AI PROJECT REPORT

CS401 - SPRING 2020

Chatbot and Image recognition application for Pakistani dishes

GROUP MEMBERS

MUHAMMAD SAAD 17K3758

SATISH KUMAR 17K3756

MUHAMMAD JUNAID SHAIKH 17K3895

SHAN AMIN 17K3712

ABSTRACT

In our life there are many moments in which we have travelled different parts of the world moving from one place to another due to studies, job or any other reason most of us don't know how to prepare local Pakistani food or if some foreigners come to our country and want to explore the Pakistani cuisines they cannot remember the names of the dishes they tried or want to make them when they are home. The advancement in AI has helped in making the impossible possible. Our Chatbot interacts with the user and responds to the queries, such as finding recipes for dishes and gives suggestions regarding the dishes through its advanced AI and machine learning strategy. Also our Food AI application solves the second part of our problem through its Food Recognition Image Module. User uploads the image of his favourite Pakistani Cuisine and the AI application recognizes the dish. If our application is unable to recognize the dish, it is uploaded to the cloud server where the Data Analyst can enter it into the dataset for future use, this way the system keeps learning and growing.

INTRODUCTION

Food AI is designed to solve the problem of people, which did not know much about Pakistani dishes they in a hurry and they try to make local food by themselves in a few minutes and if they know about the dish, but forget their names so they easily find the name by using our application we build this application by keep in mind that particular audience.

In this modern era where website's , applications and blogs have a great impact on human's but somehow it takes time for searching any website for a recipe or watching 10-20mins video, where chatbots makes everything easier it will answer for any given Query within the seconds. This chatbot is specifically designed for getting the recipe of a dish, After all it's a simple way to receive step by step instructions to put together in your favorite meal. For chatbot we used an

advanced vector space model with some python learning libraries like nltk and we talked more about libraries and modules in the method section and the chatbot interaction with users and asked for a dish to find out its recipe and chatbot provided a recipe of the particular dish.

For image detection we used TensorFlow,Keras,IBM Watson virtual recognition model we trained some classes of local food dishes on that basis our model, classify them and when any image come with the help of opencv we resize them to optimize processing time as I mentioned we talked more about IBM Watson, opencv and other libraries in a method section.

The image is read in binary format and compared with our pre-trained classes and assigned scores for it which class has the highest score it labeled with that class and gives us the result.

If our image is not classified to any image so we just simply send it to our cloud storage from where we can manually create a class for that particular image and with the help of crowdsourcing data methodology our module is starting to be more accurate when more users are used to it.

RELATED WORK

For Chatbot module:

As we said, the importance of chatbot grows everyday and most of the businesses are implementing them in their business and there are much more advantages for them. So rather than using websites and videos in which it takes time to find recipes. So chatbots will give answers within seconds.

Some of related articles which are related to our work are as follow:

https://cargocollective.com/ronankellydesign/Chop-Chop-Bot

This Article is written by **Ronan Kelly**.

https://www.mobilemarketer.com/ex/mobilemarketer/cms/news/messaging/23226.html

This Article is written by **Thai Phi Le**

https://www.foodnetwork.com/site/apps/chatbot

The common thing of these chatbots with our project is that the idea used by them to provide the recipe of dishes within a few seconds like **chop chop** chatbot is a Facebook Messenger bot, and it was created specifically for people who are absolute novices when it comes to the wonders of the kitchen. It will provide the recipe of dishes with videos, a **whole food** chatbot is more generally useful for finding recipes and the **food network** is another chatbot, accessible through Facebook Messenger. You can use the bot to sift through thousands of recipes .

For image detection module:

As I said this is very unique work in the present scenario deep learning is still in the process of advancement. A new version of libraries comes every version has different capabilities and optimization for example there is huge difference in tensorflow version 1.0 and version 2.0 many new libraries come to support deep

learning models.The Idea of these project was design by our group members we took inspiration from final year batch of FAST under the supervision of Sir Rafi which also doing a project in food but there can predicts the ingredients from food images of particular food.

Some of related articles which are related to our work are as follow:

www.researchgate.net/publication/266357771_Food_Detection_and_Recognition_Using_Convolutional_Neural_Network

This Article written by **Kiyoharu Aizawa** Professor at University Of Tokyo in November,2014

developer.ibm.com/recipes/tutorials/machine-learning-and-ibm-watson-studio/

Machine Learning Guide In IBM Watson Studio

github.com/IBM/pytorch-on-watson-studio

Module of handwritten number detection on IBM studio

These above-mentioned links are not exactly the same as how our model works. We took ideas from above these links that how we initiate our models mostly people while developing deep learning models using tensorflow and keras or pytorch depend on their utility to train models. Tensorflow v 2.0 comes with support for keras which help us to build deep learning models in a more spohisacted way.IBM watson studio also a great tool for Deep learning API. we make our own model in tensorflow and keras and then we integrated tat model with IBM API which very suitable and provide great support to AI Desktop Application

DATASET AND FEATURES

For Chatbot Module:

