Food Recognition using CNN

Team Members:
19BCE0938 (Kasa Yeshwant)
19BDS0064(Sandeep Kaki)
19BDS0032(Vishnu Sai)

Report Submitted for the First Project Review of

Course Code: CSE3047
Deep Learning

Slot: F2
Professor: Dr. W.B. Vasantha

Introduction:

Health is the most important asset any living being possesses. As they say, “Health is wealth”. Recognizing food from images is an extremely useful tool for a variety of use cases. In particular, it would allow people to track their food intake by simply taking a picture of what they consume. Current nutritional monitoring apps use traditional methods for food recognition, with algorithms based on classification. However, such applications are either semi-automatic, which identifies a group of possible types of food and requires user interaction. Image-based food recognition has in the past few years made substantial progress thanks to advances in deep learning. But food recognition remains a difficult problem for a variety of reasons.

Problem:

Description:

The goal of this project is to train models which can look at images of food items and detect, in the form of image segmentation, the individual food element. The categories of food to recognize are limited to a given set of categories decided in the function of the dataset used. Once the neural networks used for the food recognition are trained, predictions on a test set of images are made, and based on the results are computed using some metrics.

Dataset:

The dataset used for this project is the dataset provided in the Food Recognition challenge proposed by AICrowd [1], in particular, the dataset is divided into the training set, Val set, and test set.

The dataset is structured as follow:

• Training Set of 24120 (as RGB images) food images with their corresponding 39328 annotations in MS-COCO format.

• Validation Set of 1269 (as RGB images) food images with their corresponding 2053 annotations in MS-COCO format.

• Test Set where there is provided the same images as the validation set.

Motivation:

Though technology has advanced a lot in recent years, not much effort or consideration has gone into developing technology that is convenient and provides a productive and effective solution to monitoring and maintaining our health. The first step is to develop technology capable of monitoring health. The most common way that exists now is for the user to manually input the food and the number of calories they intake in order to track the food they intake. But this is neither a product nor an efficient way for the user to keep track as it is tedious.

Literature Review Summary Table:

Authors and Year (Reference)	***Title (Study)***	***Concept / Theoretical model/ Framework***	***Methodology used/ Implementation***	***Dataset details/ Analysis***	***Relevant Finding***	***Limitations/ Future Research/ Gaps identified***
Salim, Nareen OM, et al. "Study for Food	Study for Food Recognition System Using	Demonstrating that deep learning overcomes other	A prediction algorithm for deep learning, also referred to	Mangosteen detection, Food 101,	G. Ciocca, G. Micali, and P. Napoletano, "State recognition of food	(p l a c i

Recognition System Using Deep Learning." Journal of Physics: Conference Series. Vol. 1963. No. 1. IOP Publishing, 2021.	Deep Learning	strategies like manual feature extractors, standard ML algorithms, as well as DL as a practical tool for food hygiene and safety inspections. Food attributes such as type, structure, nutrients, and process types (natural products and refined food) are concerned with balanced	as a model, consisting of several layers which are shown below. The data as input passes through the layers during deep learning inference and every layer presents multiplications of matrices on data.	Indian foods, Food-11, food.	images using deep features," IEEE Access, vol. 8, pp. 32003-32017, 2020	ng food on white dishes and taking food pictu

		diet issues. Knowing the characteristics of foods (type, composition, nutrients, process types, etc.). Deep Learning for fish, palm oil production, processing of (Vegetables, Fruits, food processing, efficiency assessments) - 91.09% accuracy achieved.				res for food quantity calculation, with a chec

						k e r b o a r d . (b) Image recognition processes are performed on servers in all the systems, preventing operations from occurring interactive because of Delays in Contact.
Fakhrou, Abdulnaser, Jayakanth Kunhoth, and Somaya Al Maadeed. "Smartphone-based food recognition system	Smartphone-based food recognition system using multiple deep CNN models	Developing a new deep convolutional neural network (CNN) model for food recognit	We arousing a deep neural network	Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei	Kong et al. [4] proposed a mobile phone-based dietary assessment system named DietCam. The DietCam	The majority of them contain food dishes only but not varieties of fruits or vegetables.

using multiple deep CNN models." Multimedia Tools and Applications 80.21 (2021): 33011-33032		ion using the fusion of two CNN architectures(LeNet, AlexNet). The deep CNN model is developed using the ensemble learning approach which was trained for a customized food recognition dataset. Achieved accuracy - 95.55%		L (2009) Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition. Florida,	utilized SIFT feature descriptor algorithm and nearest neighbor algorithm to recognize the food dishes. Matsuda et al. Matsuda et al. [5] proposed a multiple-food recognition system that uses various image feature descriptors such as SIFT, CSIFT, HOG, Gabor texture, and color feature **Limitat**	

				the USA pp 248–255		

Subhi, Mohammed Ahmed, Sawal Hamid Ali, and Mohammed Abulameer Mohammed. "Vision-based approaches for automatic food recognition and dietary assessment: A survey." IEEE Access 7 (2019): 35370-35381.	Vision-Based Approaches for Automatic Food Recognition and Dietary Assessment	The built-in mechanism of deep learning algorithms adopts the features extraction automatically through a series of connected layers followed by a fully connected layer which is responsible for the final classification. It has recently become popular owing to its marginally exceptional performance with enhanced processing abilities, large	In this they use 3 deep learning models They are Resnet,vcg, Inception	C. Güngör, F. Baltacı, A. Erdem and E. Erdem, "Turkish cuisine: A benchmark dataset with Turkish meals for food recognition", *Proc. 25th Signal Process. Commun. Appl. Conf. (SIU)*, pp. 1-4, May 2017.	H. Hassannejad, G. Matrella, P. Ciampolini, I. De Munari, M. Mordonini and S. Cagnoni, "Food image recognition using very deep convolutional networks", *Proc. 2nd Int. Workshop Multimedia Assist. Dietary Manage.*, pp. 41-49, 2016	(a) The smartphone has to be placed on the tabletop with a flat surface to calibrate the cameras. This approach achieved an average estimation absolute error of 16.65% for ten types of food. (b) Segmentation is still challenging when prepared, occluded, or mixed food items are considered.

		datasets, and outstanding classification ability compared to other traditional methods.				
Hussain, Ghulam, et al. "A CNN based automated activity and food recognition using the wearable sensor for preventive health care." Electronics	A CNN Based Automated Activity and Food Recognition Using Wearable Sensor for Preventive Healthcare	The piezoelectric sensor generates different signal patterns for six different food types as the ingestion of each food is different from the others owing to their different characteristics: hardness, crunchiness, and tackiness.	using Event Similarity search algorithm with CNN archite	A novel algorithm called event similarity search (ESS) is developed to choose a segment with dynamic length, which represents signal patterns with different comple	Physical activities, Dietary Behaviour, Preliminary of CNN	(a) Particular spaces equipped with cameras, which restrict the subjects' movements affecting the brightness conditions degrading the accuracy of such systems. (b) Audio sensing fails to classify soft food types due to background noise

