ARTIFICIAL INTELLIGENCE EXTERNSHIP

PROJECT REPORT

Project Title:
Recipe Recognition with Deep Learning

Team Members:
Hartej Singh, Rupin Patel, Aatman Prajapati, Sreya Venugopal

INTRODUCTION

Food is a fundamental part of our lives. It not only gives us energy but also characterizes our personality and culture. Ingredients, flavour, and culinary recipes form distinct cuisines are part of our personal and cultural identities, while eating habits have a direct influence on our health and wellness. Nowadays people are very enthusiastic about posting and sharing pictures of food on social media. There is no doubt that the value that food has in our society is unquestionable. But do you sometimes regret forgetting the name of the food that you ate long back or do you ever get a craving for that scrumptious cupcake you saw on Instagram but couldn't make since you didn't know what all went into it? The access to information on prepared foods is restricted, making it difficult to determine exactly what we eat or see.

The technique of recognising food items from an image is a fascinating field with a wide range of applications. Precise identification and segmentation of the various food items in a food tray may be used to track food consumption and nutritional information since people are more conscious about their food and diet to avoid diseases. In self-service restaurants, automated billing can be used with the help of a food recognizing system to eliminate the cashier bottleneck. Because of its growing importance in the health and medical fields, food image categorization is a new study topic.

In this project, we propose a food image recognition system based on deep learning algorithms to improve the accuracy and analyze each of the network architectures. This project outlines the many approaches that may be taken in order to create a deep learning solution for food recognition using various methods.

PROBLEM STATEMENT

One of the most promising applications of visual object recognition in computer vision is food image recognition and food monitoring is becoming increasingly important in our daily lives. The creation of an effective and more comfortable solution for getting the recipe has been made feasible by recent advancements in smartphone and web applications. However, most of these applications require manual input of food items. Despite the thousands of applications, methods, and systems available, the machine learning and computer vision groups have yet to properly handle the challenge of identifying food due to the lack of distinct and spatial arrangement, which is often present in images showing scenes or items. Also the Image recognition of food products is typically difficult due to the large diversity of varieties of food. Therefore recognizing the food and getting the recipe in one place can be a lifesaver. The goal of this project is to create a system to identify food from the images, solve the challenges of automatically recognising an image of food and generating the corresponding recipe.

SOLUTION

Deep learning has recently been demonstrated to be a highly strong image identification technology. The biggest distinguishing feature is that improved image features for identification are derived automatically via training. One of the approaches that meets the requirements of the deep learning approach is the convolutional neural network (CNN). CNN is used for food classification since it can process a huge quantity of data and estimate characteristics automatically. By using deep learning and various CNN methods a system that can recognize an image of food can be developed.

We are using a small dataset of 2418 images belonging to 3 classes and CNN layers for recognizing the images in our model. Convolutional neural networks, unlike traditional artificial neural networks, can estimate the score function directly from picture pixels. Data augmentation techniques based on geometric transformation were applied to increase the size of training images. We also use the Max-Pooling function for the data, and the features extracted from this function are used to train the network. A dataset of 600 images belonging to the 3 classes is used for testing the model. It is demonstrated that using data augmentation and by increasing the number of convolution layers, these networks exhibited much better performance, making these networks suitable for practical applications. The VGG-16 network improves the performance of automated food image categorization. The suggested VGG-16 has achieved a substantial improvement in accuracy of more than 90% due to increased network depth.

In this project, we use convolutional neural networks to detect the recipe in food and focus on applications of automatic food identification. And this model will categorise images of food into food groups and provide a recipe that corresponds to them.

