Fruit/Vegetables Classification and Recipe Generation

A Machine Learning and Deep Learning Based Approach

Project By:

Kachhawa Goutham
Data Science Enthusiast
Email: kachhawa.1@iitj.ac.in
GitHub:
LinkedIn:

Introduction:

The objective of this project is to classify images of fruits and vegetables into 36 distinct categories and subsequently generate recipes based on the identified fruits or vegetables. The data used in this project was sourced from Kaggle, consisting of images of 36 classes. After classifying the images using various ML and Deep Learning techniques, the project leverages the **Edamam API** to generate recipes based on the predicted items.

Machine Learning Pipeline Overview:

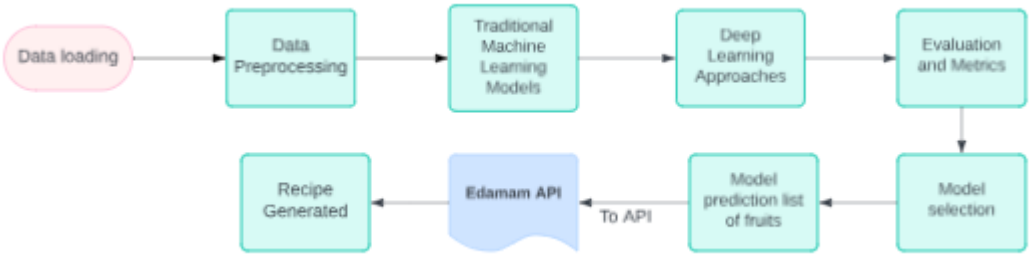

Fig: Pipeline

Data Preprocessing:

Preprocessing is crucial to ensure that the data is ready for input into machine learning models. In this case, the following preprocessing steps were applied:

Normalization: The pixel values of the images were scaled to a range of [0, 1] by dividing by 255. This ensures that the data has a consistent scale, which improves the stability and performance of the models.

Flattening (Traditional ML): For the traditional machine learning models like SVM, KNN, etc., the images were flattened into 1D arrays, converting each image from a 2D structure to a linear vector.

Traditional Machine Learning Models:

Several traditional ML models were implemented to classify the fruits which includes:

K-Nearest Neighbors (KNN):

KNN was trained on the flattened images. The parameter n_neighbors was tuned using validation accuracy. The model achieved a **Validation Accuracy of 97.1%** and a **Testing Accuracy of 97.2%**.

Random Forest (RF):

RF performed comparably well, with hyperparameter tuning of n_estimators. The model achieved a **Validation Accuracy of 96.5%** and a **Testing Accuracy of 96.6%**.

Naive Bayes:

This model performed poorly, with an accuracy of just 26.7%, indicating it was not suited for this image classification task dataset.

Logistic Regression:

This model outperformed well with a **Validation Accuracy of 96.0%** and a **Testing Accuracy of 96.1%**.

To improve robustness, an **ensemble model (Voting Classifier)** was created using the top-performing models (KNN, Random Forest, and Logistic Regression). The ensemble improved the accuracy to **97.2%** on the test set.

Evaluation Matrix for Voting Classifier:

Precision of the Model: 0.9731762065095398

It means the model has a high precision metric; it means that the model is good at correctly identifying positive instances out of all the instances that it has predicted as positive.

Recall of the Model: 0.9719135802469138

A high recall indicates that when there is a positive instance in the data, the model is highly likely to identify it correctly.

F1-Score of the Model: 0.9715538847117795

It is worth noting that precision and recall are often trade-offs, and it is important to consider both metrics together when evaluating a model. For example, a model with high recall may have a lower precision, meaning that it may identify more positive instances but also have more false positives. Similarly, a model with high precision may have a lower recall, meaning that it may identify fewer positive instances but also have fewer false positives. The choice between precision and recall will depend on the specific requirements of the problem at hand. But for this particular dataset both are high, and it is overfitting the dataset hence sometimes when new images are offered to the model it falls.

Data Augmentation was used for the generating more images to solve the problem of overfitting in the deep learning models.

Deep Learning Approaches:

Once the traditional approaches have been utilized, Various Deep Learning models were used for better performance.