8.12 (2019): 1425			ct ur e.	xities equally well. For the effective representation of the signal patterns of the activities and foods, employed dynamic segmentation. Accuracy - 91.3(CNN), 94.3%(SVM).			
Shen, Zhidong, et al. "Machine learning-based approach on food recognition and	Machine Learning-based approach for food recognition and Nutrient estimation	Presenting a novel system based on machine learning that automatically performs	Designing the prototype system based on the client-server model, the client		The American Medical Association Annual Meeting (June	The dataset they chose has very limited parameters like camera angles, different backgrounds, lighting	

nutrition estimation ." Procedia Computer Science 174 (2020): 448-453.		accurate classification of food images and estimates food attributes. Proposing a deep learning model consisting of a CNN that classifies food into specific categories in training the prototype system.	sends an image detection request and processes it on the server-side . Totally it is designed with three main software components including a pre-trained CNN model training module for classification purposes, a text data training module for attribute estimation models, server-side module.		18, “No Title.” [Online].)	conditions, etc, moreover it needs a lot of training algorithms in Recognizing and detecting various foods, Enhancement of systems and Datasets, Calories Awareness, and Nutrition aware
Giovany, Stanley, et al.	Indonesian Food	The challenging part of	Designing the prototype	Top-1 Testing Accurac	Hinton,	Lack of a diverse and quality data

"Indonesian Food Image Recognition Using Convolutional Neural Network." *Computer Science On-line Conference.* Springer, Cham, 2019.	Image Recognition Using Convolutional Neural Network	food image recognition is to recognize a food image with different backgrounds, intensities, and perspectives. Convolutional Neural Network (CNN) seems to be the right choice to build a powerful model that is able to recognize food images accurately. Current researches in food recognition use American fast food and	system based on the client-server model, the client sends an image detection request and processes it on the server-side. Totally it is designed with three main software components including a pre-trained CNN model training module for classification purposes, a text data training module for attribute estimation models, server-side module.	y 76.3% for standard network, 95.2% for the inception-v4 network	G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580	set causes a decrease in the accuracy. The usage of a traditional CNN limits the performance of the inception-v4 architecture.

		Japanese food as the dataset. The different dataset has different treatment to obtain a good result. Color and presentation have important roles as features.			(2012)	
Lu, Y., Stathopoulou, T., Vasiloglou, M. F., Christodoulidis, S., Blum, B., Walser, T., ... & Mougiakakou, S. G. (2019, July). An artificial intelligence-based system for nutrient intake	An Artificial Intelligence-Based System for Nutrient Intake Assessment of Hospitalised Patients	The nutritionists in every hospital take care of the admitted patient on their diet and food intake but unfortunately the time required and amount of Nutritionists required to serve	An AI-based, complete automatic monitoring system for nutrient intake by hospitalized patients, by analyzing the RGB-D image pairs captured before and after the meal.	Top-1 testing accuracy 84%	Imoberdorf, R.; Rühlin, M.; Beerli, A.; Ballmer, P.E. Mangelernährung Unterernährung. Swiss Med. Forum 2011, 11, 782–786	The main problem with this AI-based Nutrient Monitoring system is the system estimates almost the nutrient intake with an MRE of almost 15%

assessment of hospitalized patients. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 5696-5699). IEEE		patients in big hospitals are insufficient, So this completely AI human-less monitoring machine shares/completely take in charge of serving patients in large numbers efficiently.				
M o h d N o r h i s h a m R a	This research under takes an empirical evaluation of recent transfer	This research under takes an empirical evaluation of recent transfer	the model is implemented into a web application system as an attempt toautomate food recognition In this model, a fully connected layer with 11 and 10	The VIREO-Food172 Dataset and a newly established Sabah Food Dataset are used to evaluate the food recognit	From the evaluation, the research found that the Efficient Net feature extractor-based and CNN classifie	some accuracy performance drawback

z a l i 1 , Ervin Gubin Moung 2,* , Farashazi llah Yahya 2 , Chong Joon Hou 2 , Rozita Hanapi 1 , Raihani Mohamed 3 and Ibrahim Abakr Targio Hashem	learni ng mode ls for deep learni ng feature extraction for a food recognitio n model.	learni ng mode ls for deep learni ng feature extraction for a food recognitio n model.	Softmax neurons is used as the classifier for food categories in both datasets.	ion model.	r achieve d the highest classificati on accuracy of 94.01% on the Sabah Food Dataset and 86.57% on VIREO-F ood172 Dataset.	
Chang Liu1, Yu Cao1, Yan Luo1, Guanling Chen1, Vinod Vokkaran e1, Yunsheng Ma2	Deep Learning-based Food Image Recogniti on for Computer -aided Dietary Assessme nt	goal of this research is to develop computer-aided technical solutions to enhance and improve the	However, how to derive the food informatio n (e.g., food type and portion size) from food image effectively and efficiently			There is no limitations mentioned in the research paper.

		accuracy of current measurements of dietary intake.	remains a challenging and open research problem. So they proposed a new Convolutional Neural Network (CNN)-based food image recognition algorithm to address this problem.			
Hokuto kagaya,kiyoharu Aizawa, Makota Ogawa	Food Detection and Recognition Using Convolutional Neural Network	deep learning has been shown recently to be a very powerful image recognition technique, and CNN is a state-of-the-art approach to deep learning. They applied	In this paper, they compare the optimization of some of these parameters. In our research, we use cuda-convnet1, which is a GPU implementation of a CNN in C++ and Python, for the CNN	They made use of the data produced by Food-Log (FL). The general public can use FL for their own food recording using both photos	Inthispaper, the have addressed the effectiveness of CNNs for food image recog	There is no limitations mentioned in the research paper

		CNN to the tasks of food detection and recognitio n through parameter optimizati on	library.	and text.	nitio n and detec tion.t hey foun d that CNN perfo rmed much bette r than did tradit ional meth ods using hand craft ed featu res.	

Objective:

We propose away, where our project can detect the food from pictures or videos. This method proves to be less tedious than manually entering the food and encourages the users to monitor and maintain their health.

Novelties / Innovation component in the Project:

➔ Accurately recognizing different types of food.

- ➔ Extracting the nutritional value of each food.
- ➔ Both video and image recognition.
- ➔ Differentiating different types of liquids like red wine and glass of water.
- ➔ Encouraging users to monitor their food intake and health.