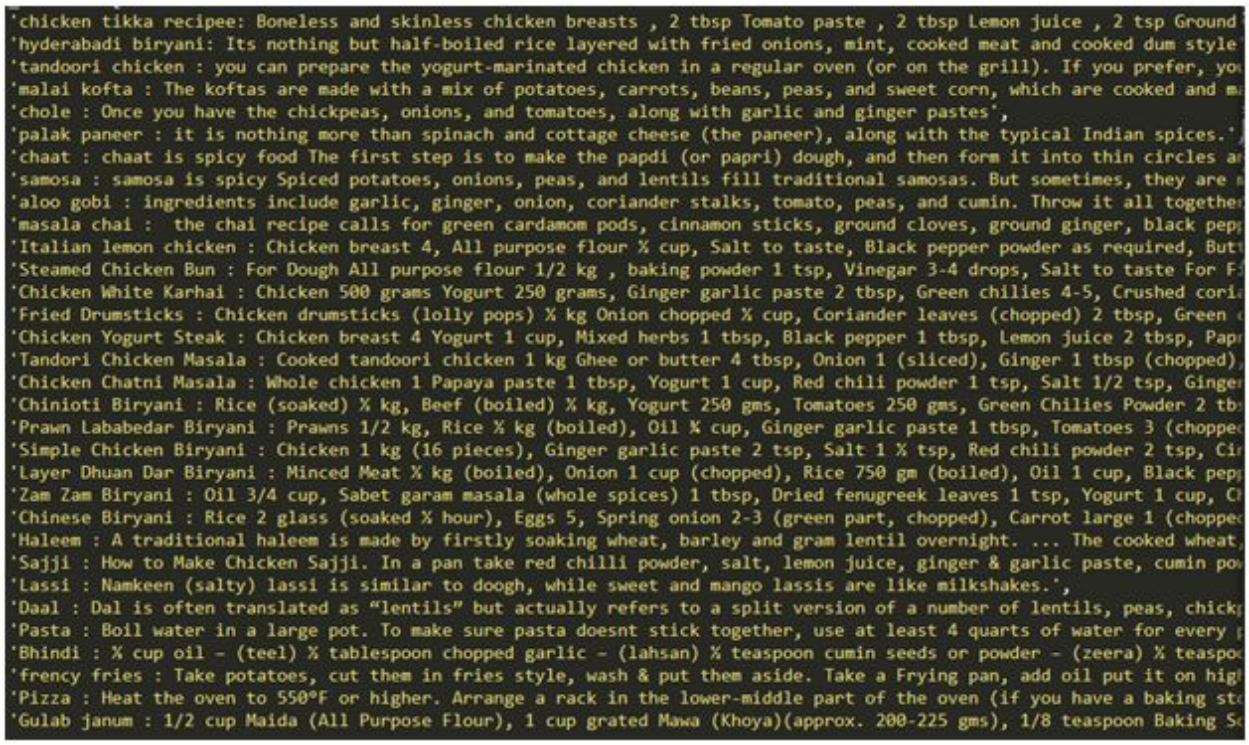
The dataset will contain the 32 recipes of dishes in the training set and 16 Queries in the test set which are used to identify how efficiently chatbot is working.

Recipes of dishes in dataset = [**chicken tikka, hyderabadi biryani, tandoori chicken, malai kofta, chole, palak paneer, chaat, samosa, aloo gobi, masala chai, Italian lemon chicken, steamed chicken bun, chicken karahi, fried drumsticks, chicken yogurt steak, tandori chicken masala, chicken chatni masala, Prawn Lababdar Biryani, Simple Chicken Biryani, Layer Dhuan Dar Biryani, Zam Zam Biryani, Chinese Biryani, Egg, maggi, Haleem, Sajji, Lassi, Daal, Pasta, Bhindi, french fries, Pizza, Gulab janum**]

Source of the these pakistani recipe's is available in this link:

https://madeeasy.com.pk/?utm_source=Google&utm_medium=Search&utm_campaign=MadeEasy%20-%20PK%20Search%20-%202019&gclid=EAIaIQobChMIu5av2o6i6QIVWpnVCh0aHwY8EAAYASAAEgL6v_D_BwE

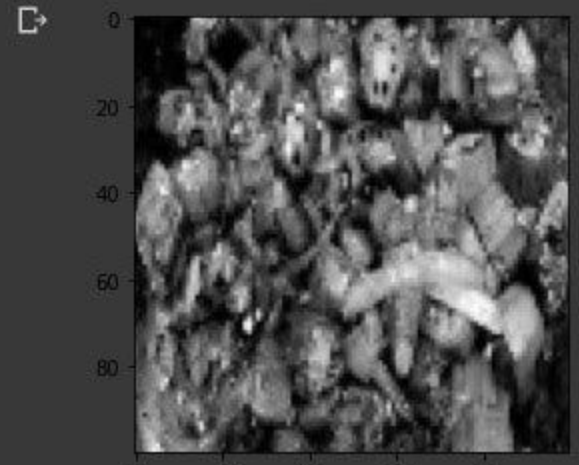
View of dataset



For Image Detection Module:

We did around more than 52 training /validation/test combinations to achieve accuracy with different numbers of layers we discussed about different combinations in the experiment section. It's generally a good idea to "normalize" your data. This typically involves scaling the data to be between 0 and 1, or maybe -1 and positive 1. In our case, each "pixel" is a feature, and each feature currently ranges from 0 to 255.we used image resolution of 100 but we allowed in our application that it should be at least 24 resolution size image. We used grayscale pixel values as features.The simplest way to create features from an image is to use these raw pixel values as separate features.

```
[16] IMG_SIZE=100
     new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
     plt.imshow(new_array, cmap='gray')
     plt.show()
```



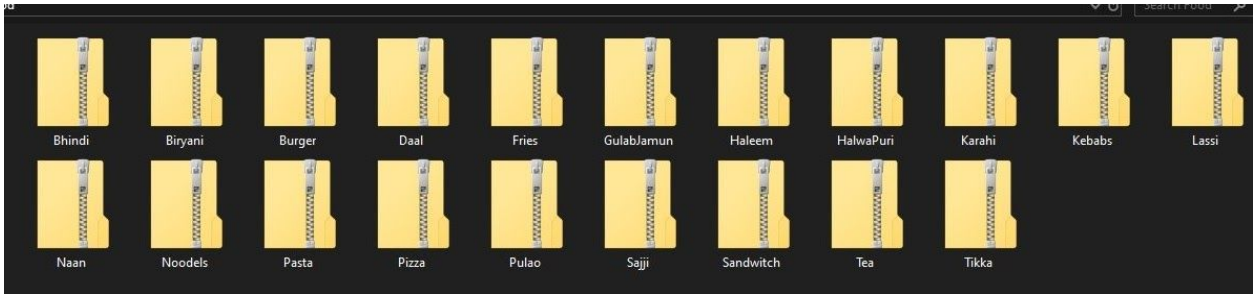
We have around 250 images dataset and around 30 images test dataset.We classify 20 local dishes in our model name as:

['Bhindi', 'Biryani', 'Burger', 'Daal', 'Fries', 'GulabJamun', 'Haleem', 'HalwaPuri', 'Karahi', 'Kebabs', 'Lassi', 'Naan', 'Noodles', 'Pasta', 'Pizza', 'Pulao', 'Sajji', 'Sandwich', 'Tea', 'Tikka']

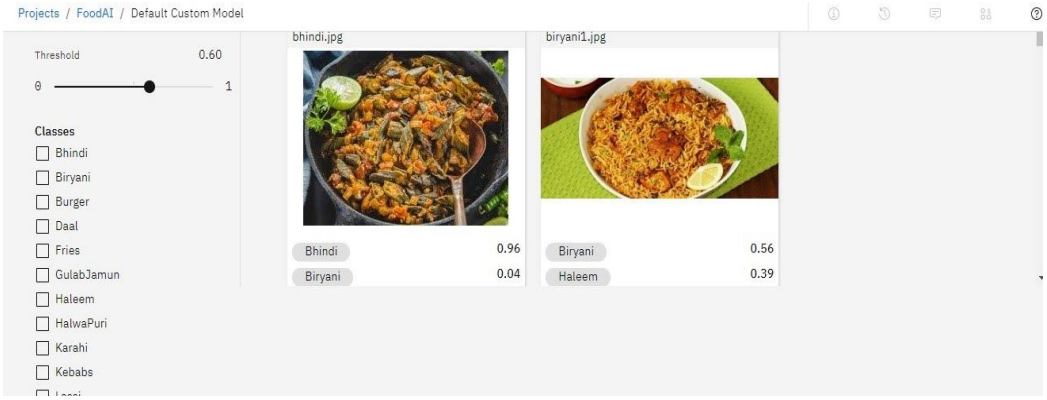
There is not image data set available for Pakistani dishes so we create our own dataset y collecting images from Google Images To access data for these model go to link below:

kaggle.com/muhammadsaad21/foodpaistandishes

Testdata



Validation Data



Test

METHODS

METHOD FOR CHATBOT MODULE:

For chatbot module we use vector space model with using the libraries of sklearn which provide us a way to perform TfidfVectorizer and cosine similarity in our project and beside this we also use the nltk libraries for preprocessing ,lemmatization, punctuation remove etc.

Working of vector space model:

Vector space model. This use case is widely used in information retrieval systems and machine learning. Given a set of documents/dataset and search term(s)/query we need to retrieve relevant documents/information that are similar to the search query.

It is an algebraic model, involving two steps, in the first step we represent the text documents into a vector of words and in the second step we transform to numerical format so that we can apply any text mining techniques.

Suppose we have two document's/text:

Text1: Egg : 2 minutes recipe 2 large EGGS, 1/4 tsp salt , 1/4 cup Gruyere cheese

Text 2: maggi : Boil 1 1/2 cups water in a pan and add Maggi Noodles and Tastemaker. Simmer for two minutes till the Maggi is fully cooked.

Query: maggi

vectors representation:

This step includes breaking each document into words, applying preprocessing steps such as removing stopwords, punctuations, special characters etc. After preprocessing the documents we represent them as vectors of words.

Below is a sample representation of the document vectors.

Text 1: (Egg, 2, minutes, recipe, 2, large, EGGS, 1/4tsp, salt, 1/4cup, Gruyere, cheese)

Text 2: (maggi, Boil, 1, 1/2cups, water, in, a, pan, and, add, Maggi, Noodles, and, Tastemaker, Simmer, for, two, minutes, till, the, Maggi, is, fully, cooked)

Query: (maggi)

the relevant Text to Query = greater of (similarity score between (Text1, Query), similarity score between (Text2, Query)

Next step is to represent the above created vectors of terms in numerical format known as term document matrix.

Term Document Matrix:

A term document matrix is a way of representing documents/Text vectors in a matrix format in which each row represents term vectors across all the Texts/documents and columns represent document/Text vectors across all the terms. The cell values frequency counts of each term in corresponding document/Text. If a term is present in a document/text, then the corresponding cell value contains 1 else if the term is not present in the document/text then the cell value contains 0.

After creating the term document matrix, we will calculate term weights for all the terms in the matrix across all the documents/Text. It is also important to calculate the term weightings because we need to find out terms which uniquely define a document/Text.

We should note that a word which occurs in most of the Text/documents might not contribute to represent the Text/document relevance whereas less frequently occurring terms might define Text/document relevance. This can be achieved using a method known as term frequency - inverse document frequency (tf-idf), which gives higher weights to the terms which occurs more in a Text/document but rarely occurs in all other Text/documents, lower weights to the terms which commonly occurs within and across all the Text/documents.

$$idf_j = log\left[\frac{n}{df_j}\right]$$

Inverse document frequency

TFIDF

For a term i in document j:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

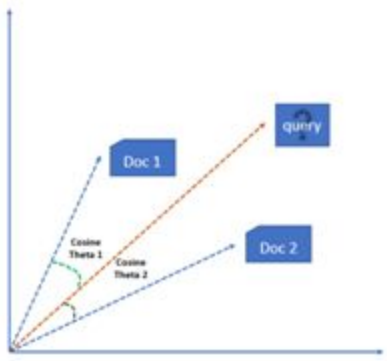
tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Term frequency * Inverse document frequency

Similarity Measures: cosine similarity

Mathematically, closeness between two vectors is calculated by calculating the cosine angle between two vectors. In similar lines, we can calculate cosine angle between each Text/document vector and the query vector to find its closeness. To find relevant Text/document to the query term , we may calculate the similarity score between each Text/document vector and the query term vector by applying cosine similarity . Finally, whichever Text/documents having high similarity scores will be considered as relevant Text/documents to the query term.

