PRML major project(B21CS021,B21CS022,B21CS087)

Topic: Creating recipe using given images.

Data Description: Data is divided into 3 folders namely Train,Test and Validation wit=ch contains 36 classes each. Each class has an average 60 images per class.Each image has an format of .jpg

Our pipeline/Approach: We have tried to explore different methods of Machine learning through our problem statement where we first started with Traditional Machine learning implementing different models like logistic regression,SVM,KNN etc and we hyperparameter tuned and used ensemble learning then we moved on to exploring the concepts of deep learning where we used ANNs and the Neural net specialty designed for Image classification CNNs then we tried Data augmentation to our curiosity we also tried to implement some State-of-Art architecture so we implemented ResNet50 to test on our data. And we used an API to give our recipe.

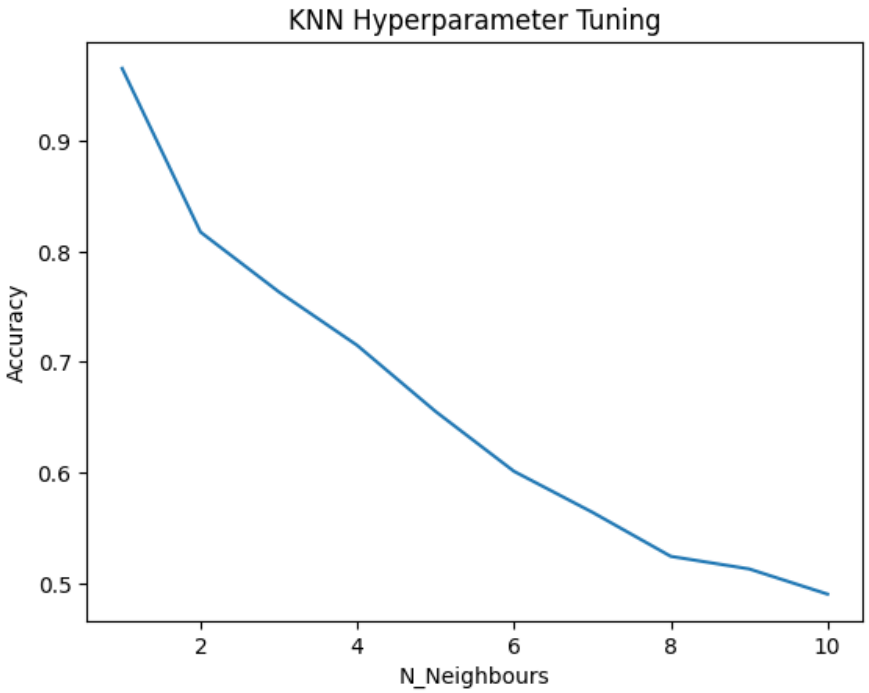
Part 1:- Traditional approach.

Idea: We iteraterated through each image and converted it into a tensor. We flattened the tensor and then trained our different models then we tuned our models and used an ensemble of our models.

Imported the data set from kaggle on google collab.
` Created Directory using tensorflow.
Visualization visualized data from different classes.

Pre processing: For preprocessing we Normalized the photo
: Appending images and labels in arrays
: Flattened images for applying traditional Machine learning approaches

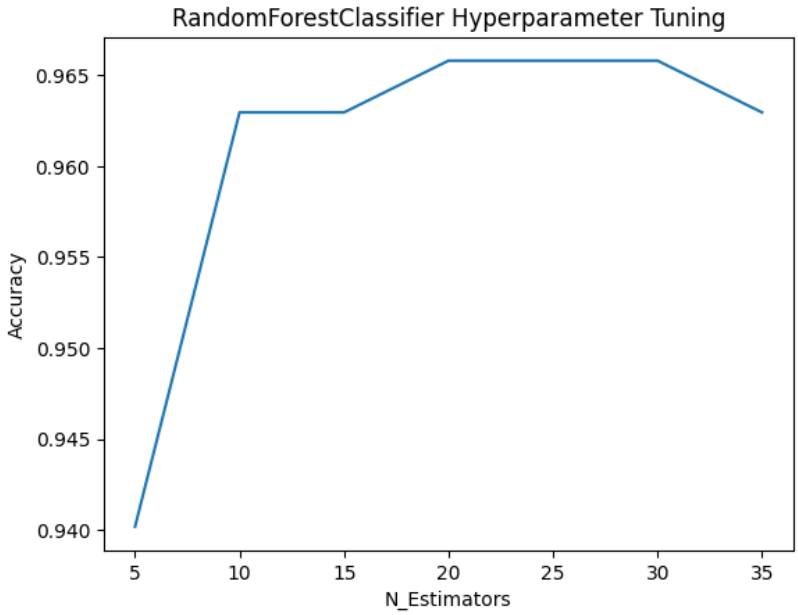
Model Training: Training KNN and tuning hyperParameters Where the parameter is n neighbors and evaluation matric is accuracy.
We get the following results



