# AGRO-ML

AGRO-ML

Team: type-ers

Github repo: Agro-ML

Members:

- Sitanshu Shrestha
- Suvashish Shrestha
- Biraj Sapkota
- Abiral Manandhar

**Problem Statement**

AGRO-ML aims to replace time consuming and expensive services with free of cost and quick Artificial Intelligence services to small scale farmers through a web-based application.

**MODULES**

1. Leaf Disease Detection
2. Fertilizer Composer
3. Market Price Predictor

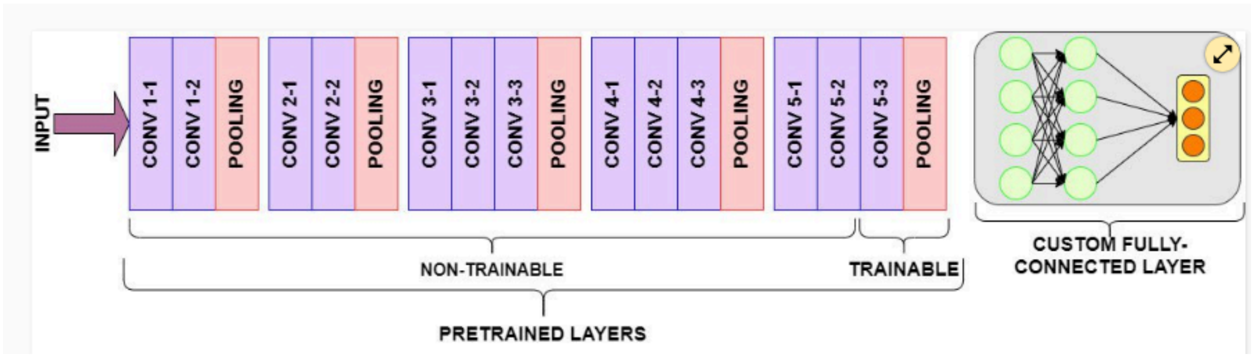
**1) Leaf Disease Detection**

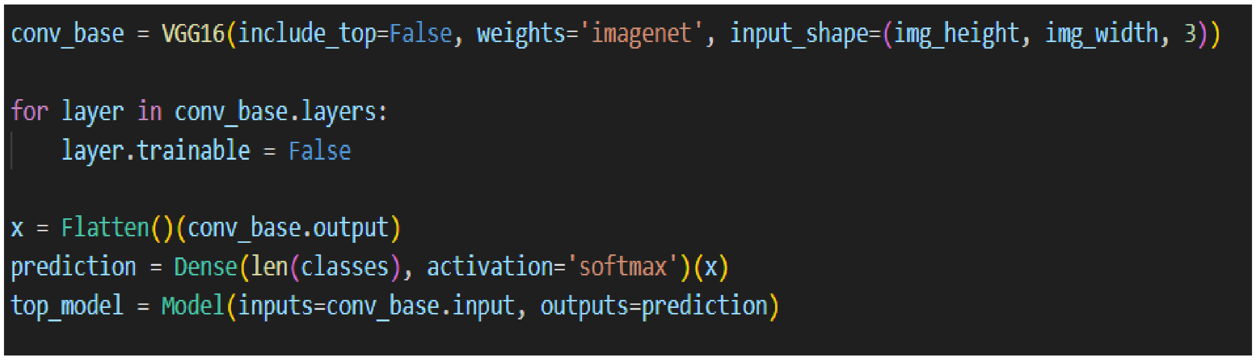
**Function**: It takes in images of leaves as inputs and puts it through a neural network in order to generate a prediction for the state of the leaf. The output could be “healthy” or a particular disease on which the model has been trained on.