FLOWCHARTS AND DIAGRAMS

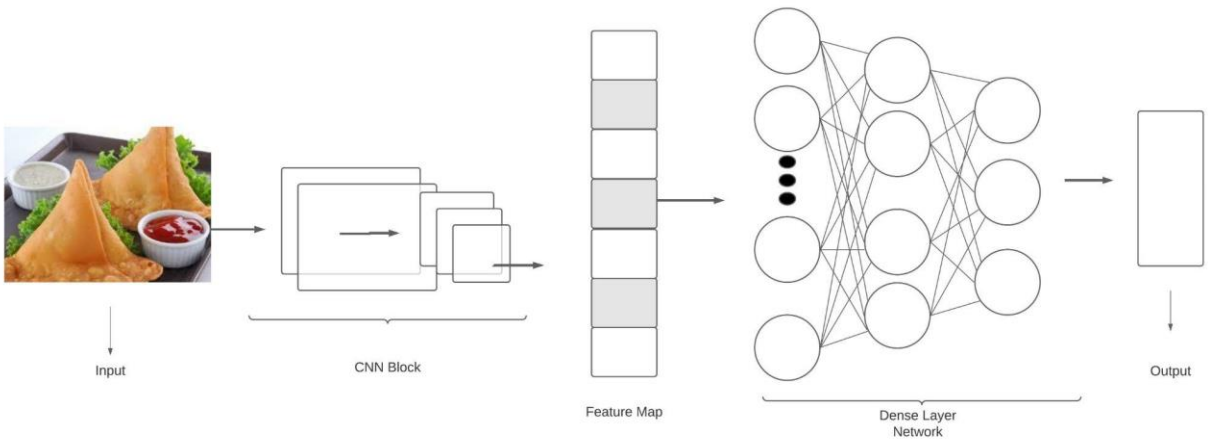


Figure 1.0

A typical CNN structure for image classification. A set of quadrilaterals stacked together represent the output after the calculation by convolutional layers. The circular symbols filled within this figure represent neurons in the Dense layer block, and the lines between circular symbols denote the weights and bias.