Approach / Methodologies Adapted:

- **Preprocessing:** The first preprocessing operation that has been done is the filtering of the categories that the model can detect. It has been necessary to do this since the dataset provides lots of categories and images that, if totally used for training a neural network, would require high computation capability. So, it has been agreed to reduce the number of categories down to 16 and in the same way include only those images of the dataset where these classes are present. Following this idea, for the training, it has been created a subset of the entire dataset. The categories that have been taken into account are the most present in the dataset considering the entire distribution of categories is mentioned as follows: Water, bread-white, salad-leaf-salad-green, tomato, butter, carrot, coffee-with-caffeine, rice, egg, mixed-vegetables, wine-red, apple, jam,potatoes-steamed, banana, cheese.

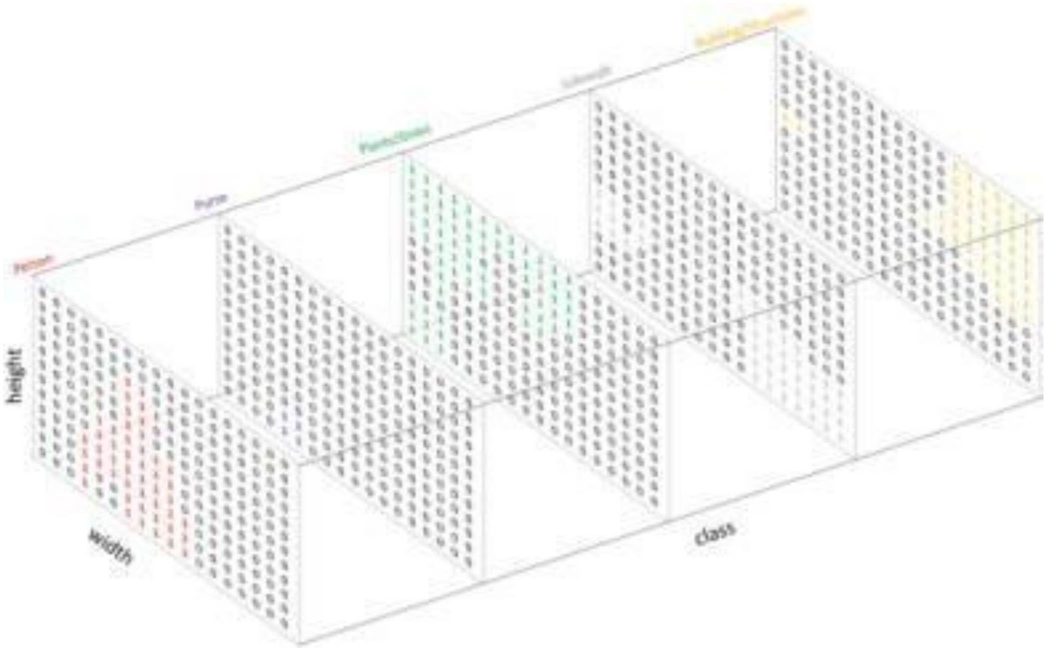
  These first 16 classes are the selected ones. Including the background class, the total number of categories has become 17. The dataset has been reduced as follows:

  <u>Train set:</u> from 24120 images to 10819 (and annotations from 39328 annotations to 12869).

  <u>Val set:</u> from 1269 images to 554.

  <u>Test set:</u> from 1269 images to 554.

- **Multichannel Masks:** One of the most important choices to do was to decide how to encode the classes present in the images into the corresponding masks. In particular how to represent the different categories that are inside a single image. The selected technique, in order to deal with multiclass images, is the use of the so-called multichannel masks associated with every image. This technique consists in creating an output channel for each of the possible classes, where every single channel represents the target by one-hot encoding the class labels. In order to apply this method, it has been used Numpy matrices put in the last dimension of the encoding of every channel.



The above image indicates multichannel masks with one-hot encoded channels for every category.

- **Data Generator:** Even if the dataset used for the training is a subset of the original, it is still too big to be loaded into memory at once and you could run out of RAM. That is the reason why there was the need to find other ways to do that task efficiently. A good solution has been found in the use of data generators, that can read images on the fly when they are used for training. Keras allows the creation of custom data generators inheriting the properties of *keras.utils.Sequence* in order to leverage nice functionalities such as multiprocessing. Every single Sequence implements the methods:

• __getitem__, that should return one batch of data (X,y) • __len__, that should return the number of batches per epoch • on_epoch_end, which is triggered once at the very beginning as well as at the end of each epoch.

It allows to shuffle the dataset to make some randomness at each epoch Each image and mask is processed by the DataGenerator using pycocotools API [2] to convert annotations in multichannel masks and then they are resized to 128x128. So the shape of the input and output data in each batch becomes:

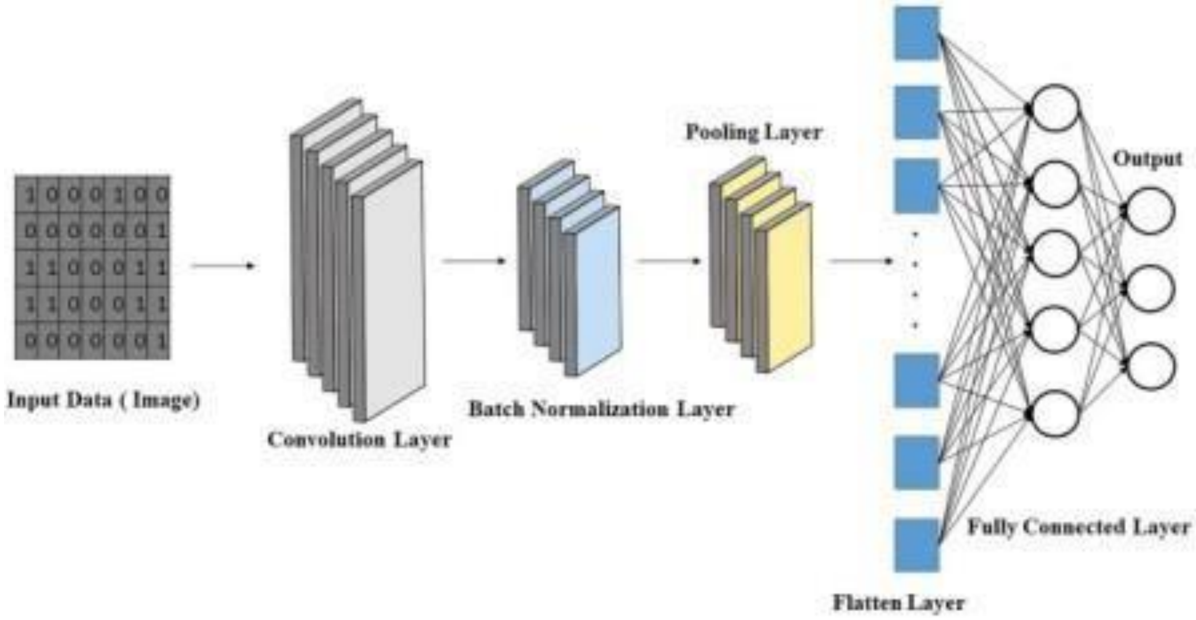
- X.shape = [batch_size, 128, 128, 3]
- y.shape = [batch_size, 128, 128, 17]

The channel 0 on the output represents the background and it is composed of all 1s where the category mask is absent.