**Overview:** It has been trained to detect the following diseases  for the following plants:

| Plant | Disease |
|---|---|
| Apple | Brown Spot,Mosaic |
| Tomato | Late Blight, Septoria Leaf Spot, Yellow Leaf Curl |
| Rice | Leaf Blight, Brown Spot,Leaf Blast, Leaf Scald, Narrow Brown Spot |
| Potato | Fungi, Nematode |
| Corn | Blight, Common Rust |

**Methodology:** A convolutional neural network model was trained for each plant by fine-tuning (involves adjusting the parameters of pre-trained layers to adapt them to a new task or dataset, typically by unfreezing some or all of the layers and retraining the model) the VGG16 convolutional neural network). Images of leaves of various diseases were fed into the network as an input.



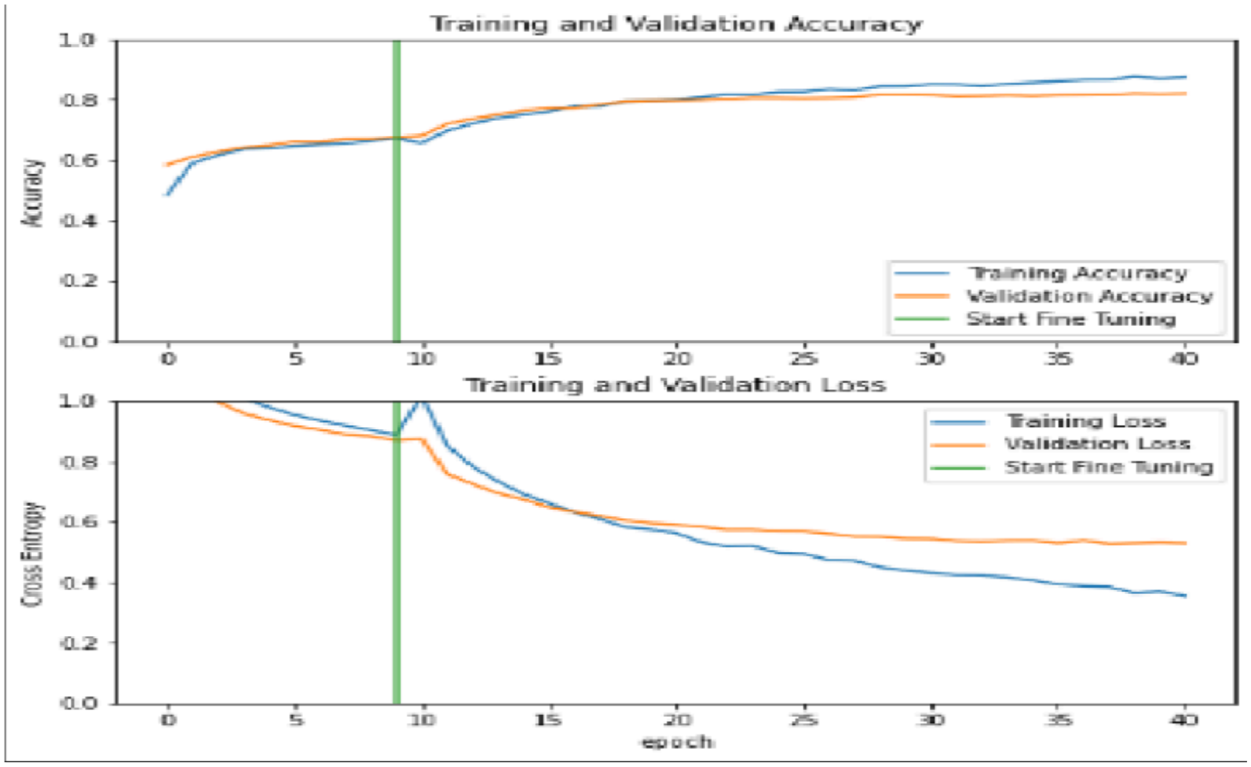


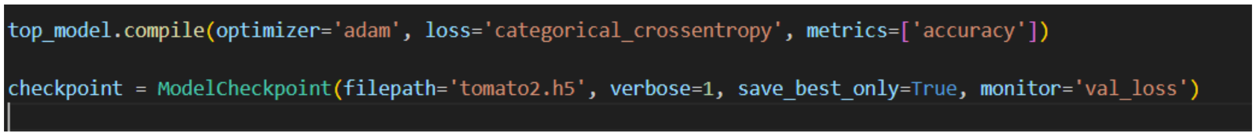
```python
conv_base = VGG16(include_top=False, weights='imagenet', input_shape=(img_height, img_width, 3))

for layer in conv_base.layers:
    layer.trainable = False

x = Flatten()(conv_base.output)
prediction = Dense(len(classes), activation='softmax')(x)
top_model = Model(inputs=conv_base.input, outputs=prediction)
```