When we plot the term document matrix, each Text/document vector represents a point in the vector space. In the below example query, Text 1 and Text 2 represent 3 points in the vector space. We can now compare the query with each of the Text/document by calculating the cosine angle between them.



This graph shows doc2 is more similar with Query.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

NLTK:

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).

It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.

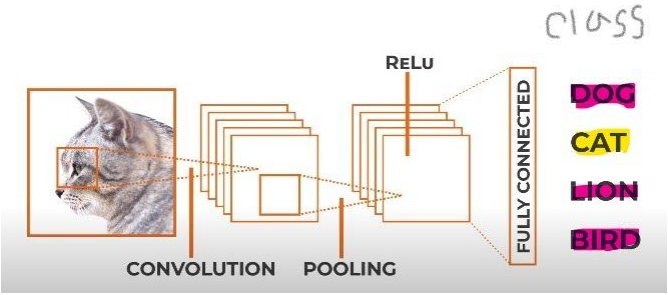
METHOD FOR IMAGE DETECTION MODULE:

For Image detection we used conventional neural network aka CNN it's network, which widely used in AI field, but before going to CNN that how we implement first talk about the effectiveness of CNN that how CNN better than other neural

network in these Deep learning worlds

A regular network has an input layer hidden layer and output layer hidden layer where all mathematical calculation happens and output these calculations come out of as an Output each layer has a neuron and each neuron has some weight they are connected with their previous layer.

But when we deal with image classification we need to build a network with more spatial networks. CNN solved this problem, it simplified the image by filtering it instead of a traditional network each network connected to all other layers. In CNN only similar or most closed datasets are connected to each other. So, In CNN we only focused on Spatial dataset In our CNN we used four layer convolution layer, pooling layer, relu layer and activation we talked about each layer in detail and how they layer are used in our model



So, first we talked about convolution layers which work as a filter on an array of pixels when images are coming as input we convert into an array of pixels and then place a filter over it to look at their specific features. It's like you look, an image from a small window to evaluate its feature in our case we set this window to 3x3 dimension and in our case, let instance the one filter is the color of food or may it's shape which create a featured map of 3x3

Now come to the pooling layer it makes the process much faster because it reduces the number of parameters from the feature map as the result it produces pooled feature maps which produce from one of these two methods Average pooling feature map, Maximum pooling feature map. In our network we used a maximum pooling feature map

In Relu(Rectified Linear Unit Layer) and Sigmoid which works as an activation

we talked about the activation function later on. The final layer is the fully connected layer used for classification in our network.

The basic CNN structure is as follows:

Convolution -> Pooling -> Convolution -> Pooling -> Fully Connected Layer -> Output

Now as I mentioned earlier we talked about activation and optimizer function later on so it's time to talk about that but before we start this discussion we just review that Convolution is the act of taking the original data, and creating feature maps from it. Pooling is down-sampling, most often in the form of "max-pooling," where we select a region, and then take the maximum value in that region, and that becomes the new value for the entire region. Fully Connected Layers are typical neural networks, where all nodes are "fully connected." The convolutional layers are not fully connected like a traditional neural network.

We used a sequential deep learning model. A sequential model is what you're going to use most of the time. It just means things are going to go in direct order. A feed forward model.

We used two hidden layers in our model for better accuracy so for two hidden layers we used two different activation functions Relu,Sigmoid Functions.Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

ReLu (**Rectified Linear Unit Layer**)Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural networks.**Equation** ***A(x)*** = ***max(0,x)***. It gives an output x if x is positive and 0 otherwise.**Value Range** [0, inf) **Nature** non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.**Uses** ReLu is less computationally expensive than tanh and

sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

Sigmoid Function It is a function which is plotted as **'S'** shaped graph.**Equation** A = 1/(1+ e-x) **Nature** Non-linear. Notice that X values lie between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y. **Value Range** 0 to 1 **Uses** Usually used in the output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.

The reason behind using two activation functions is accuracy. In the first hidden layer we used **Relu** function which activates only relevant class then in the second hidden layer we used **Sigmoid** function which helps to detect the most relevant class for the image because it's only chosen frIt's easy to get lost in the complexity of some of these new optimizers. Just remember that they all have the same goal: minimizing our loss function. Even the most complex ways of doing that are simple at their core.

om 0 to 1.

For error detection we use binary_loss entropy because by using these losses our model goes to 87 percent accuracy because we used sigmoid activation function which works on binary classification so to detect most relevant loss we used binary_loss .Also called Sigmoid Cross-Entropy loss. It is a Sigmoid activation plus a Cross-Entropy loss

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Equation of calculating binary loss

Now come to the optimization part for optimization we used **Adam** Optimizer. Adam stands for adaptive moment estimation, and is another way of using past gradients to calculate current gradients. Adam also utilizes the concept of momentum by adding fractions of previous gradients to the current one. This optimizer has become pretty widespread, and is practically accepted for use in training neural nets.The authors describe

Adam has combined the advantages of two other extensions of stochastic gradient descent(Stochastic gradient descent is an iterative method for optimizing an objective function with suitable smoothness properties.) Specifically:

- **Adaptive Gradient Algorithm** (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).

- **Root Mean Square Propagation** (RMSProp) that also maintains

per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).

Adam realizes the benefits of both AdaGrad and RMSProp.

EXPERIMENTALS/RESULT/DISCUSSION

CHATBOT MODULE EXPERIMENTALS/RESULT/DISCUSSION:

We have 32 recipe of dishes in our chatbot which we train using vector space model And 25 queries in the testing dataset which we are using to evaluate our model, by finding the confusion matrix, F1 score, precision, recall, and accuracy these things help us to identify how much our chatbot was efficiently working.

To identify how much our model was efficiently working first we write the actual results of testing data in the list y_true then write the predictive results in another list y_pred.

