Tomato\_mosaic\_virus
  + Tomato\_Leaf\_Mold
  + Tomato\_Bacterial\_spot
  + Tomato\_Late\_blight
  + Tomato\_Early\_blight
  + Tomato\_Tomato\_YellowLeaf\_Curl\_Virus

**Step 3: Convert Images to Numerical Data**

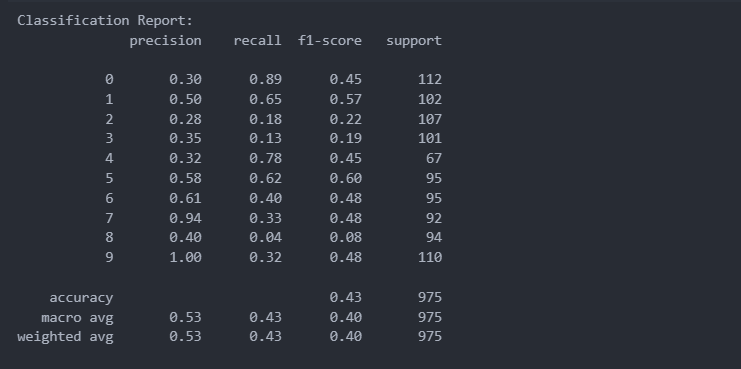
* Since the data consists of images, extract numerical features for classification:
  + Use **Histogram of Oriented Gradients (HOG)** to extract features from each image.
  + **Function used**: extract\_hog\_features(image\_path)
    - Converts images to grayscale.
    - Resizes images to a fixed size (e.g., 64x64 pixels).
    - Extracts the HOG feature vector.

**Step 4: Standardize and Split the Data**

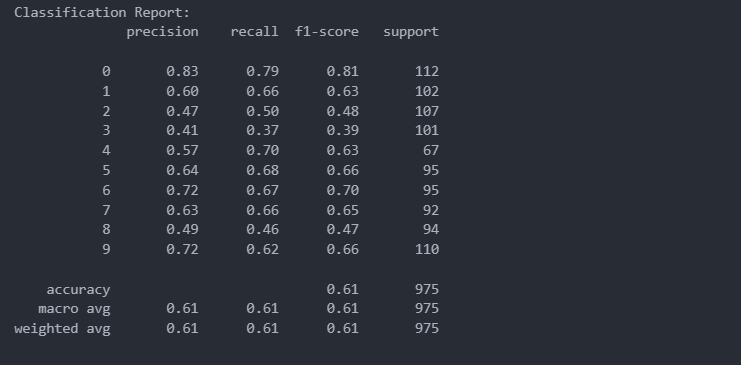
1. **Standardize Features:**
   * Normalize the HOG feature vectors to ensure consistent scaling across features.
2. **Train-Test Split:**
   * Split the standardized data into training and testing subsets.
   * Typical split ratio: **80% training, 20% testing**.

**Step 5: Train and Evaluate Models**

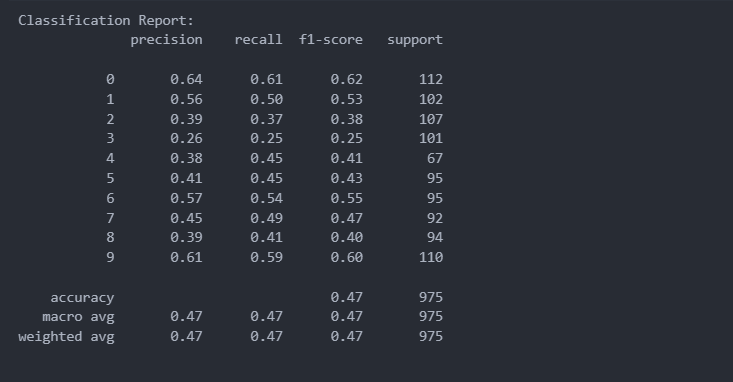
**1. k-Nearest Neighbors (k-NN)**

* **Training:** Trained using the extracted HOG features.
* **Accuracy Score:** **42.67%**
* **Classification Report (k-NN):** 