1) Prediction: This line creates a dense layer with a softmax activation function.
   The Dense layer is a fully connected layer where each input neuron is connected to each output neuron.
   The len(classes) parameter specifies the number of output neurons, which typically corresponds to the number of classes in the classification task.
   Softmax activation is used to convert raw scores (logits) into probabilities for each class, ensuring that the output values sum up to 1.
2) top_model:
   This line creates a new model by specifying the inputs and outputs.
   The Model function constructs a Keras model from an input layer and an output layer. conv_base.input represents the input layer of the pre-trained convolutional base model (conv_base), while prediction represents the output layer created in the previous line.
   Thus, top_model is a new model that takes input from the pre-trained convolutional base and produces predictions using the dense layer defined above.

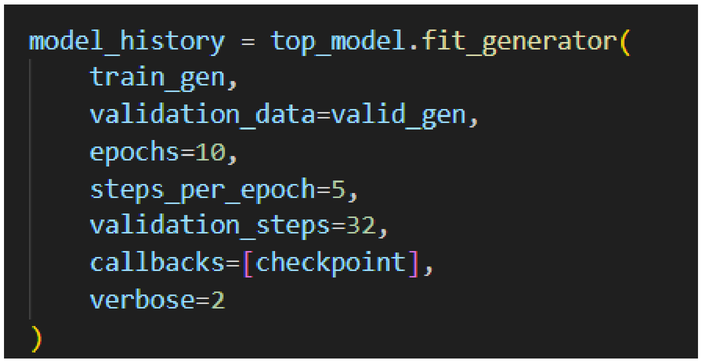




```
top_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

checkpoint = ModelCheckpoint(filepath='tomato2.h5', verbose=1, save_best_only=True, monitor='val_loss')
```

1) optimizer='adam': This line specifies the optimization algorithm used during training. Adam is a popular choice for its adaptive learning rate properties, making it efficient and effective for training deep neural networks.
2) loss='categorical_crossentropy': This line defines the loss function used to measure the difference between the model's predictions and the actual labels. Categorical cross-entropy is commonly used for classification tasks with multiple classes.
3) metrics=['accuracy']: This line specifies the evaluation metric to be monitored during training. In this case, it's accuracy, which measures the proportion of correctly classified images among the total number of images.



```
model_history = top_model.fit_generator(
    train_gen,
    validation_data=valid_gen,
    epochs=10,
    steps_per_epoch=5,
    validation_steps=32,
    callbacks=[checkpoint],
    verbose=2
)
```

1) train_gen: generator object that yields batches of training data. Generators are used to efficiently load data in batches, especially when dealing with large datasets that cannot fit into memory all at once.
2) validation_data:generator object that yields batches of validation data. Validation data is used to evaluate the model's performance on data that it hasn't seen during training,
3) epochs: determines the number of complete passes through the entire training dataset. Each epoch consists of one forward pass and one backward pass (optimization step) for all training samples.
4) steps_per_epoch: defines the number of batches to draw from the training generator in each epoch
5) validation_stepsspecifies the number of batches to draw from the validation generator at the end of each epoch for validation evaluation
6) callbacks: a list of callback functions to apply during training.
7) verbose:controls the verbosity mode during training.

- verbose=0: Silent mode, no output.
- verbose=1: Progress bar mode, displays a progress bar that updates after each epoch.
- verbose=2: One line per epoch mode, displays a single line of output for each epoch, showing the epoch number and training/validation metrics.