Validating the model on the tuned parameter we get

Validation Accuracy: 0.9658119658119658
Testing Accuracy: 0.9665738161559888

2.Next model we trained was Random forest we use the hyper parameter n-estimator the following result is:



The accuracy obtained for tuned parameter here was:
Validation Accuracy: 0.9658119658119658
Testing Accuracy: 0.9665738161559888

3. SVM for multiclass : decision_function_shape='ovo'
accuracy obtained where:Validation Accuracy: 0.8319088319088319
Testing Accuracy: 0.83008356545961

4. Naive bayes model didn't perform well with accuracy of 0.2
5. Logistic regression performed very well with accuracy of
Validation Accuracy: 0.9601139601139601
Testing Accuracy: 0.9610027855153204

We picked our top 3 models and used them for voting and used the ensemble of them to make our model more robust. The accuracy we obtained were

Validation Accuracy: 0.9715099715099715
Testing Accuracy: 0.9721448467966574

For prediction we have picked an image of apple and predicted with each model and have reported its prediction analysis cell and it is correctly classified.

Recipe generation: We have used EDMAMA Api which has over 2,00,000 recipes pre saved. We are generating a prompt to generate buy to images feed by us to generate a recipe.
We give the model the photos and a list of fruits/vegetables is generated this list is feed to the API and the recipe for the same is given.

We have saved the model and have integrated it in a website for easier use for the user.

We have calculated some metrics for evaluation of our model on testing data.

Precision of the Model : 0.9747755331088663

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.

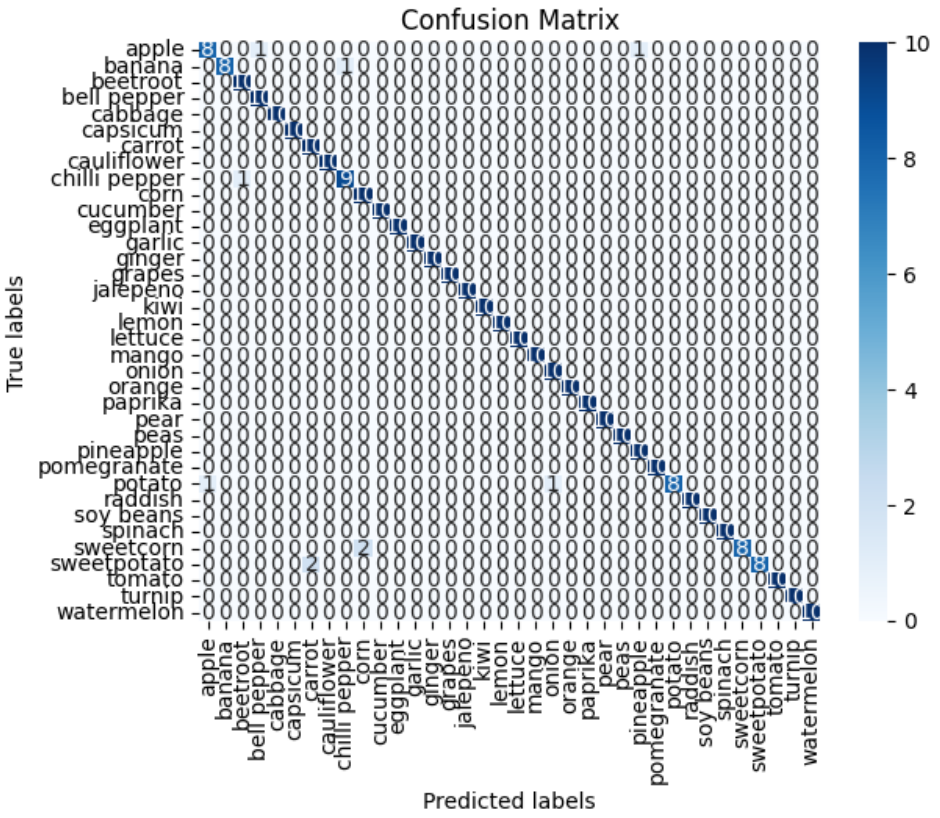
F1-Score of the Model : 0.9716015007810673