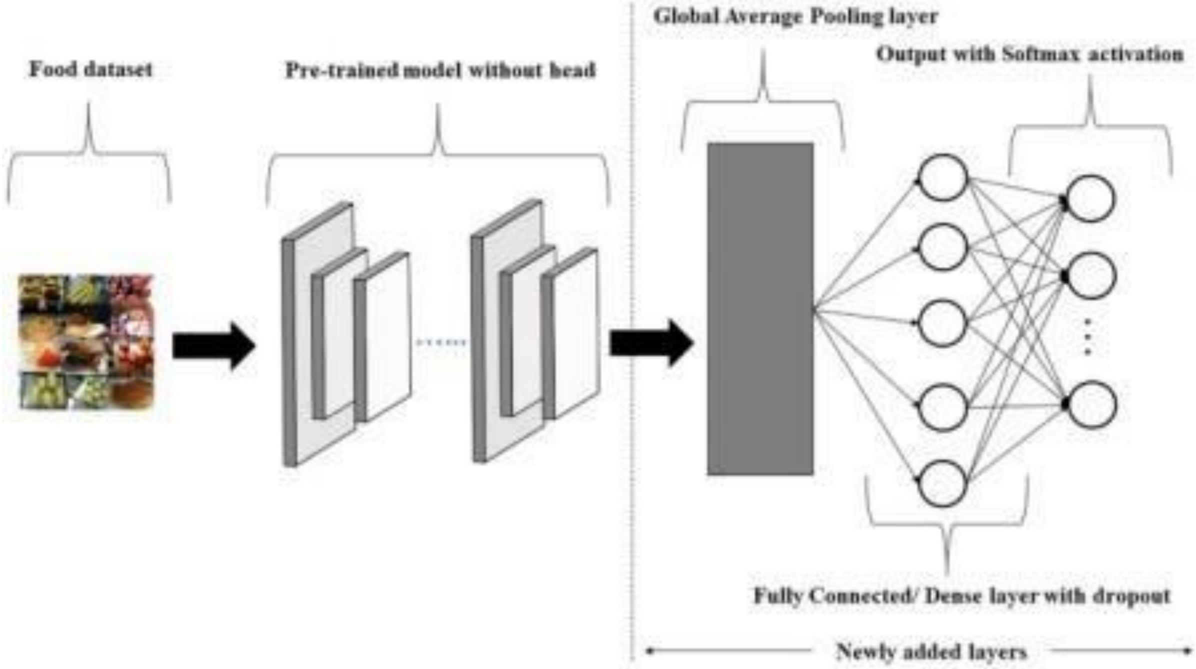
- **Data Augmentation:** Data augmentation is the application of a wide range of techniques used to generate a new randomly transformed batch of training samples replacing the original one [3]. Furthermore, it is a good method in order to avoid the phenomenon of overfitting that could affect the model in the training phase. In order to apply augmentation to the project, Data Generator allows the creation of a batch of augmented images and their correspondent masks. The modifications are applied to the images using the Keras ImageDataGenerator class that, passing some parameters, applies the same transformations to the image and its related mask.
- **Activation Functions:** The activation function of the last layer in a convolutional neural network is the one in charge of classifying the category each pixel belongs to. Since the output should be a probability, <u>softmax</u> and <u>sigmoid</u> are the principal candidates.

For a *multi-label classification* problem, where there is more than one "right answer" (the outputs are not mutually exclusive), it is usually better
to use a sigmoid function on each raw output independently. Indeed, the sigmoid allows having a high probability for all classes, for only some of them, or for none of them.

Instead of a *multi-class classification* problem, where there is only one "right answer" (the outputs are mutually exclusive), it is usually better to use a softmax function. Indeed, the softmax enforces that the sum of the probabilities of output classes are equal to one, so in order to increase the probability of a particular class, the model must correspondingly decrease the probability of at least one of the other classes.



Basic setting up of a deep CNN Architecture



Transfer learning via finetuning the pre-traineddeep CNN

Preprocessing

The first preprocessing operation that has been done is the filtering of the categories that model can detect. It has been necessary to do this since the dataset provides lots of categories and images that, if totally used for training a neural network, would require high computation capability. So, it has been agreed to reduce the number of categories down to 16 and in the same way include only those images of the dataset where these classes are present. Following this idea, for the training it has been created a subset of the entire dataset. The categories that has been taken into account are the most present in the dataset considering the entire distribution of categories that is shown in 3.1:

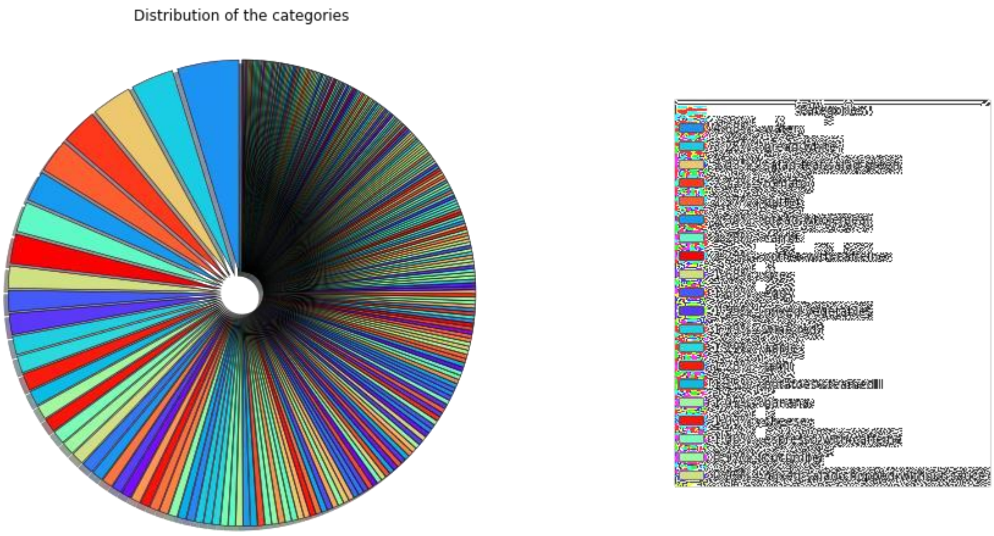


Figure 3.1: Pie chart of the distribution of the categories in the dataset

As we can see the most frequent categories are:

1. water
2. bread-white
3. salad-leaf-salad-green
4. tomato
5. butter
6. carrot
7. coffee-with-caffeine

8. rice
9. egg
10. mixed-vegetables
11. wine-red
12. apple
13. jam
14. potatoes-steamed
15. banana
16. cheese

These first 16 classes are the selected ones. Including the *background* class, the total number of categories has become 17. The dataset has been reduced as follow:

- **Train set :** from 24120 images to 10819 (and annotations from 39328 annotations to 12869)
- **Val set:** from 1269 images to 554
- **Test set :** from 1269 images to 554

Model

→Architectures

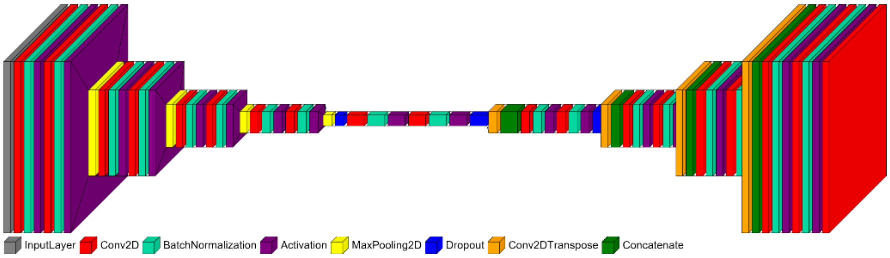
The chosen architecture for the problem resolution is U-Net, originally developed for segmenting biomedical images. When visualized its architecture looks like the letter U and hence the name U-Net. Its architecture is made up of two parts, the left part (the contracting path) and the right part (the expansive path). The purpose of the contracting path is to capture context, while therole of the expansive path is to aid in precise localization. Two U-Net like models have been investigated in order to compare their effectiveness on food recognition task.

→Basic U-Net

The first architecture is based on the original one proposed by Olaf Ronneberger et al. , and itis is illustrated in figure

The contracting path starts from an image of size *(128x128x3)* and follows the typical architecture of a convolutional network. It consists of four repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step the number of feature channels are doubled (starting from 32 up to 512), while the spatial dimensions is cutted by half (starting from 128 up to 8). The bottleneck uses two 3X3 convolutional layers to produce an internal encoding of the input image.

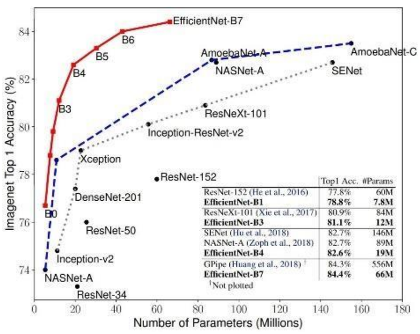
The expansion section consists of several expansion blocks with each block passing the input to two 3X3 convolutional layers and a 2x2 “up-convolution” that halves the number of feature channels. It also includes a concatenation with the correspondingly cropped feature map from the contracting path. The cropping is necessary due to the loss of border pixels in every convolution. In the end, 1X1 convolutional layer is used to produce 17 feature maps as same as the number of categories which are desired in the output.

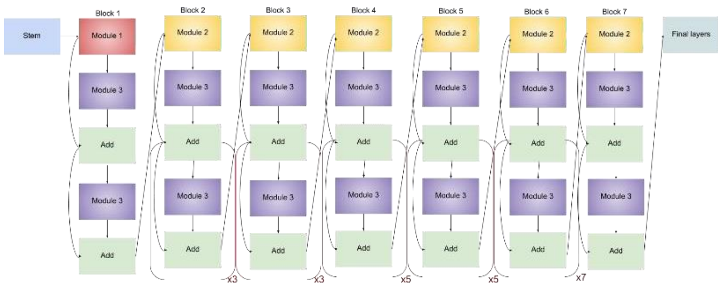


→Eff U-Net

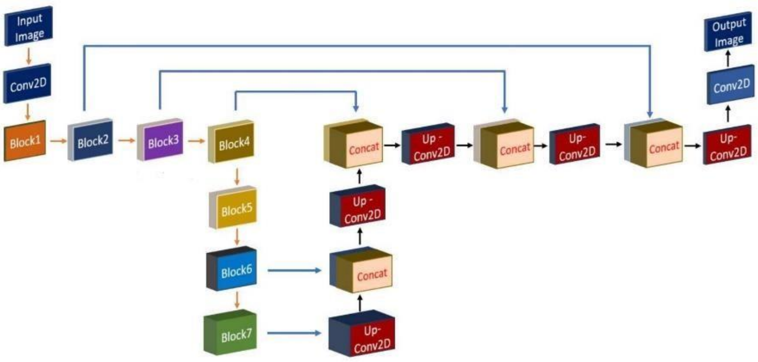
The second architecture is inspired by Baheti B. et al. [2] which propose a novel approach for semantic segmentation that use EfficientNet as feature extractor in encoder and UNet as decoder.

The EfficientNet, as proposed in [13], consists of the compound coefficient which studied model scaling and adjusted the depth, width, and the resolution of the network for better performance. A new baseline architecture called EfficientNetB0 was designed initially and it is scaled up to generate family of EfficientNet by compound scaling method. Powered by this approach, there are eight variants of the EfficientNets, namely EfficientNetB0 to EfficientNetB7. As illustrated in figure





The structure of EfficientNetB5, illustrated in figure, is composed by 7 building blocks of several mobile inverted bottleneck convolutions (MBConv) with squeeze and excitation optimization. The final Eff U-Net model appear like in figure where the decoder is similar to the original U-Net and as usual blocks from the contracting path are connected with the corresponding layers in the expansion path.