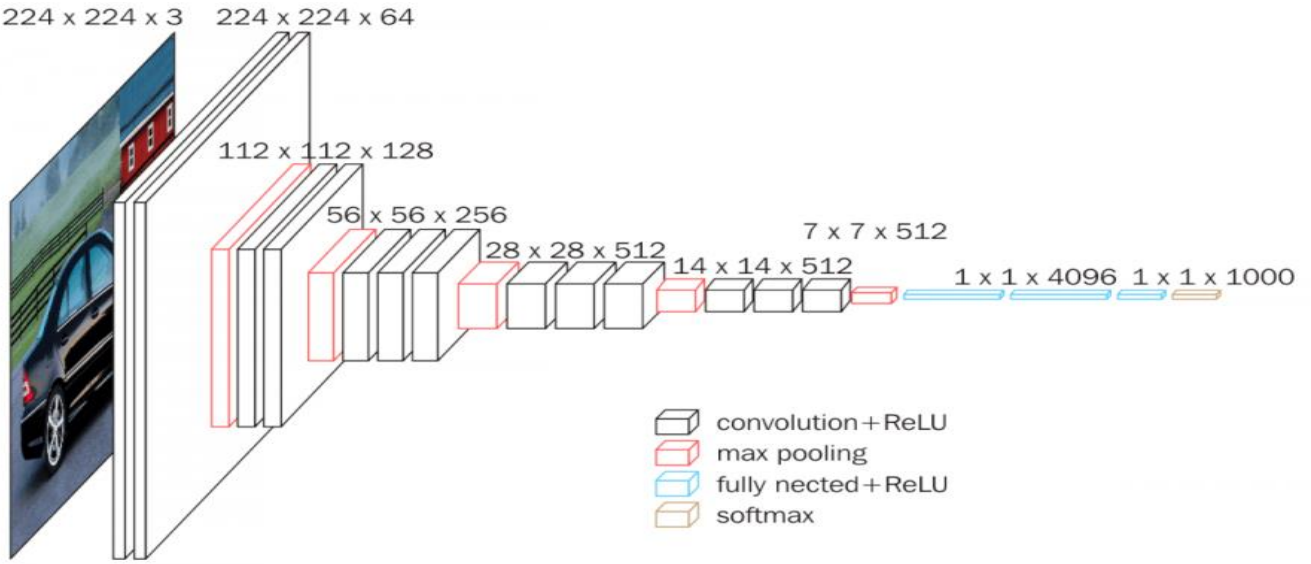


Figure 1.1
VGG Architecture

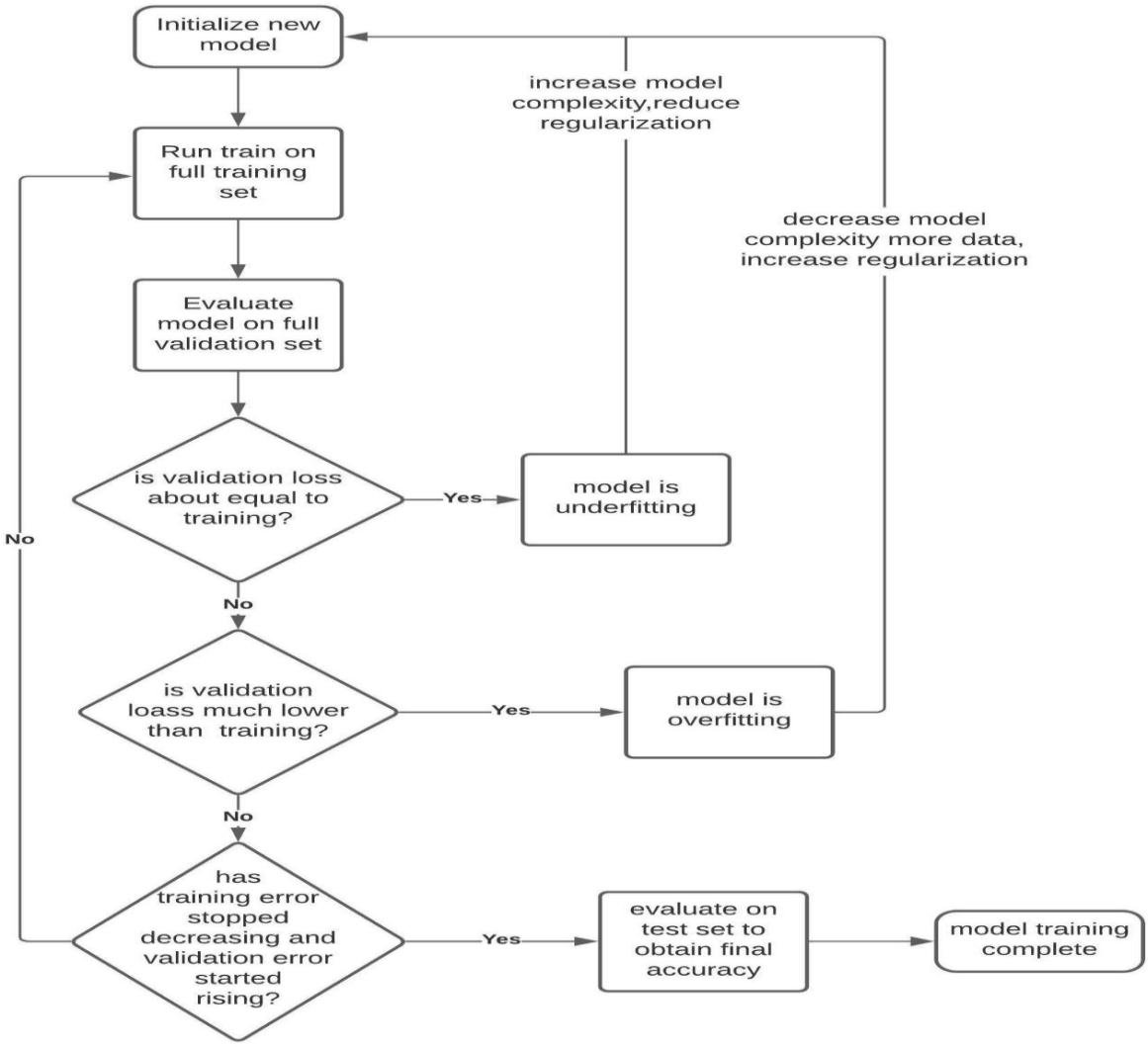


Figure 1.2
Training and Testing Mechanism

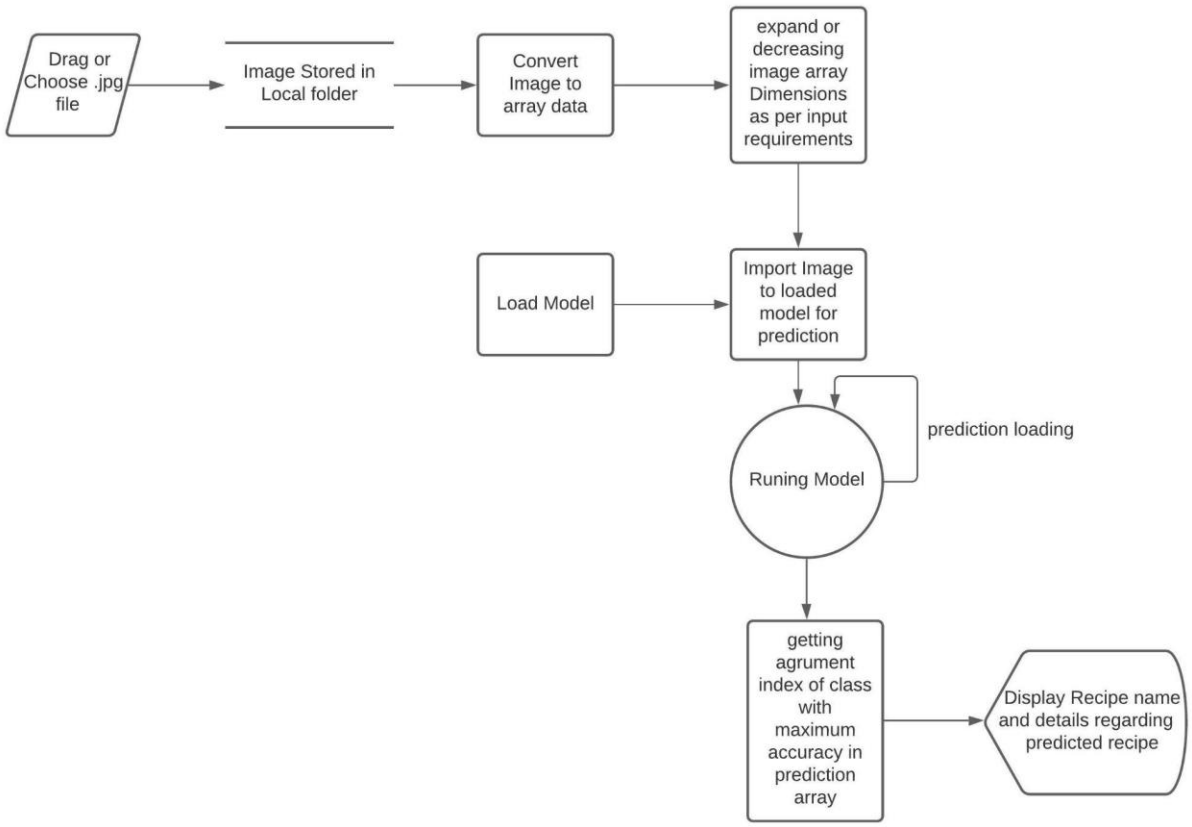


Figure 1.3
Data Flow

LITERATURE SURVEY

❖ Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images

By: Srikanth Tammina

Published in : International Journal of Scientific and Research Publications,
Volume 9, Issue 10, October 2019

Link: https://www.academia.edu/download/62094236/ijsrp-p942020200213-39389-1ytpyzm.pdf

The above mentioned research paper most appropriately defines the very problem with building Machine Learning models with a very isolated approach towards a particular dataset/specific feature space and same distribution.

Change of features and distribution is a very real possibility in real world situations - to expand the model or to re-frame it again for various purposes. In such cases it becomes an arduous process to collect related training data and rebuild the models.

To be able to transfer relevant functionality like - transfer of learning rate from disparate domains would be desirable. Transfer learning is a method of reusing a pre-trained model knowledge for another task. Transfer learning can be used for classification, regression and clustering problems.

VGG-16 accomplishes just this task with a deep convolutional network to classify images. It is a built-in library available in tensorflow.

❖ State Classification of Cooking Objects Using a VGG CNN

By: Kyle Mott

Link: https://arxiv.org/ftp/arxiv/papers/1904/1904.12613.pdf

This research paper conveys the importance of identifying & using various patterns to categorize food in various different classes based on its texture and object states.

In case of image classification using CNN there always is the possibility of the model **overfitting** - it fits too well with the training set that it becomes difficult to generalize to new examples outside of the training set. When working with a constraint on data and having applied the necessary image augmentations , we need to choose an architecture that generalizes effectively well to be used in real-life situations.

VGG with it's convolution , max pooling and dropout layers helps attain the level of classification required, as in case of food we just can't rely on pixel colours , density etc. alone , additional patterns which can come in the form of shapes and textures (and much more) should be taken into account.

Since this demands for more generalization , VGG seems to be an optimal choice and with a good optimizer we can get our accuracy in the 90+ range.