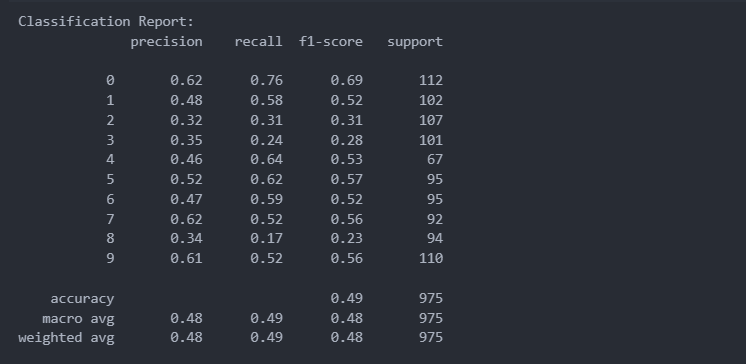
**2. Support Vector Machine (SVM)**

* **Training:** Trained using a Radial Basis Function (RBF) kernel.
* **Accuracy Score:** **60.92%**
* **Classification Report (SVM):** 

**3. Logistic Regression**

* **Training:** Used a logistic regression model with default parameters.
* **Accuracy Score:** **46.87%**
* **Classification Report (Logistic Regression):** 

**4. Random Forest**

* **Training:** Used a random forest classifier with 100 estimators.
* **Accuracy Score:** **49.23%**
* **Classification Report (Random Forest):** 

**Step 6: Observations**

* Among the models tested, **SVM** achieved the highest accuracy of **60.92%**.
* The k-NN model performed with the lowest accuracy (**42.67%**) but offers simplicity in implementation.

**Document for Training CNN Models with ResNet50, InceptionV3, and Custom Architecture**

**Step 1: Load the Dataset**

1. Load the **Plant Village Dataset**, containing images of tomato leaves.
2. Separate the dataset into:
   * **X (independent variables):** Image data.
   * **y (dependent variables):** Labels representing the classes.

**Step 2: Dataset Exploration**

* **Total Data Points:** 4,872 images.
* **Classes:**
  + Tomato\_healthy
  + Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite
  + Tomato\_Target\_Spot
  + Tomato\_Septoria\_leaf\_spot
  + Tomato\_Tomato\_mosaic\_virus
  + Tomato\_Leaf\_Mold
  + Tomato\_Bacterial\_spot
  + Tomato\_Late\_blight
  + Tomato\_Early\_blight
  + Tomato\_Tomato\_YellowLeaf\_Curl\_Virus

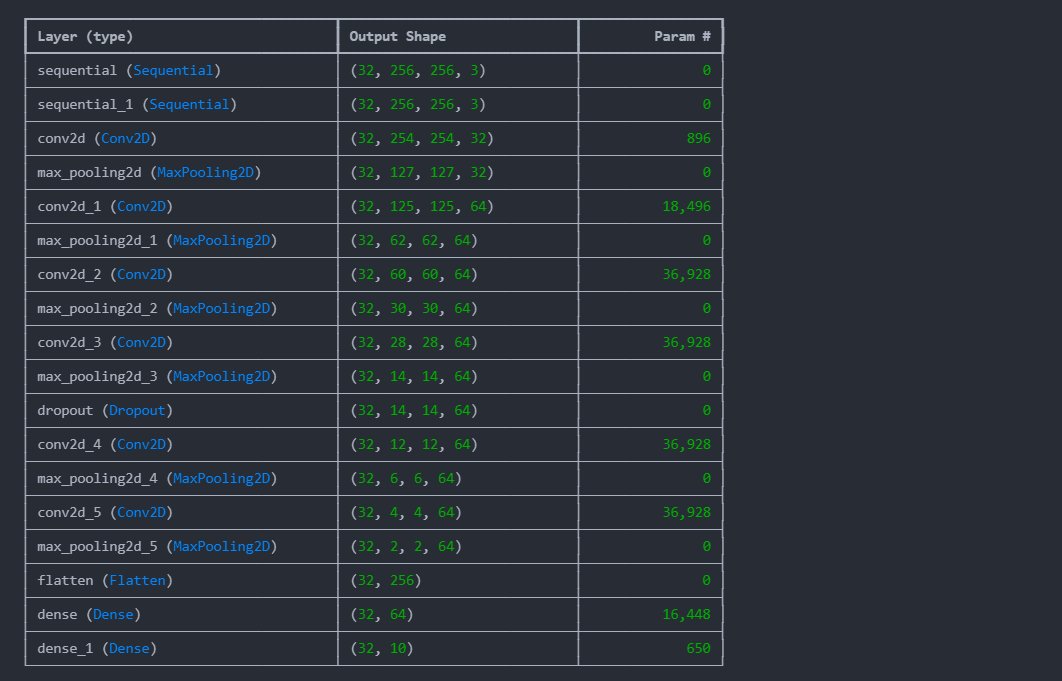
**Step 3: Train-Test Split and Preprocessing**