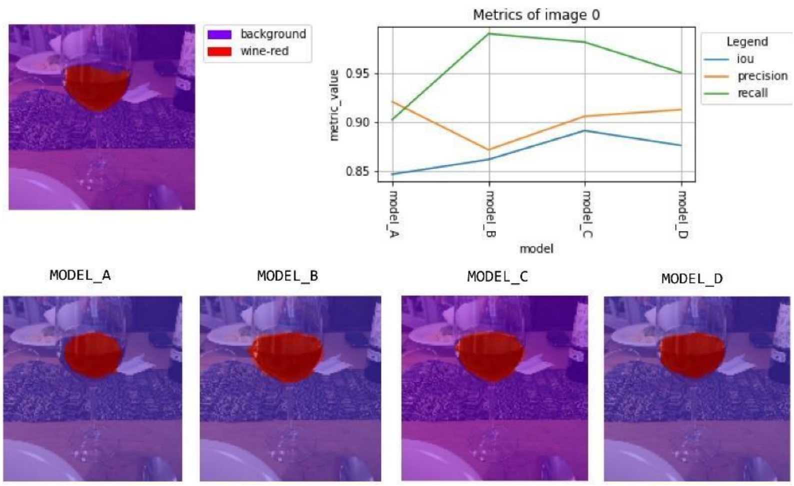


→Results

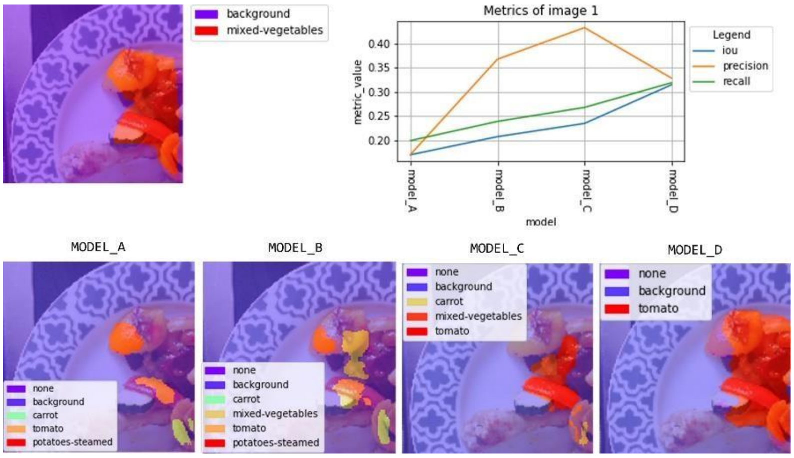
The results that can be seen in the table are the final values of every metric computing the mean of every image values. It is important to say that these results take into account all the classes present in every single image (including the background class). Following some resources it can be seen that there are conflicting opinions on whether or not to take the background class into account in the calculation of the metrics values. The final decision was to take into account also the background class following the suggestions of an AICrowd forum discussion about the evaluation criteria .

	Test		
	IoU	**Precision**	**Recall**
Model A	0.488	0.591	0.559
Model B	0.525	0.618	0.591
Model C	0.455	0.549	0.526
Model D	0.624	0.703	0.688

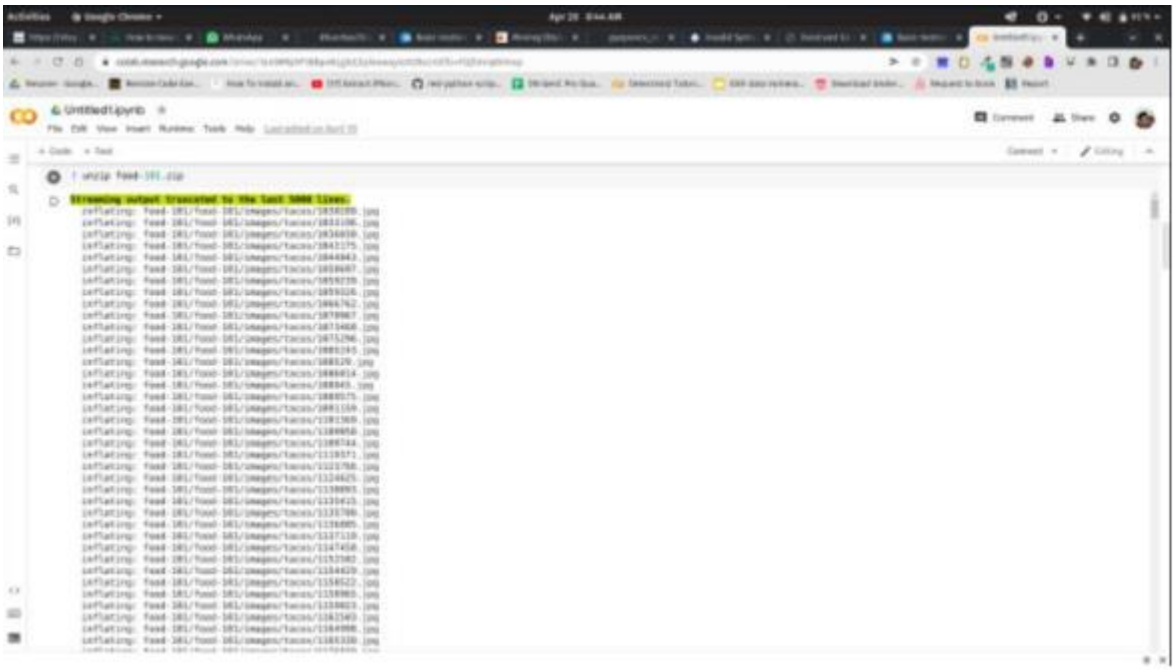
Table: Test results of every model



An example of a well done prediction



An example of a bad done prediction



```
inflating: food-101/food-101/images/tacos/1165330.jpg
inflating: food-101/food-101/images/tacos/1176039.jpg
inflating: food-101/food-101/images/tacos/1181683.jpg
inflating: food-101/food-101/images/tacos/1188062.jpg
inflating: food-101/food-101/images/tacos/1188477.jpg
inflating: food-101/food-101/images/tacos/1189793.jpg
inflating: food-101/food-101/images/tacos/1195600.jpg
inflating: food-101/food-101/images/tacos/1199299.jpg
inflating: food-101/food-101/images/tacos/1200047.jpg
inflating: food-101/food-101/images/tacos/1200178.jpg
inflating: food-101/food-101/images/tacos/1205090.jpg
inflating: food-101/food-101/images/tacos/121283.jpg
inflating: food-101/food-101/images/tacos/1213116.jpg
inflating: food-101/food-101/images/tacos/1217263.jpg
inflating: food-101/food-101/images/tacos/1224181.jpg
inflating: food-101/food-101/images/tacos/1224556.jpg
inflating: food-101/food-101/images/tacos/122809.jpg
inflating: food-101/food-101/images/tacos/1231428.jpg
inflating: food-101/food-101/images/tacos/1234952.jpg
inflating: food-101/food-101/images/tacos/123968.jpg
```