Artificial Neural Networks (ANN)

A **Multi-Layer Perceptron (MLP)** model was built with the following architecture:

- Input layer: Image resized to 28x28x3.
- Two hidden layers with 3000 and 1000 neurons, both using ReLU activation.
- Output layer with 36 neurons (for 36 classes) using Softmax activation.

The model was trained for 80 epochs using **SGD** as the optimizer and **sparse categorical cross-entropy** as the loss function. The final accuracy was **95.2%** on the validation set.

Graph of training vs validation accuracy with respect to epochs and cross entropy vs epochs of training and validation accuracy.

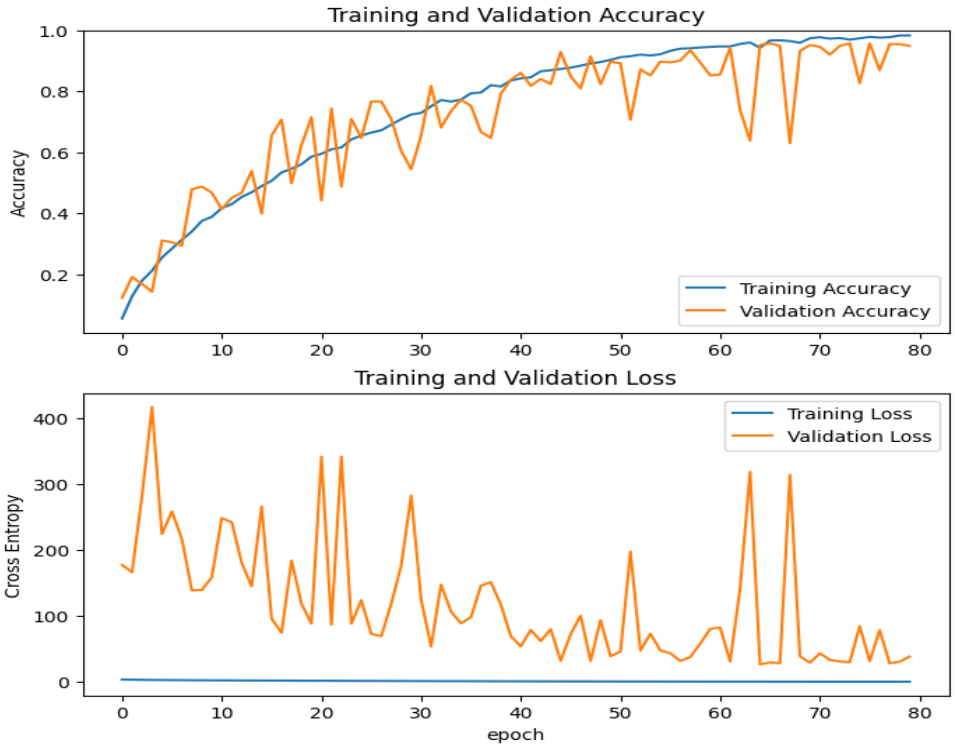

Fig: ANN Training

Convolutional Neural Networks (CNN)

CNNs were used due to their success in image classification tasks. The architecture included:

- Two **Conv2D** layers with 32 and 64 filters respectively, followed by MaxPooling.
- A **Flatten** layer followed by dense layers.
- Output layer with 36 neurons using SoftMax for classification.

This CNN model achieved strong performance, but to reduce **overfitting**, further steps were taken, leading to **CNN with Data Augmentation**:

- **Augmentation Techniques**: Rescaling, flipping, shearing, rotation, and zooming.
- Four Conv2D layers with increasing filters (32, 64, 128, and 256) were used with ReLU activations.
- Technics like BatchNormalization and Dropout were used to reduce the overfitting.

This helped the model to generalize better results on unseen data.