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 some times when new images are offered to the model it falls.

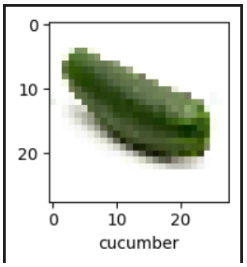
We have tried to solve this problem by augmenting the dataset later for generating more images.

Confusion Matrix:



ANN approach: Next we tried ANN/MLP to check that we get a better generalized model.

Steps involved:
pre processing and directory generation image size is reduced to 28*28 and batch size is set to 32 and the image is normalized.
The resulting image is

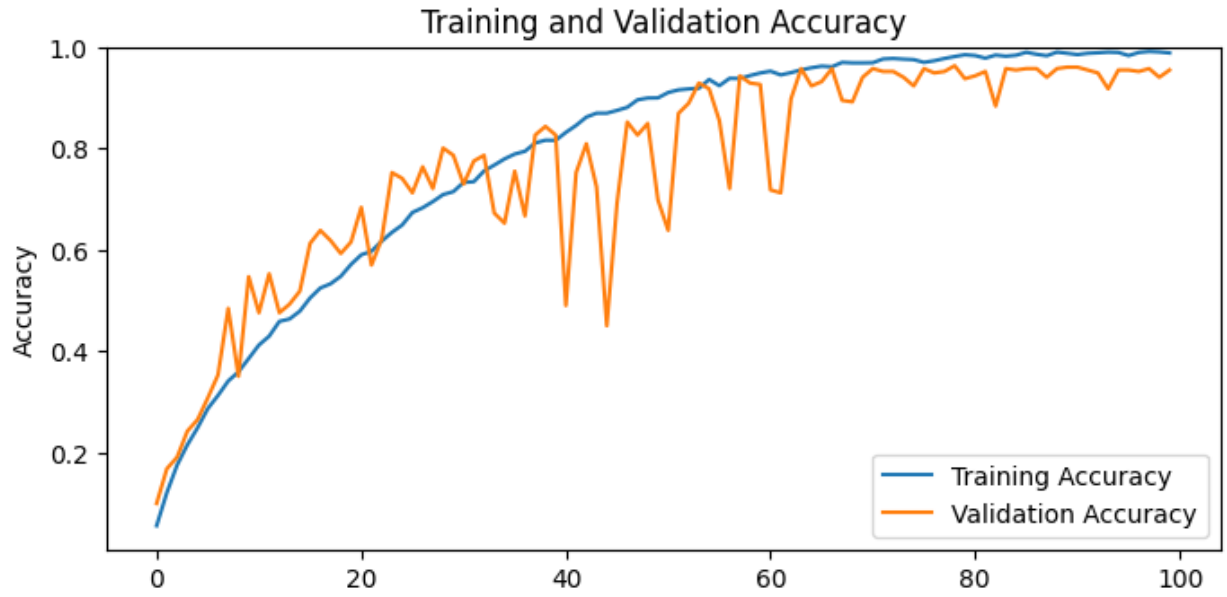


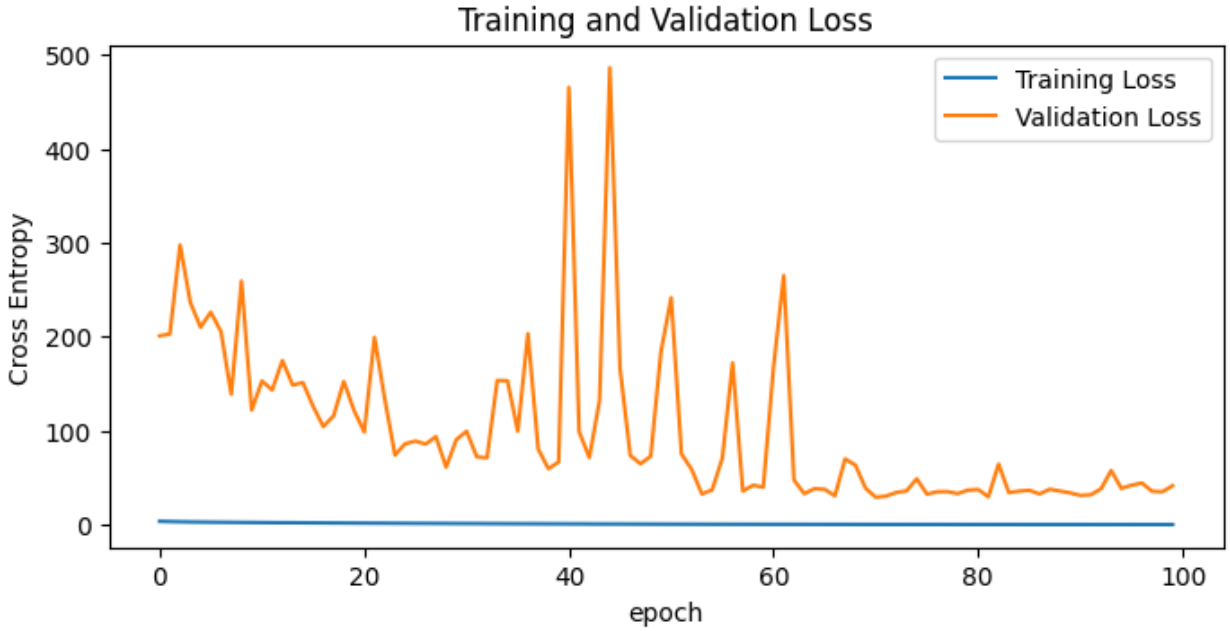
We have created a MLP of input layer of 28*28*3,
second layer of 3000 with activation layer Relu,
third layer of 1000 activation Relu and
36 output layer with activation softmax since it is multiclass classification

Optimizer used SGD and used loss function as sparse_categorical_crossentropy and evaluation metric as Accuracy

Trained model for 100 epochs and found final validation accuracy as 95.44 % and accuracy on test data as 0.9582172632217407

Then plotted a graph of training vs validation accuracy wrt epochs and cross entropy vs epochs of training and validation accuracy.


Training and Validation Accuracy
Accuracy
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100
Training Accuracy
Validation Accuracy


Training and Validation Loss
Cross Entropy
500
400
300
200
100
0
0
20
40
60
80
100
epoch
Training Loss
Validation Loss

CNN Approach: Then we tried to check for cnn