EXPERIMENTAL INVESTIGATIONS

- **First Attempt:-**

After the image augmentation , with the following model structure:

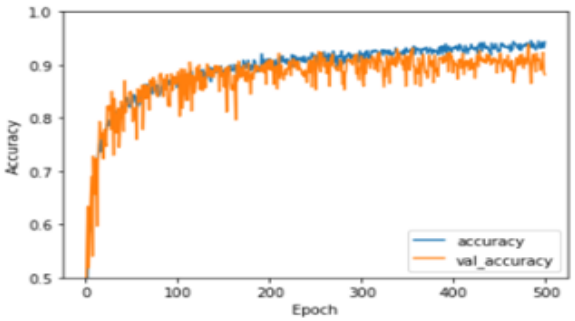
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 30752)             0
_________________________________________________________________
dense (Dense)                (None, 32)                984096
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 99
=================================================================
Total params: 986,147
Trainable params: 986,147
Non-trainable params: 0
_________________________________________________________________
```

After Using Adam optimizer & activation of Softmax in output layer at around 200 epochs:

- Accuracy - 94%
- Validation Accuracy - 88%

Graph:-

```
<matplotlib.legend.Legend at 0x1d02613a130>
```




```
test_loss, test_acc = model.evaluate(x_train)
```

```
76/76 [==============================] - 21s 276ms/step - loss: 0.1475 - accuracy: 0.9458
```

```
test_loss, test_acc = model.evaluate(x_test)
```

```
19/19 [==============================] - 4s 203ms/step - loss: 0.3898 - accuracy: 0.8817
```

Realizing that the model needs more validation accuracy, in the next attempt, we try to increase the number of convolution layers so as to make good use of the translational invariance through the model.

- **Second Attempt:-**

As we know that the more successive convolution layers we use , the easier it becomes for the model to pick up on certain traits/features/patterns in the images & the validation accuracy increases significantly.

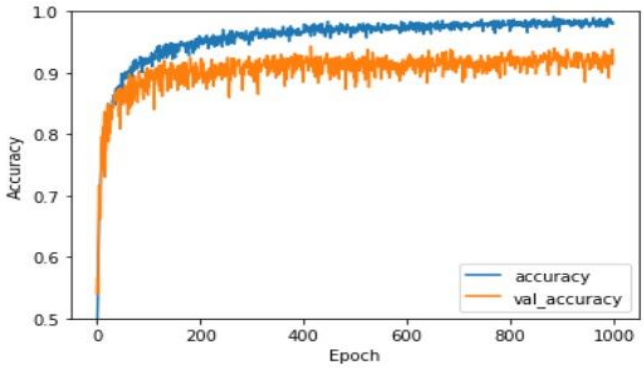
Model structure:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 6, 6, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 32)                36896
_________________________________________________________________
dense_1 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 195
=================================================================
Total params: 58,595
Trainable params: 58,595
Non-trainable params: 0
```

The result :

- Accuracy : 98% -> suggests some degree of overfitting in the model
- Validation Accuracy : 91.8 % -> meaning that more layers do help to increase the validation accuracy

Graph:-



Using this approach the model got trained very well compared to previous approach, but upon testing we found that the overfitting problem causes the predictions to be accurate in very specific types of images(the

ones which resemble closely to the training dataset). The model needs to be a bit more generalized and fine-tuned.

- **3rd Attempt:-**

Keeping in mind the advantages and problems faced by the previous models , we decided to go with the **VGG-16 model** as it solved the problem with over-fitting & further made the model better compared to the previous ones.

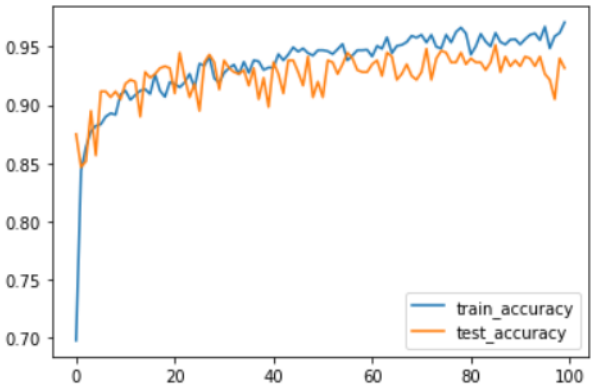
As this model is extremely slow to train (approx 90sec per epoch) we have done for 100 epochs and the model is still better than previous ones. (difference b/w acc & val_acc is least)
Model structure:-