**Working:**

**Dataset:**

1) Corn Dataset
2) Apple Dataset
3) Tomato Dataset
4) Rice Dataset
5) Potato Dataset

**2) Fertilizer Composer**

**Function**: It takes in temperature, humidity, moisture, soil type and crop type as input and gives the prediction of the type of fertilizer to be used. This could be Urea, DAP or NPK (ratios of Nitrogen, Phosphorus and Potassium are given as ratio)

**Overview:** The values the parameters can take are listed below:

| **Parameter** | **Data Type** | **Values** |
|---|---|---|
| Temperature | float | any |
| Humidity | float | any |
| Moisture | float | any |
| Soil type | string | Black, sandy, clayey, red, loamy |
| Crop type | string | Oil seeds, ground nuts, pulses, millets, barley, sugarcane, maize, cotton, tobacco, paddy, wheat |

Methodology: Random Forest Classifier was used in order to classify the fertilizer to be used based on the input features such as temperature, humidity, moisture, soil type, and crop type.

**Working:**
api call:
http://127.0.0.1:8888/api/fertilizer?temperature=26&humidity=21&moisture=10&soil_type=Loamy&crop_type=Sugarcane&nitrogen=20&potassium=9&phosphorous=6

response: {"prediction": "17-17-17"}

**Dataset:**

**3) Market Price Predictor**

**Function:The Market Trend Analyzer is a tool designed to analyze and predict market trends for a specific commodity based on historical data. It focuses on providing insights into price fluctuations and seasonal patterns to help users make informed decisions regarding the commodity.**
**Overview: The dataset used for analysis contains the following information:**

| Date | Season | Price |
|---|---|---|
| 2021-01-01 | Winter | 50.00 |
| 2021-02-01 | Winter | 52.20 |
| 2021-03-01 | Spring | 49.80 |
| ... | ... | ... |

Market Trend Analyzer Documentation

## Overview

**The Market Trend Analyzer is a tool designed to analyze and predict market trends for a specific commodity based on historical data. It focuses on providing insights into price fluctuations and seasonal patterns to help users make informed decisions regarding the commodity.**

## Dataset

**The dataset used for analysis contains the following information:**

| Date | Season | Price |
|---|---|---|

| | | |
|---|---|---|
| 2021-01-01 | Winter | 50.00 |
| 2021-02-01 | Winter | 52.20 |
| 2021-03-01 | Spring | 49.80 |
| ... | ... | ... |

## Methodology

The Market Trend Analyzer utilizes a Random Forest Regressor model to predict market trends based on historical data. The following steps outline the methodology:

Data Collection: Historical market data for the specified commodity is collected from reliable sources.
Data Preprocessing: The collected data is cleaned and formatted, ensuring consistency and accuracy in analysis.
Feature Engineering: Additional features such as seasonal indicators may be derived from the date column to capture seasonal variations in market trends.
Model Training: A Random Forest Regressor model is trained using the preprocessed data, with the date and seasonal indicators as input features, and the commodity price as the target variable.
Prediction: Given a specific date, the trained model predicts the market price of the commodity for that date, providing insights into the market trend.

## Working

The Market Trend Analyzer operates as follows:

API Call: Users make an API call to the Market Trend Analyzer endpoint, providing the name of the commodity and the desired date for prediction.
Prediction: The Analyzer processes the input parameters, including the commodity name and date, and utilizes the trained Random Forest Regressor model to predict the market price for the specified commodity on the given date.

**Response: The Analyzer returns the predicted market price as the output, allowing users to gain insights into the market trend for the chosen commodity.**

## Example API Call

**API Endpoint:**

`http://127.0.0.1:8888/api/market_trend?commodity=Wheat&date=2022-01-01`

**Response:** `{"prediction": 52.00}`

**In this example, the Market Trend Analyzer predicts the market price of Wheat for January 1, 2022, to be $52.00.**

**Dataset:data2.csv**