1. **Split the Data:**
   * Training and testing data split in an **80%-20% ratio**.
2. **Preprocess Images:**
   * Resize all images to **256x256 pixels**.
   * Normalize pixel values to the range [0, 1].
3. **Apply Data Augmentation:**
   * **Augmentation Techniques:** Horizontal flip, rotation, zoom, and shifting.

**Step 4: Model Architectures and Training Results**

**Custom CNN Architecture**

* **Architecture:**



* **Training Results:**
  + **Epochs:** 30
  + **Training Accuracy:** 93.79%
  + **Validation Accuracy:** 91.33%
  + **Test Accuracy:** 94.40%
  + **Test Loss:** 0.1513

**InceptionV3 Transfer Learning**

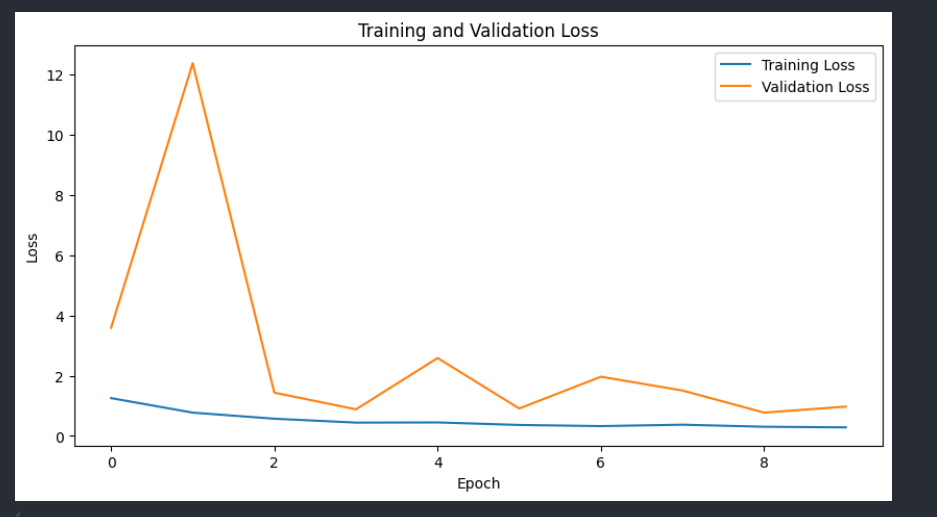
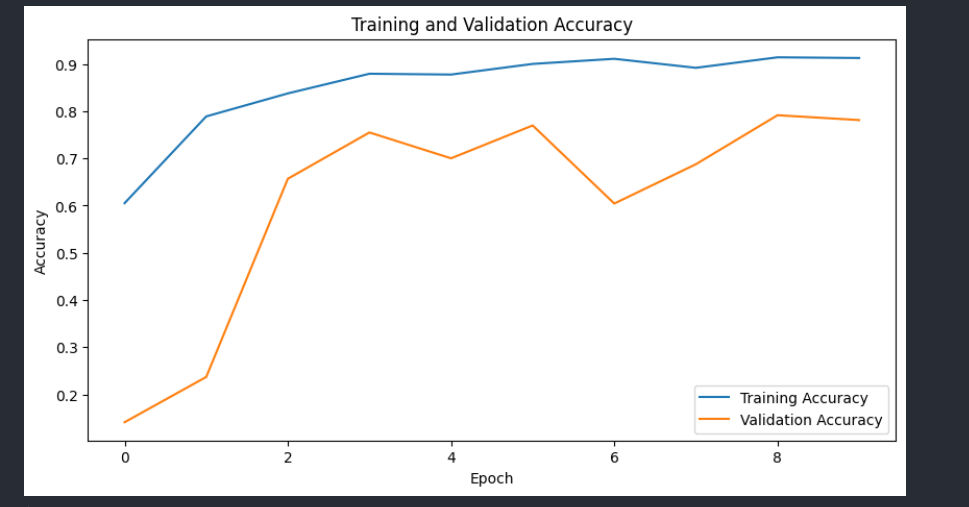
* **Model Description:**
  + Pre-trained **InceptionV3** with a custom dense layer for 10-class classification.
* **Training Results:**
  + **Epochs:** 10
  + **Training Accuracy:** 91.83%
  + **Validation Accuracy:** 78.11%
  + **Training Loss:** 0.2744
  + **Validation Loss:** 0.9733
  + **Test Results:**
    - **Test Accuracy:** 7.73%
    - **Test Loss:** 14.3094