We tried a CNN with two convolutional layers, two max-pooling layers, and two fully connected layers, followed by a softmax activation function for multiclass classification.

The input layer of the network is a `Conv2D` layer with 32 filters, a kernel size of (3,3), and a rectified linear unit (ReLU) activation function. This layer expects input images of size 256 x 256 x 3, where 3 represents the number of color channels in the image (RGB).

After the input layer, there is a `MaxPooling2D` layer that performs spatial downsampling by taking the maximum value in a 2x2 window. This layer reduces the spatial dimensions of the feature maps by a factor of 2.

Following the first pooling layer, there is another `Conv2D` layer with 64 filters, a kernel size of (3,3), and a ReLU activation function.

Another `MaxPooling2D` layer follows the second convolutional layer, which reduces the spatial dimensions of the feature maps by a factor of 2.

Next, there is a `Flatten` layer that flattens the output of the previous layer into a 1D vector.

After flattening the output, there is a fully connected `Dense` layer with 64 neurons and a ReLU activation function.

Finally, there is an output `Dense` layer with 36 neurons (equal to the number of classes) and a `softmax` activation function. The `softmax` function converts the outputs of the last layer into a probability distribution over the 36 classes.

The model Performed Well on Given Dataset but not on custom so we tried to use cnn with augmentation as it was overfitting on the data set.