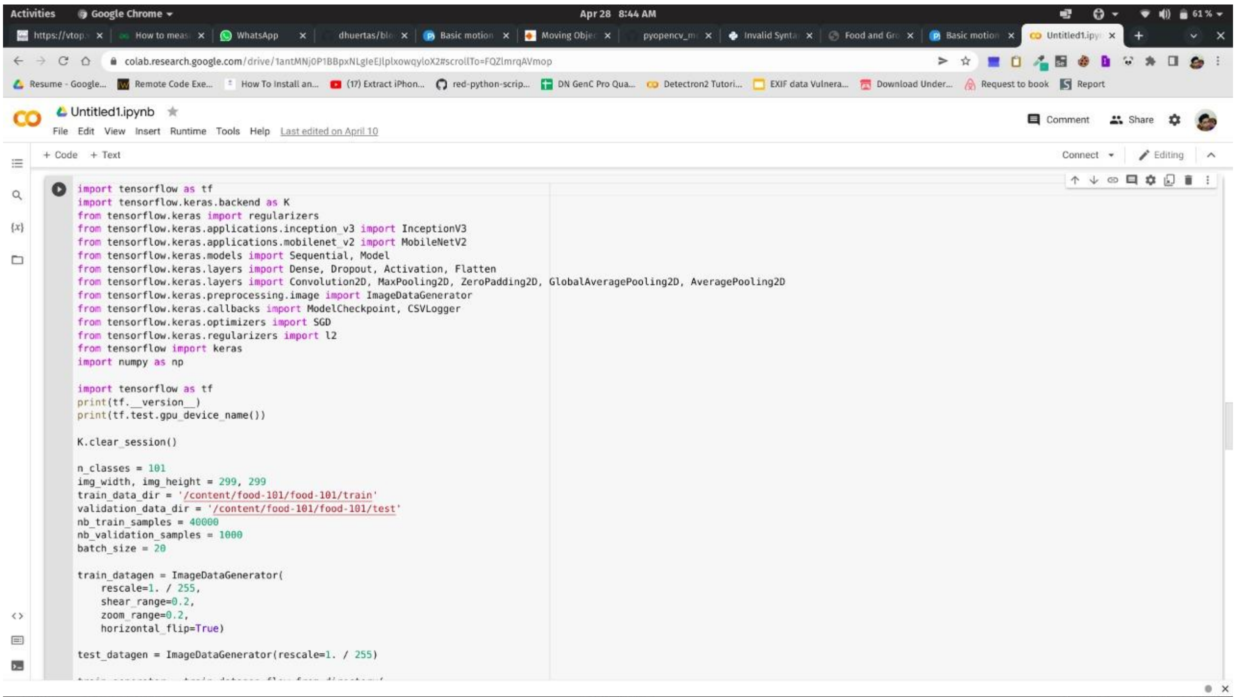
```
!nvidia-smi
```

```
NVIDIA-SMI has failed because it couldn't communicate with the NVIDIA driver. Make sure that the latest NVIDIA driver is installed and running.
```

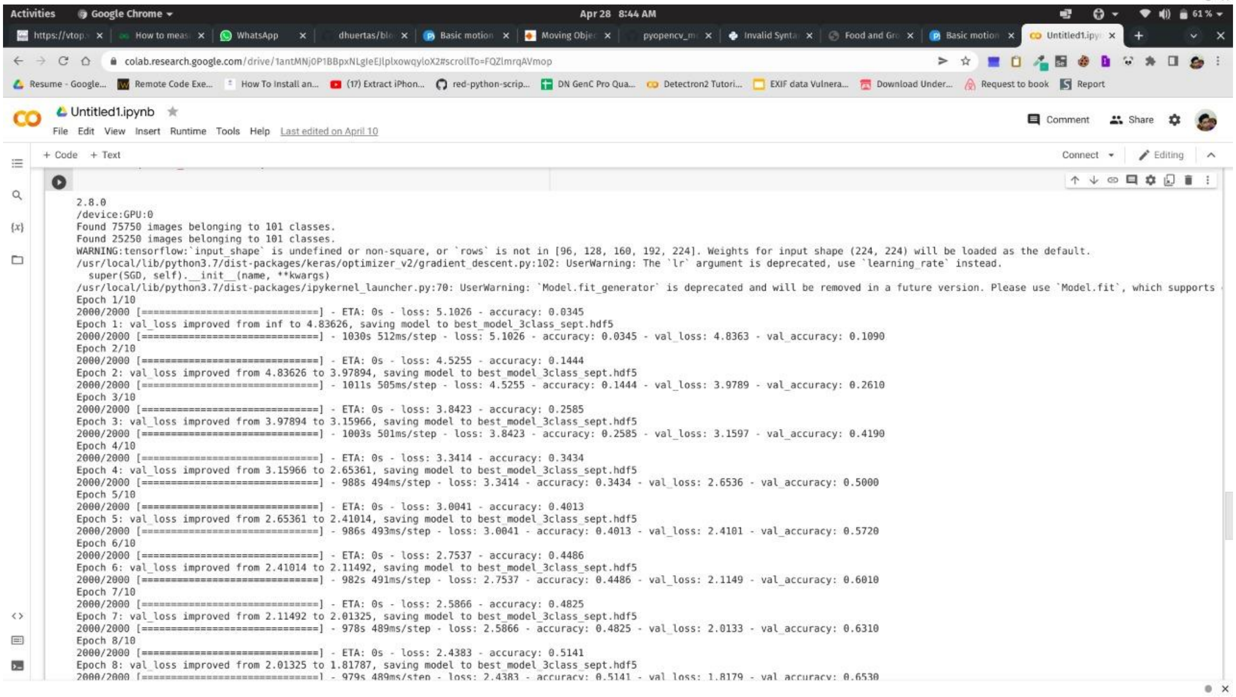
```python
from shutil import copy
from collections import defaultdict
import os


def prepare_data(filepath, src,dest):
  classes_images = defaultdict(list)
  with open(filepath, 'r') as txt:
      paths = [read.strip() for read in txt.readlines()]
      for p in paths:
        food = p.split('/')
        classes_images[food[0]].append(food[1] + '.jpg')
```

```python
from shutil import copy
from collections import defaultdict
import os


def prepare_data(filepath, src,dest):
  classes_images = defaultdict(list)
  with open(filepath, 'r') as txt:
      paths = [read.strip() for read in txt.readlines()]
      for p in paths:
        food = p.split('/')
        classes_images[food[0]].append(food[1] + '.jpg')

  for food in classes_images.keys():
    print("\nCopying images into ",food)
    if not os.path.exists(os.path.join(dest,food)):
      os.makedirs(os.path.join(dest,food))
    for i in classes_images[food]:
      copy(os.path.join(src,food,i), os.path.join(dest,food,i))
  print("Copying Done!")

prepare_data('/content/food-101/food-101/meta/train.txt', '/content/food-101/food-101/images', '/content/food-101/food-101/train')
prepare_data('/content/food-101/food-101/meta/test.txt', '/content/food-101/food-101/images', '/content/food-101/food-101/test')
```

```
Copying images into  apple_pie

Copying images into  baby_back_ribs

Copying images into  baklava

Copying images into  beef_carpaccio

Copying images into  beef_tartare

Copying images into  beet_salad

Copying images into  beignets
```