Accuracy.

accuracy is a valid measure of model performance. Which indicates how many predictions are correct.

```
In [4]: import numpy as np
        import pandas as pd
        import seaborn as sn
        from sklearn.metrics import accuracy_score
        from sklearn import metrics
        y_pred = [3,3,3,1,2,0,3,1,2,0,3,3,3,0,0,2,0,3,1,0,0,2,3,0]
        y_true = [3,0,3,1,2,0,2,1,2,0,3,3,3,0,0,2,0,0,1,0,0,2,3,3]
        accuracy_score(y_true, y_pred)

Out[4]: 0.8333333333333334
```

Accuracy of the chatbot is 83.333%

Recall.

recall is an extremely important model evaluation metric. recall refers to the percentage of total relevant results correctly classified by your algorithm.

```
In [49]: metrics.recall_score(y_true, y_pred, average='micro')
Out[49]: 0.8333333333333334
```

Recall of model is 83.33%

Precision.

Precision is also an extremely important model evaluation metric. precision refers to the percentage of your results which are relevant classified by your algorithm.

```
In [48]: metrics.precision_score(y_true, y_pred, average='macro')
Out[48]: 0.8854166666666666
```

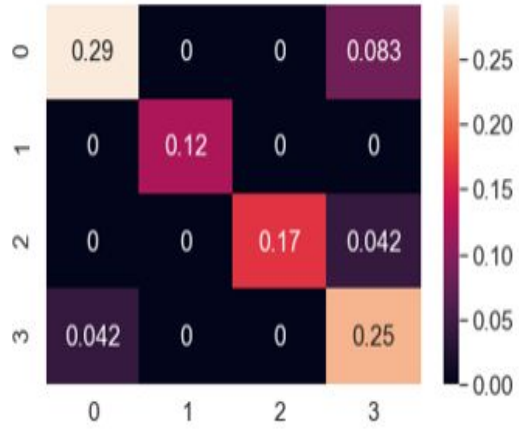
Precision of model is 88.54%

Normalization Confusion Matrix.

A confusion matrix is a performance measurement technique for Machine learning classification. It is a kind of table which helps you to know the performance of the classification model on a set of test data for which the true values are known.

```
In [9]: var2 = confusion_matrix(y_true, y_pred, normalize='all')
        df_cm = pd.DataFrame(var2, range(4), range(4))
        sn.set(font_scale=1.4)
        sn.heatmap(df_cm, annot=True, annot_kws={"size": 16})
```

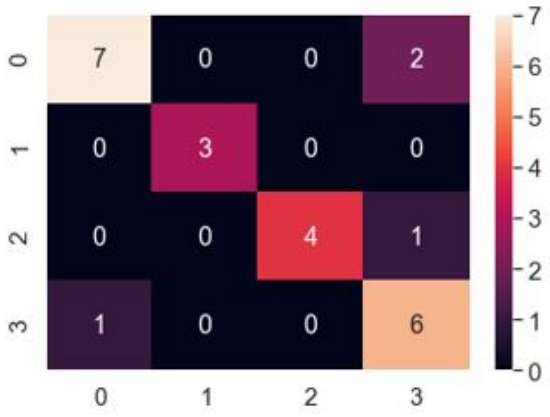
```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x17e3e0988d0>
```



Without normalization Confusion Matrix.

```
In [8]: from sklearn.metrics import confusion_matrix
        var = confusion_matrix(y_true, y_pred)
        df_cm = pd.DataFrame(var, range(4), range(4))
        sn.set(font_scale=1.4)
        sn.heatmap(df_cm, annot=True, annot_kws={"size": 16})
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x17e3bf6f198>
```



F1 Measure.

The F measure (F1 score or F score) is a measure of a test's accuracy and is defined as the weighted harmonic mean of the precision and recall of the test.

F1 score of model is 83.77%

IMAGE MODULE EXPERIMENTALS/RESULT/DISCUSSION:

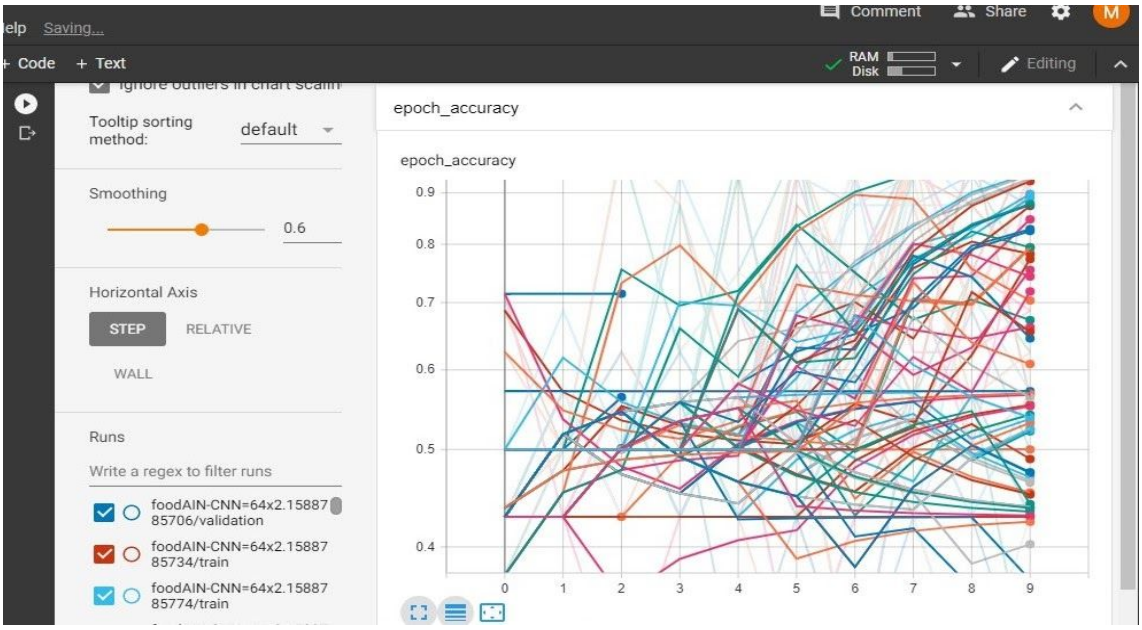
We have around 250 images dataset and around 30 images test dataset we used two hidden layer one with Relu activation and other with sigmoid our approach is that whenever image came it convert into array of pixel then it gets in to CNN network in relu layer they only connected or associated with two most near relevan or similar class in sigmoid layer we chose from that two classes as we perform binary classification to produce more outcome result. We build our Model In watson studio to get better accuracy with help of tensorflow and keras support our model accuracy is around 87.5 percent our batch size is 20 because we used **MiniBatch Gradient Descent** for stability of the network we used many different layers combination to produce more accurate result

dense_layers = [0, 1, 2] , layer_sizes = [32, 64, 128], conv_layers = [1, 2, 3]