CNN With augmentation:

What Augmentation Tricks we applied:

1: Rescaled :The pixel values in the input images are scaled by dividing them by 255 to ensure that they are in the range [0, 1]. This is done using the `rescale` parameter.

2: Horizontally flipped the Images

3:applied random shearing transformations to the images during training, which can help increase the robustness of the network to deformations in the input data.

4:applied random rotations to the images during training, which can help increase the variation in the training data and prevent overfitting.

5:applied random zooms to the images during training, which can help increase the variation in the training data and prevent overfitting.

6:Applied height Shift and width shift

We applied four Conv2D layers.

The first Conv2D layer has 32 filters, a kernel size of (3,3), and a ReLU activation function.

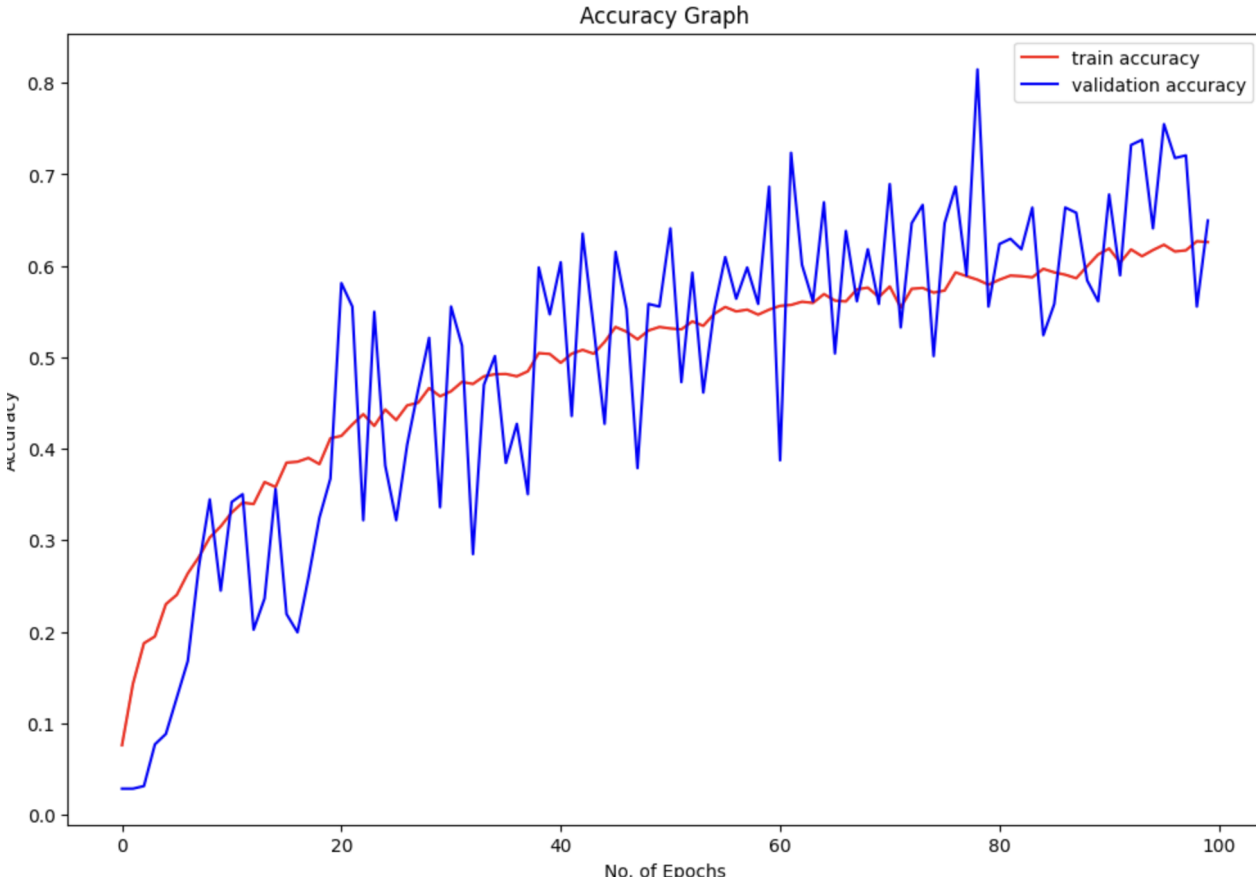
The second Conv2D layer has 64 filters, a kernel size of (3,3), and a ReLU activation function.