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              25691136
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_4 (Dense)              (None, 3)                 3075
=================================================================
Total params: 40,408,899
Trainable params: 25,694,211
Non-trainable params: 14,714,688
```

The results we got were :

- Accuracy - 98.6%
- Validation Accuracy - 93.2%

Graph:-




```
In [27]: train_loss,train_acc=model.evaluate(x_train)
         76/76 [==============================] - 272s 4s/step - loss: 0.0359 - accuracy: 0.9859

In [28]: test_loss,test_acc=model.evaluate(x_test)
         19/19 [==============================] - 62s 3s/step - loss: 0.3437 - accuracy: 0.9317

In [ ]:
```

Hardware & Software Specifications:-

➔ Anaconda :-

Tool used for using standard libraries and packages for scientific computing.(Derived from Python and R Programming languages)

Jupyter Notebook :

- Used for Data pre-processing(in this case image pre-processing) , Model building and Model Testing.

➔ Pycharm IDE:

For developing the web-app(Flask) , with the appropriate directory organization, using Python programming language

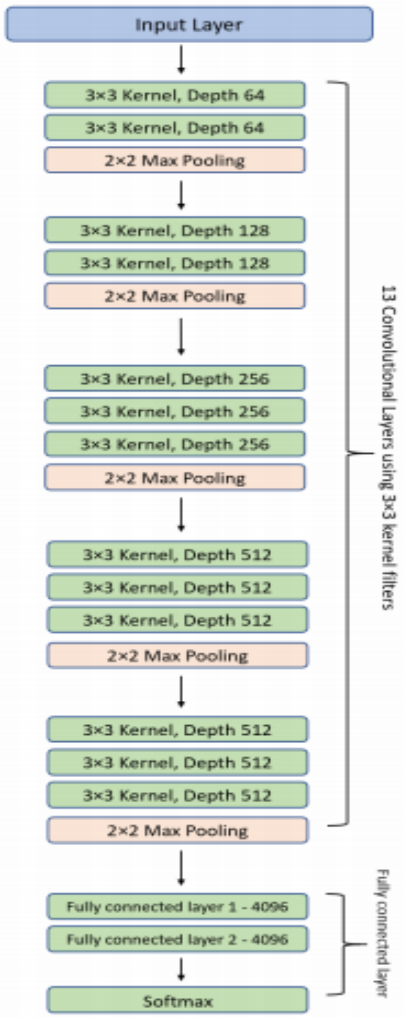
Keywords:-

- **VGG-16:-**

  VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”.

  The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014.

- **VGG-16 architecture:-**



As we can see, the vgg-16 model consists of extensive (DEEP) layers of Convolution coupled with Max-Pooling where ever required before going into the traditional set of hidden inter-connected (DENSE) layers.

➔ The input to the cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center).

- ➔ In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity).

- ➔ The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers.

- ➔ Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. Layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

Conclusion:-

In conclusion, we applied the knowledge of Convolution Neural Networks as its showed significantly higher accuracy than did traditional support-vector-machine-based methods to identify the input image and with an additional help of VGG library, we achieved a satisfactory accuracy,
Secondly, we deployed our trained model using Flask to a website. It contains a home page where the user uploads the photo and gets a brief result based on the prediction. The user can later see the detailed recipe in the following page.
Hence, our project will classify images into food categories and to output a matching recipe.

Future scope:-

We found that the convolution kernels show that color dominates the feature extraction process. For food image detection, CNN also showed significantly higher accuracy than a conventional method did.
Future work would involve more optimization on hyperparameters and model aspects such as which layers to freeze versus make trainable during transfer learning. Due to computing resource and time constraints, most model implementation decisions were made by examining the convergence of the model and relative metrics from training versus validation, but an exhaustive hyperparameter search would have been a more empirical approach.

Bibliography:-

https://recipes.timesofindia.com/recipes/
https://numpy.org/doc/stable/numpy-ref.pdf
https://pandas.pydata.org/docs/
https://www.tensorflow.org/api_docs
https://keras.io/guides/
https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3
https://flask.palletsprojects.com/en/2.0.x/