It all together make 54 new combination we try all 54 new combination

We classify 20 local dishes in our model name as:

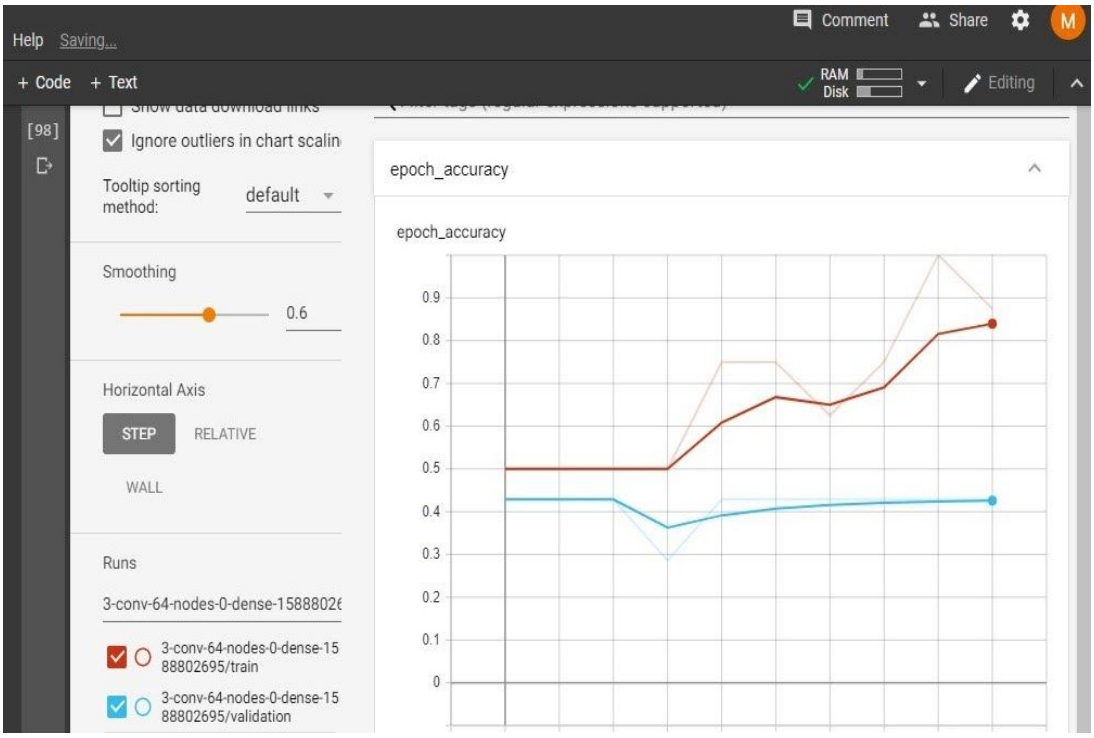
['Bhindi', 'Biryani', 'Burger', 'Daal', 'Fries', 'GulabJamun', 'Haleem', 'HalwaPuri', 'Karahi', 'Kebabs', 'Lassi', 'Naan', 'Noodles', 'Pasta', 'Pizza', 'Pulao', 'Sajji', 'Sandwich', 'Tea', 'Tikka']