The third Conv2D layer has 128 filters, a kernel size of (3,3), and a ReLU activation function.

The fourth Conv2D layer has 256 filters, a kernel size of (3,3), and a ReLU activation function.

Each convolutional layer is followed by a max pooling layer that reduces the spatial dimensions of the output by a factor of 2. There are also several other layers in the model, including batch normalization, dropout, flatten, and fully connected (dense) layers.

Plotted Accuracy v/s epochs



ResNet(transfer learning approach):

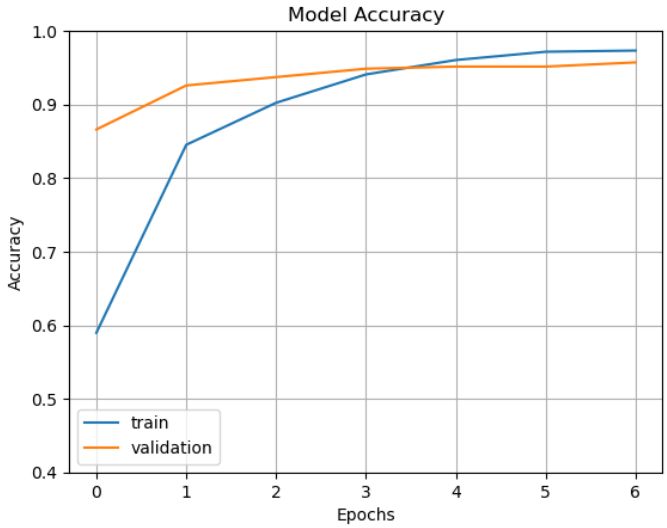
After trying all the standard approaches we tried to use the cutting edge image classification algorithm with would give us the best possible result.Hence we tried to implement resnet to test the capabilities.

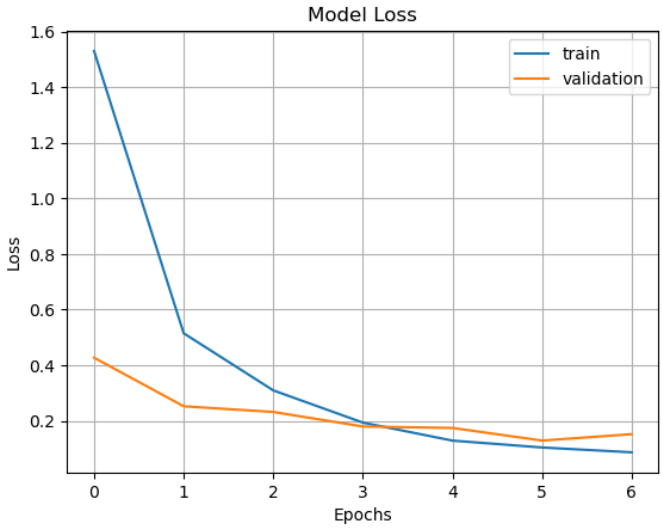
ResNet, short for "Residual Network," is a type of deep neural network architecture that was developed to solve the problem of vanishing gradients in very deep neural networks. The vanishing gradient problem occurs when the gradients become very small as they propagate through the network during the training process, which can cause the network to become untrainable.
It is trained on Image net which is a very huge data set and it borrows the weights it has learned from it and it fine tunes it to our model.

Image preprocessing is the same as above with normalization and size set to 256*256.

For the model we have used first layer
As 256*256*3 list.
Second layer is the pretrained model
Third layer is of 512 nodes activation relu
And third 36 layer which is output layer with activation as softmax

Optimizer id ADAM and loss function is “Saprse_catagorical_crossentropy”
And is trained over 7 epochs.
We have used 7 echos as the model world overfit over the data has reduced the accuracy hence we have used 7 epochs to make the model more general and to prevent overfitting.





Ideas we thought of implementing:

1.We thought of implementing 36 different CNN and each Cnn can only identify 1 fruit/vegetable and gives out the probability of whether it is or not and we would take an ensemble of the 36 trees and pick out the max probability .

2.We thought of downloading more image more image and increasing the data set to make our CNN more generalized.