**ResNet50 Transfer Learning**

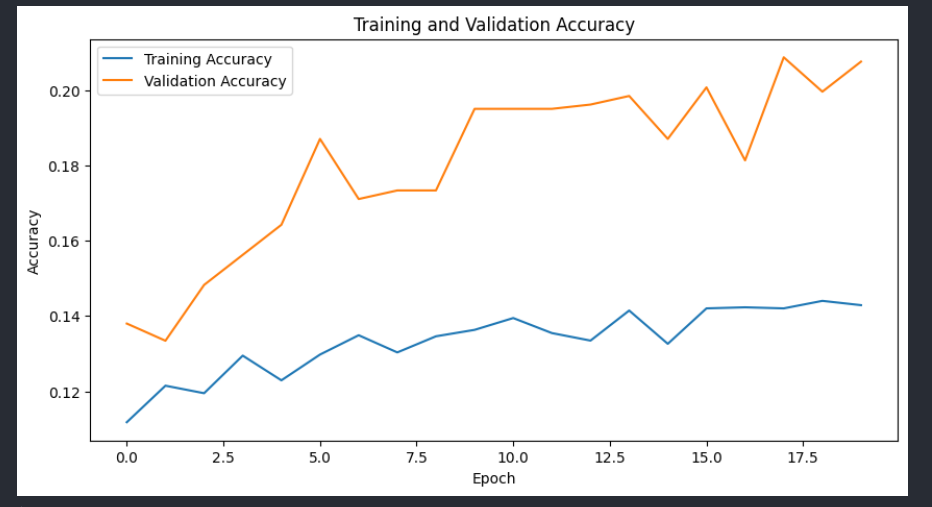
* **Model Description:**
  + Pre-trained **ResNet50** with a modified classification head for 10 classes.
* **Training Results:**
  + **Epochs:** 20
  + **Training Accuracy:** 13.88%
  + **Validation Accuracy:** 20.75%
  + **Training Loss:** 2.2191
  + **Validation Loss:** 2.1549
  + **Test Results:**
    - **Test Accuracy:** 8.86%
    - **Test Loss:** 9.5750

