**FOOD IMAGE CLASSIFICATION.**



Fig.Food and Non-Food Images

Food Image classification is the process of taking an input like “image”and output us in the form of class like (“food”,”non-food”) oraprobability of the particular class (there’s is a probability of 95% that it belongs to class food).

**Goal:**Building a machine learning model that can distinguish between **food** and **non-food**class using CNN for given an input of image.

**Brief overview of the project:**

This is the high level architecture of the project that am going to perform for this task shown below.

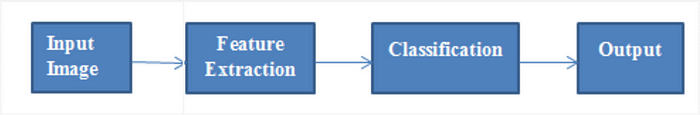


Fig.Steps involved in the classification.

So, in the process of first step to input an image and then extract the features by using the concept of CNN in deep learning and after these generated features set as input to the classification models like Logistic Regression(LR),Support Vector Machines(SVM) and these particular classifier predicts the desired output.

L**ets get started:**To solve this problem we some decent amount of image data that consists of “**food**” like dishes and “**non-food**” like animals,nature,insects,clothes,vehicles etc.

Here ,we are using a dataset contains nearly 5000 images of which 2500 images for **food** class and 2500 images for **non-food** class which organised them into two sub-folders .

Once ,after you have the dataset ready then ,the next step is convert these images into feature vector.

U**sing VGG-16 (**16-layer convolutional Neural Network) **to convert images into vectors.**

What is VGG-16 — (also called OxfordNet) is a Convolutional Neutral Network used for image classification. In 2010, ImageNet which has millions of labeled images (and it’s one of the largest high-quality datasets of images in the world currently) has hosted an annual challenge where teams of researchers present solutions to image classification and other tasks by training on the ImageNet dataset.

## But how to use it ?

For a normal computer, training a large neutral network with millions of images or data takes lot of computing power and is very expensive so better idea is to use a pre-trained model like **VGG-16** without having to master the skills necessary to tune and train those models which have been already trained with lots of images as we can directly use the weights and architecture obtained and apply the learning on our problem statement. This method is also known as **transfer learning**. We “transfer the learning” of the pre-trained model to our specific problem statement. So, loading the pre-trained models and using that model for prediction is relatively straightforward and described here.

def bottleneck\_features(BASE\_PATH,CLASS\_PATH):  
 #Function for Feature Extraction Using VGG16 img\_width,img\_height = 350,350  
 target\_size = (img\_width,img\_height)  
 PATH = BASE\_PATH+CLASS\_PATH  
 number\_samples = len(list(os.listdir(PATH))) print(“Found {} images for processing of {} class”.format(number\_samples,CLASS\_PATH.split(‘/’)[0]))  
 data = []  
 labels = []  
 for img in os.listdir(PATH):  
 #Storing label  
 labels.append(int(img.split(‘.’)[0].split(‘\_’)[0]))  
   
 temp\_img = image.load\_img(PATH+img,target\_size = target\_size) #Loading the image  
 temp\_img = image.img\_to\_array(temp\_img)   
 data.append(temp\_img)  
   
 data = np.array(data)  
 data = preprocess\_input(data)  
 model = applications.VGG16(include\_top=False,weights=’imagenet’)  
   
 bottleneck\_features = model.predict(data)   
 bottleneck\_features = bottleneck\_features.reshape(number\_samples,51200) print(“Saving Features and Labels of {} Class”.format(CLASS\_PATH.split(‘/’)[0])) np.save(open(CLASS\_PATH.split(‘/’)[0]+’\_features.npy’, ‘wb’), bottleneck\_features)  
 np.save(open(CLASS\_PATH.split(‘/’)[0]+’\_labels.npy’, ‘wb’), np.array(labels))

**Training and Testing**

Once after done with feature extraction,the next big important thing is that building the classification models.But before building the models, data must be splitted into train,cross-validation and test.

**The main reason for splitting the dataset** — to evaluate the performance of classifier .We are interested in how well the classifier generalizes its recognition capability to unseen data.

We feed the **train set**to model to learn the patterns of the data that present in it and get that ability to recognize the data that is present in the **test set.**

**Then what is the use of Cross-validation :**the main aim of the **cross-validation set** is to check how well the model recognizes the pattern on the unseen data in the training phase itself ,if it has any problems like “**overfitting**” and “**underfitting**”.

So,finally our dataset is divided into 64% into Train dataset,16% into Cross-validation dataset , and remaining 20% into Test dataset by using sklearn train\_test\_split() function.