All Combination Accuracy

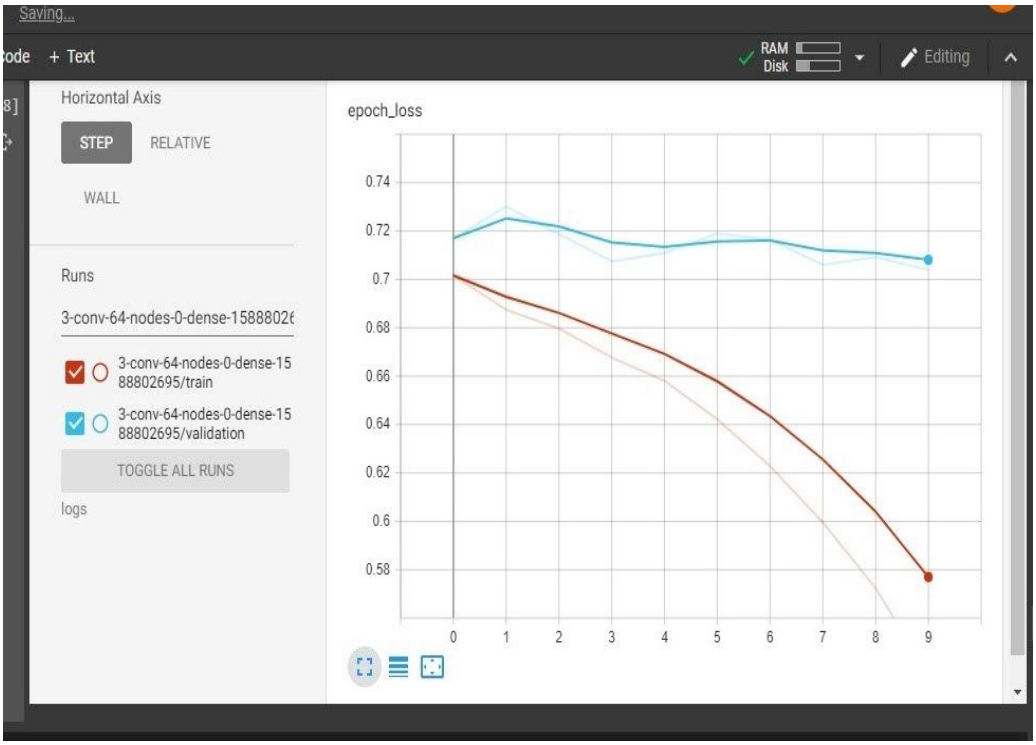
All Combination Loss

We get most accurate result at 0x64x3 where we can achieve 87.05 accuracy

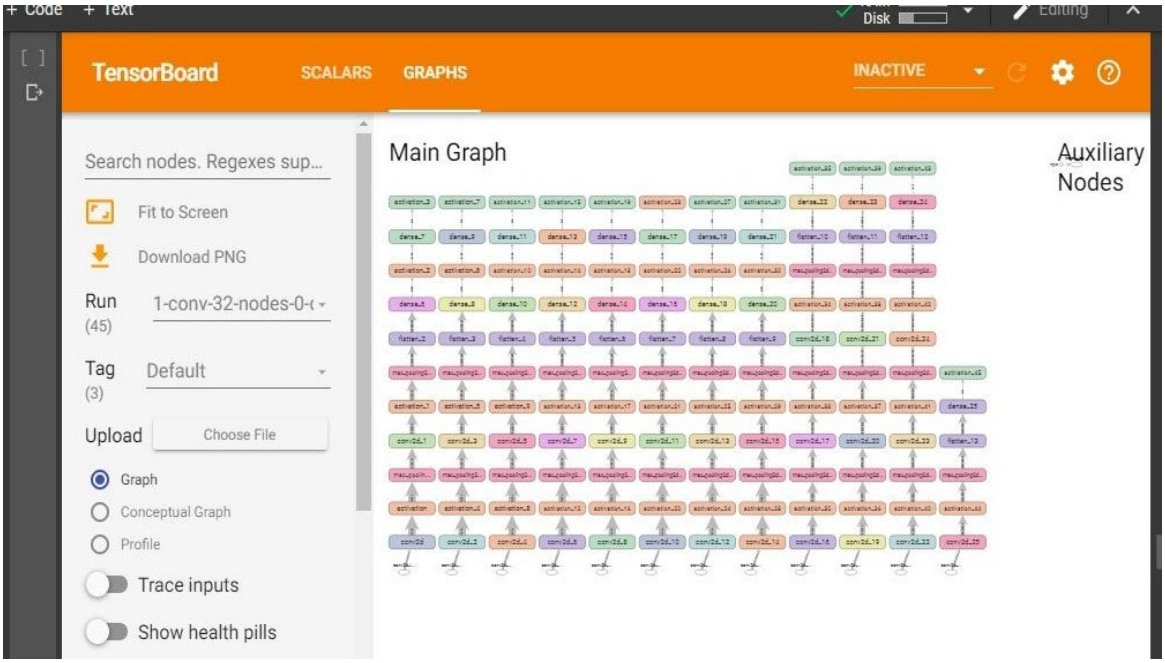


Model Accuracy

As you seen from graph our model is overfit because our accuracy is greater then validation accuracy for mitigate these we try different combination and activation functions



Model Loss



2D CNN Layer Graph

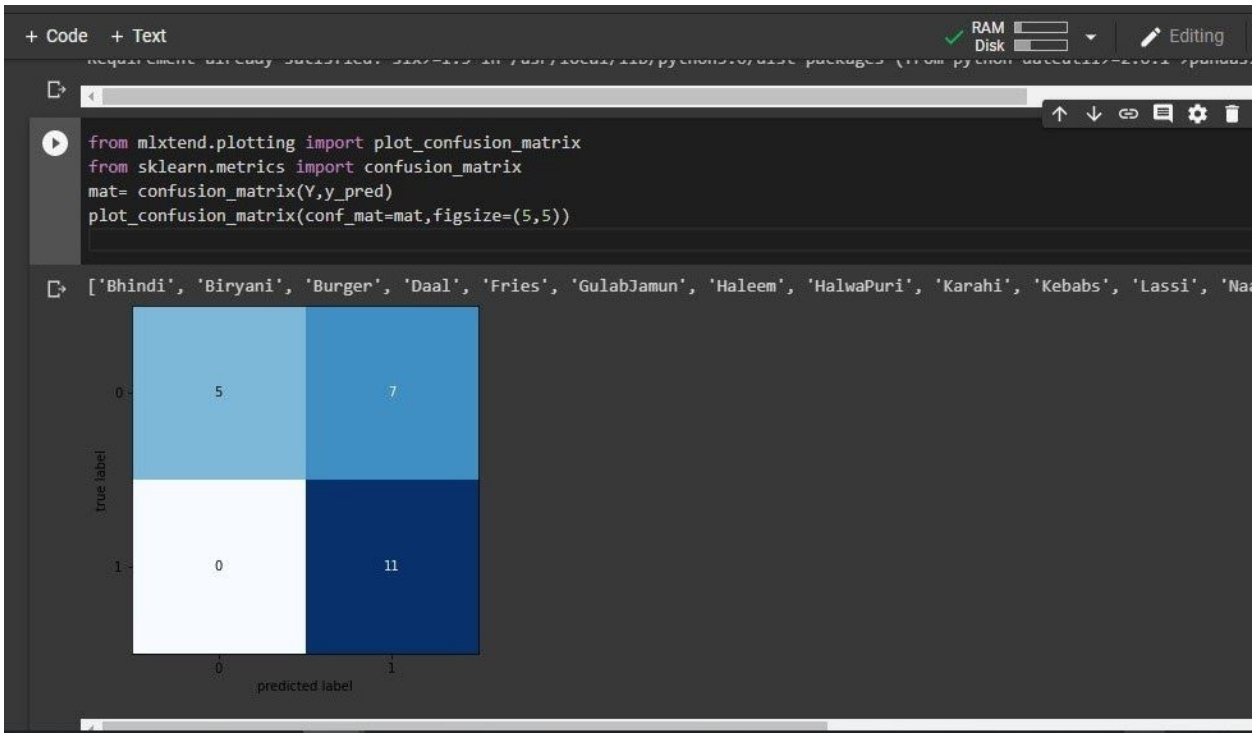
```
from sklearn.metrics import classification_report
print(classification_report(Y,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.42      0.59        12
           1       0.61      1.00      0.76        11

    accuracy                           0.70        23
   macro avg       0.81      0.71      0.67        23
weighted avg       0.81      0.70      0.67        23
```