Untitled1.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 10
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras import regularizers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling2D, AveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.regularizers import l2
from tensorflow import keras
import numpy as np
import tensorflow as tf
print(tf.__version__)
print(tf.test.gpu_device_name())
K.clear_session()
n_classes = 101
img_width, img_height = 299, 299
train_data_dir = '/content/food-101/food-101/train'
validation_data_dir = '/content/food-101/food-101/test'
nb_train_samples = 40000
nb_validation_samples = 1000
batch_size = 20
train_datagen = ImageDataGenerator(
rescale=1. / 255,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1. / 255)



Untitled1.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 10
2.8.0
/device:GPU:0
Found 75750 images belonging to 101 classes.
Found 25250 images belonging to 101 classes.
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
super(SGD, self).__init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:70: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports
Epoch 1/10
2000/2000 [==============================] - ETA: 0s - loss: 5.1026 - accuracy: 0.0345
Epoch 1: val_loss improved from inf to 4.83626, saving model to best_model_3class_sept.hdf5
2000/2000 [==============================] - 1030s 512ms/step - loss: 5.1026 - accuracy: 0.0345 - val_loss: 4.8363 - val_accuracy: 0.1090
Epoch 2/10
2000/2000 [==============================] - ETA: 0s - loss: 4.5255 - accuracy: 0.1444
Epoch 2: val_loss improved from 4.83626 to 3.97894, saving model to best_model_3class_sept.hdf5
2000/2000 [==============================] - 1011s 505ms/step - loss: 4.5255 - accuracy: 0.1444 - val_loss: 3.9789 - val_accuracy: 0.2610
Epoch 3/10
2000/2000 [==============================] - ETA: 0s - loss: 3.8423 - accuracy: 0.2585
Epoch 3: val_loss improved from 3.97894 to 3.15966, saving model to best_model_3class_sept.hdf5
2000/2000 [==============================] - 1003s 501ms/step - loss: 3.8423 - accuracy: 0.2585 - val_loss: 3.1597 - val_accuracy: 0.4190
Epoch 4/10
2000/2000 [==============================] - ETA: 0s - loss: 3.3414 - accuracy: 0.3434
Epoch 4: val_loss improved from 3.15966 to 2.65361, saving model to best_model_3class_sept.hdf5
2000/2000 [==============================] - 988s 494ms/step - loss: 3.3414 - accuracy: 0.3434 - val_loss: 2.6536 - val_accuracy: 0.5000
Epoch 5/10
2000/2000 [==============================] - ETA: 0s - loss: 3.0041 - accuracy: 0.4013
Epoch 5: val_loss improved from 2.65361 to 2.41014, saving model to best_model_3class_sept.hdf5
2000/2000 [==============================] - 986s 493ms/step - loss: 3.0041 - accuracy: 0.4013 - val_loss: 2.4101 - val_accuracy: 0.5720
Epoch 6/10
2000/2000 [==============================] - ETA: 0s - loss: 2.7537 - accuracy: 0.4486
Epoch 6: val_loss improved from 2.41014 to 2.11492, saving model to best_model_3class_sept.hdf5
2000/2000 [==============================] - 982s 491ms/step - loss: 2.7537 - accuracy: 0.4486 - val_loss: 2.1149 - val_accuracy: 0.6010
Epoch 7/10
2000/2000 [==============================] - ETA: 0s - loss: 2.5866 - accuracy: 0.4825
Epoch 7: val_loss improved from 2.11492 to 2.01325, saving model to best_model_3class_sept.hdf5
2000/2000 [==============================] - 978s 489ms/step - loss: 2.5866 - accuracy: 0.4825 - val_loss: 2.0133 - val_accuracy: 0.6310
Epoch 8/10
2000/2000 [==============================] - ETA: 0s - loss: 2.4383 - accuracy: 0.5141
Epoch 8: val_loss improved from 2.01325 to 1.81787, saving model to best_model_3class_sept.hdf5
2000/2000 [==============================] - 979s 489ms/step - loss: 2.4383 - accuracy: 0.5141 - val_loss: 1.8179 - val_accuracy: 0.6530

Untitled1.ipynb

```python
import matplotlib.pyplot as plt
import numpy as np
import os
from tensorflow.keras.models import load_model


def create_foodlist(path):
    list_ = list()
    for root, dirs, files in os.walk(path, topdown=False):
      for name in dirs:
        list_.append(name)
    return list_


my_model = load_model('model_trained.hdf5', compile = False)
food_list = create_foodlist("/content/food-101/food-101/images")


def predict_class(model, images, show = True):
  for img in images:
    img = image.load_img(img, target_size=(299, 299))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img /= 255.

    pred = model.predict(img)
    index = np.argmax(pred)
    food_list.sort()
    pred_value = food_list[index]
    if show:
        plt.imshow(img[0])
        plt.axis('off')
        plt.title(pred_value)
        plt.show()

images = []
images.append('/content/pie2.jpg')
#images.append('29744.jpg')
```

Untitled1.ipynb

```python
def predict_class(model, images, show = True):
  for img in images:
    img = image.load_img(img, target_size=(299, 299))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img /= 255.

    pred = model.predict(img)
    index = np.argmax(pred)
    food_list.sort()
    pred_value = food_list[index]
    if show:
        plt.imshow(img[0])
        plt.axis('off')
        plt.title(pred_value)
        plt.show()

images = []
images.append('/content/pie2.jpg')
#images.append('29744.jpg')
#images.append('pie.jpg')


print("PREDICTIONS BASED ON PICTURES UPLOADED")
predict_class(my_model, images, True)
```

```
PREDICTIONS BASED ON PICTURES UPLOADED
```

waffles

References:

[1] Vardan Agarwal. Complete Architectural Details of all EfficientNet Models. 2020. URL: Complete Architectural Details of all EfficientNet Models | by Vardan Agarwal | Towards Data Science

[2] Tsung-Yi Lin. COCO API. URL: GitHub - cocodataset/cocoapi: COCO API - Dataset @ http://cocodataset.org/

[3]Adrian Rosebrock. Keras ImageDataGenerator and Data Augmentation. URL: Keras ImageDataGenerator and Data Augmentation - PyImageSearch

[4] Kong F, Tan J (2012) Dietcam: Automatic dietary assessment with mobile camera phones. Pervasive Mob Comput 8(1):147–163 URL:https://www.sciencedirect.com/science/article/abs/pii/S1574119211001131?via%3Dihub

[5] Matsuda Y, Hoashi H, Yanai K (2012) Recognition of multiple-food images by detecting candidate regions. In 2012 IEEE International Conference on Multimedia and Expo. Melbourne, Australia, pp 25–30

→https://www.academia.edu/51004823/Study_for_Food_Recognition_System_Using_Deep_Learning

→https://www.academia.edu/58490038/Smartphone_based_food_recognition_system_using_multiple_deep_CNN_models

→https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8666636

→https://www.mdpi.com/2079-9292/8/12/1425

→https://www.sciencedirect.com/science/article/pii/S1877050920316331