#Spliting dataset into Train,Cross Validation,Testtrain\_df, X\_test, train\_y, y\_test = train\_test\_split(features\_data, labels, stratify=labels, test\_size=0.2)X\_train, X\_cv, y\_train, y\_cv = train\_test\_split(train\_df, train\_y, stratify=train\_y, test\_size=0.2)

**Building Machine Learning Model**

Before building any machine learning model ,we some business and performance constraints.

So,we will look some of them which are needed for this problem

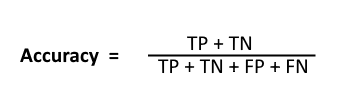
***Constraints:***

1. . Low latency requirement
2. Errors can be very costly
3. Probability of a data-point belonging to each class is needed

***Key Performance Indicators(KPI):***

1. Accuracy
2. Confusion Matrix
3. Log-loss

These **KPI’s**are used to check how our model performs to unseen data.



**Log-loss:**We know basically that the range of log-loss is in between [0,infinite].The main thinktank is to minimize the log-loss as much as possible to make the model better.A perfect model will have a log-loss of **0.**The log-loss value increases when the predicted output of class deviates from it’s original class.

**How to know our model good using Log-loss:**We predict the class label for every point in test set randomly ,there by calculate the log-loss and compare with the other classification models to check performance.

**Using Random Model :**

Log loss on Cross Validation Data using Random Model 0.8585008870605865  
Log loss on Test Data using Random Model 0.889529930069323

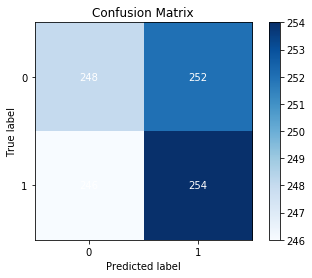


Fig.Random Model Confusion Matrix

Now,you can able to see that the log-loss of random model on the test dataset is 0.89. In order to get the best model then,log-loss of that model must be less than random model and in the range [0,0.89]

**Note:**As dimensionality of the features vectors are very large and also we have very strict latency constraints .So,we are we are going with linear models because these models are very fast to train the data.

* 1. **Using Logistic Regression :**

The train log loss is : 0.007921241973755228  
The cross validation log loss is : 0.028716149196689137  
The test log loss is : 0.029961496101542376

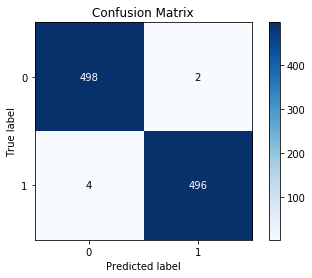


Fig. LR Confusion Matrix

**II. Using Support Vector Machines(SVM) :**

The train log loss is : 0.013700328169421657  
The cross validation log loss is : 0.036394944878339705  
The test log loss is : 0.027419064972506946

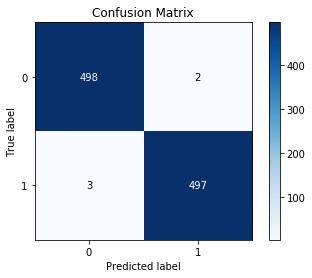


Fig. SVM Confusion Matrix

**Applying the model with real-world:**

As you can see the results of both the Logistic Regression and SVM models perform well.But we took LR model to test our real-time data.Here are the results below…



Food Image: 0.3414319141420361  
Non-Food Image: 0.6585680858579639  
I Guess it is Non-Food



Food Image: 0.0024575699560875452  
Non-Food Image: 0.9975424300439125  
I Guess it is Non-Food



Food Image: 0.09818049533607233  
Non-Food Image: 0.9018195046639277  
I Guess it is Non-Food



Food Image: 0.9299781343302631  
Non-Food Image: 0.07002186566973694  
I Guess it is Food



Food Image: 0.08400190657975289  
Non-Food Image: 0.9159980934202471  
I Guess it is Non-Food

You can observe that the last image output predicted by the model is wrong.May be the model couldn’t recognize because only a small portion of image is related to food and the remaining is different.

However,expect the last one all the remaining predicted outputs by the model are good.

Thats all for now for the project !. Still to improve the performance of the models we need to add more images and retrain again.

**Future Work:**

To develop an app that can distinguish between “food” and “non-food” images.(Food Classification)

**References:**

## [Food Image Dataset Permission is hereby granted, without written agreement and without license or royalty fees, to use, copy, modify, and…](https://mmspg.epfl.ch/downloads/food-image-datasets/?source=post_page-----8dc9611d0a84--------------------------------" \t "_blank)

[mmspg.epfl.ch](https://mmspg.epfl.ch/downloads/food-image-datasets/?source=post_page-----8dc9611d0a84--------------------------------" \t "_blank)