Precision , Recall , f1 score



Confusion Matrix

We have made 20 classes of food in confusion matrix we only get two classes because after the Relu Layer we used Sigmoid activation function which classify our image in to two classes Actually relu layer comes out the two most relevant classes and these classes further selected by sigmoid model in hidden layer

Our Image detection accuracy is still not good to overcome these issue we integrated our model with IBM watson Studio Machine Learning Model which increased our model accuracy upto 95%(It’s a claim of IBM watson we did not

check IBM accuracy so we still believe that our model is accuracy is 87.5 %) the main reason of poor accuracy is the lack of data as we only trained 250 images with 20 classes(each class has at least 10 images). We are dealing with local pakistani dishes so we create our own data set by collecting images from Google Images.

CONCLUSION / FUTURE WORK

The rise and popularity of chatbots increase day by day is clearly outlined in the figure. With this in mind the data gathered from testing the chatbot justifies the recent growth and demand for companies wanting to integrate a chatbot. It was determined that chatbots perform at a very high standard and provide reliable and rapid responses to users compared to that of traditional methods and learn after each conversation/interaction with the users. The results also show that the image detection module also provides nearly accurate results and is gaining popularity due to its use cases which can be implemented in many different environments.

In future more advanced algorithms will emerge with better accuracy and performance so we will try to optimize our model and integrate those features in it. We plan to shift these applications from desktop based to mobile devices and web as well for convenience .We will try to make our chatbot more user friendly and more interesting and effective outside our domain and try to expand our chatbot by covering more variety of food region wise. Implement location based food detection AI models which also tells some description about food quality, price , reviews etc

CONTRIBUTION

We believe teamwork is the most important aspect in these projects as Hellen Keller said that “Alone we can do so little; together we can do so much ”. We divide ur work so we can work at productively and effectively

Following are the list of work done by the member of the groups along with their names and Id’s

Muhammd Saad 17K3758

Build a deep learning model for image detection

Satish Kuamr 17K3756

Build Chatbot for Food AI

Muhammad Juanid Shaikh 17K3895

Integrate the deep learning model with IBM watson Studio

Shan Amin 17K3712

Making GUI of the Application

REFERENCES / BIBLIOGRAPHY

https://www.techopedia.com/definition/30343/natural-language-toolkit-nltk

https://scikit-learn.org/stable/

Now come to Libraries and integrations which we used in our **Image detection module.**

www.researchgate.net/publication/266357771_Food_Detection_and_Recognition_Using_Convolutional_Neural_Network

This Article written by **Kiyoharu Aizawa** Professor at University Of Tokyo in November,2014

developer.ibm.com/recipes/tutorials/machine-learning-and-ibm-watson-studio/

Machine Learning Guide In IBM Watson Studio

github.com/IBM/pytorch-on-watson-studio

Module of handwritten number detection on IBM studio

www.ncbi.nlm.nih.gov/US/pmc/articles/PMC5537777/

This Article published from National Center for Biotechnology Information (NCBI) US website it's about predicting diet plane by looking the food images tell about their calories

Now come to Libraries and integrations which we used in our **Image detection module.**

TensorFlow is an open source Deep Learning library developed by Google that is used to perform complex numerical operations and several other tasks to model Deep Learning models. Its architecture allows easy deployment of computations across multiple platforms like CPU's, GPU's, etc.

Keras is a neural networks library written in Python that is high-level in nature – which makes it extremely simple and intuitive to use. It works as a wrapper to

low-level libraries like TensorFlow or Theano high-level neural network library, written in Python that works as a wrapper to TensorFlow or Theano.

The IBM Watson Visual Recognition service uses deep learning algorithms to analyze images for scenes, objects, and other content. IBM Watson Studio provides a collaborative environment in the cloud where you can work with your images and your Visual Recognition custom models.

We Used IBM visual Recognition model to achieve more accuracy in our model. So, first we make model on tensorflow and keras and then we integrated our model to IBM visual recognition to achieve more processing and accuracy we talked more about accuracy in experiment section

OpenCV is a library of Python bindings designed to solve computer vision problems. OpenCV makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays.We used OpenCV for image resizing and converting images into numpy arrays for computation

Firebase used as cloud services if our image detect that particular image so it's sent to the our firebase cloud from where data analyst manually can put these images to our classification depend on image content.So whenever next time user used **Food AI** it find more optimal or accurates result that's how we keep improving our mode accuracy by crowdsourcing methodology

Tkinter used as a GUI of the whole application

LANGUAGES / PLATFORM

We mainly used python version 3.6 and 3.7 in our application and json for data handling

Python 3.6 recommended for machine and deep learning model

Google Colab and Pycharm used as a platform in these application

Google Colab recommended for machine and deep learning model

VIDEO / GITHUB LINK

1. Complete Project Demonstration:

   www.youtube.com/FOODAIN

2. Image Dataset:

   https://www.kaggle.com/muhammadsaad21/foodpaistandishes

3. Complete SourceCode Of FOODAIN

   https://github.com/MuhammadSaad20/FoodAIN

   (Note: For Security reasons we hide all API keys from the code in github if you want to access those keys or you want run this application in you PC contact to our group)