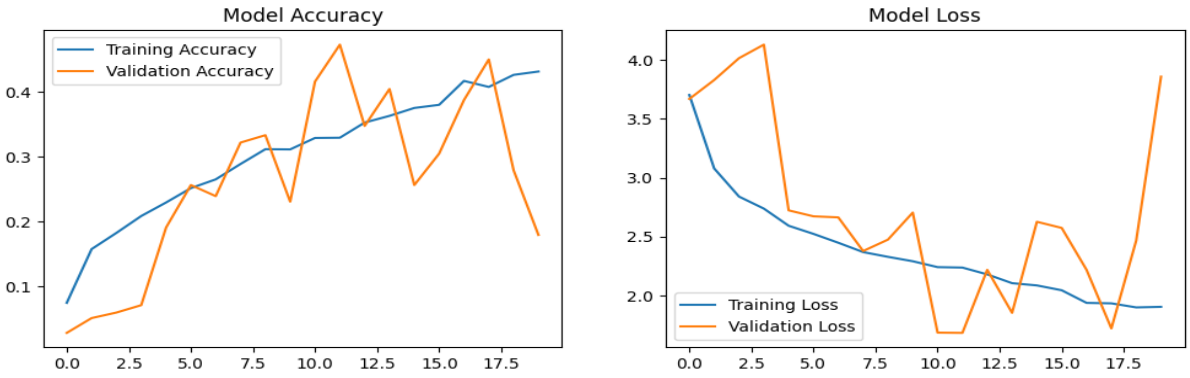

Fig: CNN Training

Transfer Learning with ResNet50

To achieve state-of-the-art performance **ResNet50**, a pre-trained deep learning model, was fine-tuned on the dataset:

- Images were resized to 256x256x3.
- A ResNet50 pre-trained model was used as the backbone.
- A dense layer with 512 neurons (ReLU) and an output layer with 36 neurons (Softmax) were added.

ResNet50, with its residual connections, mitigates the problem of vanishing gradients in deep networks, improving training efficiency and accuracy. Trained for 7 epochs, this model produced the best results without overfitting.

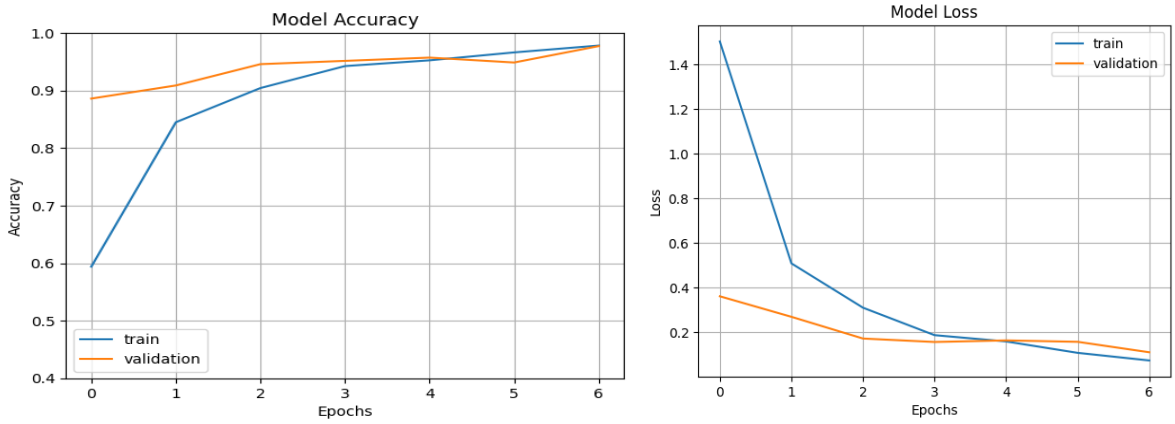

Fig: ResNet50 Training

Recipe Generation:

Once the fruit classification was complete, the identified fruit was passed to the **Edamam API**. This API, containing over 200,000 recipes, generates a recipe based on the list of detected fruits/vegetables from the image.

- **API Workflow**: Input image → Model predicts fruit → List of fruits generated → Passed to API → Recipe generated.

This seamless integration provides a useful application of the classification model, linking it directly to real-world use cases.

Conclusion:

The project successfully implemented a machine learning pipeline combining traditional models, deep learning, and state-of-the-art techniques like ResNet50. The use of data augmentation and transfer learning resulted in a robust and accurate fruit classification model. The extension to recipe generation through API integration adds real-world functionality, making the project a comprehensive solution for image-based recipe recommendation.