**Plots:**

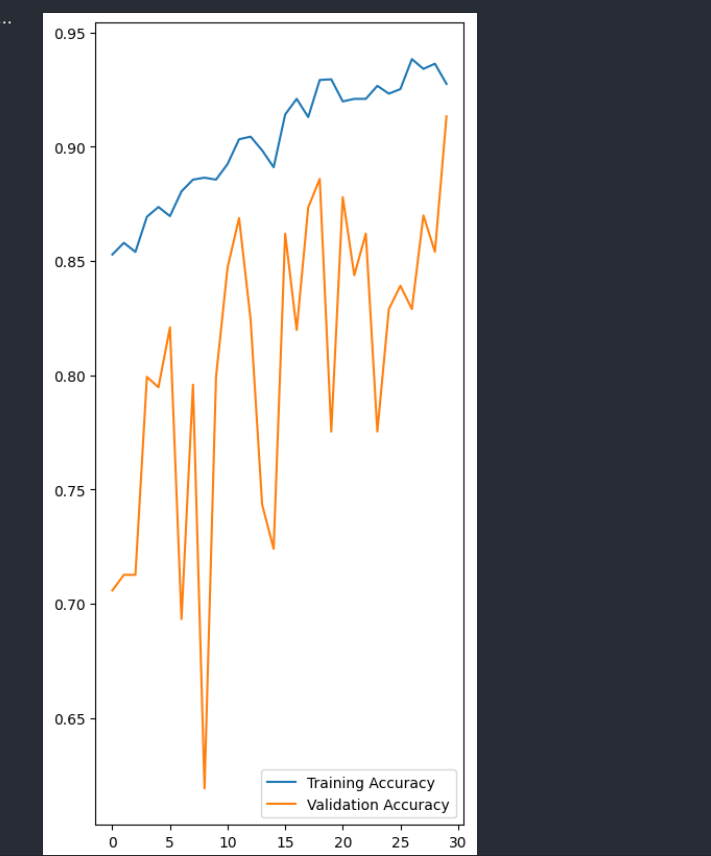
**Inception v3:**

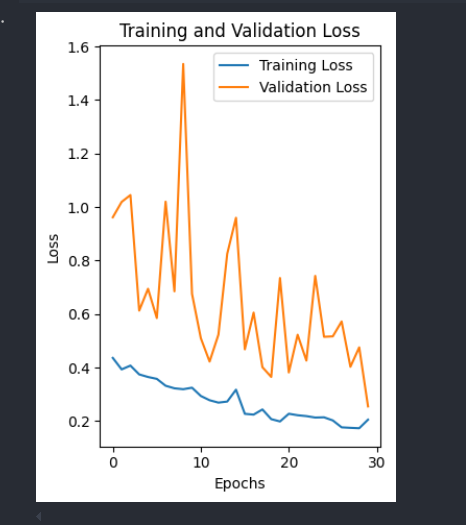


**Resnet 50:**



**CNN:**





**Conclusion: Results and Observations from Machine Learning Models on the Plant Village Dataset**

**Step 1: Initial Model Training with Traditional Machine Learning Algorithms**

* The **k-Nearest Neighbors (k-NN)**, **Support Vector Machine (SVM)**, **Logistic Regression**, and **Random Forest** models were trained using HOG (Histogram of Oriented Gradients) features extracted from the dataset.
* **Results:**
  + **k-NN:** Accuracy of 42.67%
  + **SVM:** Accuracy of 60.92% (Highest among traditional models)
  + **Logistic Regression:** Accuracy of 46.87%
  + **Random Forest:** Accuracy of 49.23%
* **Observation:**
  + These models performed poorly because they rely on handcrafted features (HOG), which might not capture the complex patterns and nuances in image data.

**Step 2: Training with Pre-Trained CNN Architectures**

* We used **InceptionV3** and **ResNet50** (transfer learning) for disease classification, expecting better results due to their ability to learn features from large-scale datasets.
* **Results:**
  1. **InceptionV3:**
     + Training Accuracy: 91.83%
     + Validation Accuracy: 78.11%
     + Test Accuracy: 7.73%
     + Test Loss: 14.3094
  2. **ResNet50:**
     + Training Accuracy: 13.88%
     + Validation Accuracy: 20.75%
     + Test Accuracy: 8.86%
     + Test Loss: 9.5750
* **Observation:**
  1. Both InceptionV3 and ResNet50 failed to generalize well on the test data, leading to low test accuracies.
  2. The likely cause is insufficient fine-tuning of the pre-trained models for the specific task and dataset size. Pre-trained models are not always optimal for smaller, domain-specific datasets without substantial fine-tuning.

**Step 3: Custom CNN Architecture**

* To overcome the limitations of traditional ML and pre-trained models, we developed a **custom CNN architecture** tailored specifically for the Plant Village dataset.
* **Results:**
  + Training Accuracy: 93.79%
  + Validation Accuracy: 91.33%
  + Test Accuracy: 94.40%
  + Test Loss: 0.1513
* **Observation:**
  + The custom CNN outperformed all other models, achieving the highest test accuracy. This highlights the effectiveness of designing an architecture specific to the dataset, enabling the model to learn intricate patterns and features unique to tomato leaf diseases.

**Final Conclusion:**

1. **Traditional Machine Learning Models:**
   * Models like k-NN, SVM, Logistic Regression, and Random Forest performed poorly due to their reliance on handcrafted features, which failed to capture the complexity of the image data.
2. **Pre-Trained CNN Models (InceptionV3 and ResNet50):**
   * These models showed poor generalization on the test set due to limited fine-tuning and possible overfitting on the training data. Pre-trained models may require a larger dataset or domain-specific fine-tuning for better performance.
3. **Custom CNN Architecture:**
   * The custom CNN model demonstrated the best results, with a test accuracy of 94.40%. The tailored architecture effectively learned the specific patterns in the dataset, making it the most suitable model for this classification task