26-Sept-2022

STUDENTS

Abhijith V Pilllai
Sooraj Varmma
Yadu Krishna C
Nedhiya Sunny

Food Classification using Deep Learning

Table of Contents

I INTRODUCTION

- Food plays a vital role in human's life, it provides us different nutrients
- It is very necessary to maintain a watch on the eating habits.
- people across the world suffering from obesity are more than 10% and it increasing in an alarming rate.
- Statistics show that 95% of the people no longer follow any dietary plan.

food image classification can improve food experiences across the board, such as to recommend dishes and new eateries, improve cuisine lookup, and help people make the right food choices for their diets. In this paper, we explore the problem of food image classification through training convolutional neural networks, both from scratch and with pre-trained weights learned on a larger image dataset (transfer learning), achieving an accuracy of 61.4% and top-5 accuracy of 85.2%.

II AIM

Through our project we aim to make people recognise what they eat and to help them understand the nutrient values and help them fix their diet.

Giving an image of a dish as the input to the model, it predicts the food and its nutrient values.

We also collect the data like height,weight and activity of the user to help them understand their current BMI,BMR,AMR values and to provide them with daily calorie need, also help them fix their daily diet plan by just uploading what they want to have.

III EXISTING SYSTEM

Currently Google lens provides all the necessary information about the given food item which mainly includes recipes.
A subsequent study on food image classification focused solely on the use of CNNs [5] constructed a five-layer CNN to recognize a subset of ImageNet data [6], which consisted of ten food classes. Lu's approach showcases the higher potential of CNN versus a bag-of-features (BoF) model, with the CNN model outperforming BoF by 74% to 56% accuracy. Additional data augmentation techniques were applied to bring up accuracy to 90%, which far outpaces the best BoF performance. However, given the much reduced number of classes, the paper's model performance cannot be directly mapped to model performance on the Food-101 dataset.

III EXISTING SYSTEM

More recently, Liu et al. implemented DeepFood [7], a CNN-based approach inspired by LeNet-5 [8], AlexNet [9], and GoogleNet [10], employing Inception modules to increase the overall depth of the network structure. DeepFood achieved 77.4% top-1 accuracy on the Food-101 dataset after 300,000 epochs. On a separate food dataset, they were able to further improve their model performance by utilizing bounding boxes to crop the image to just the dish, eliminating background noise.

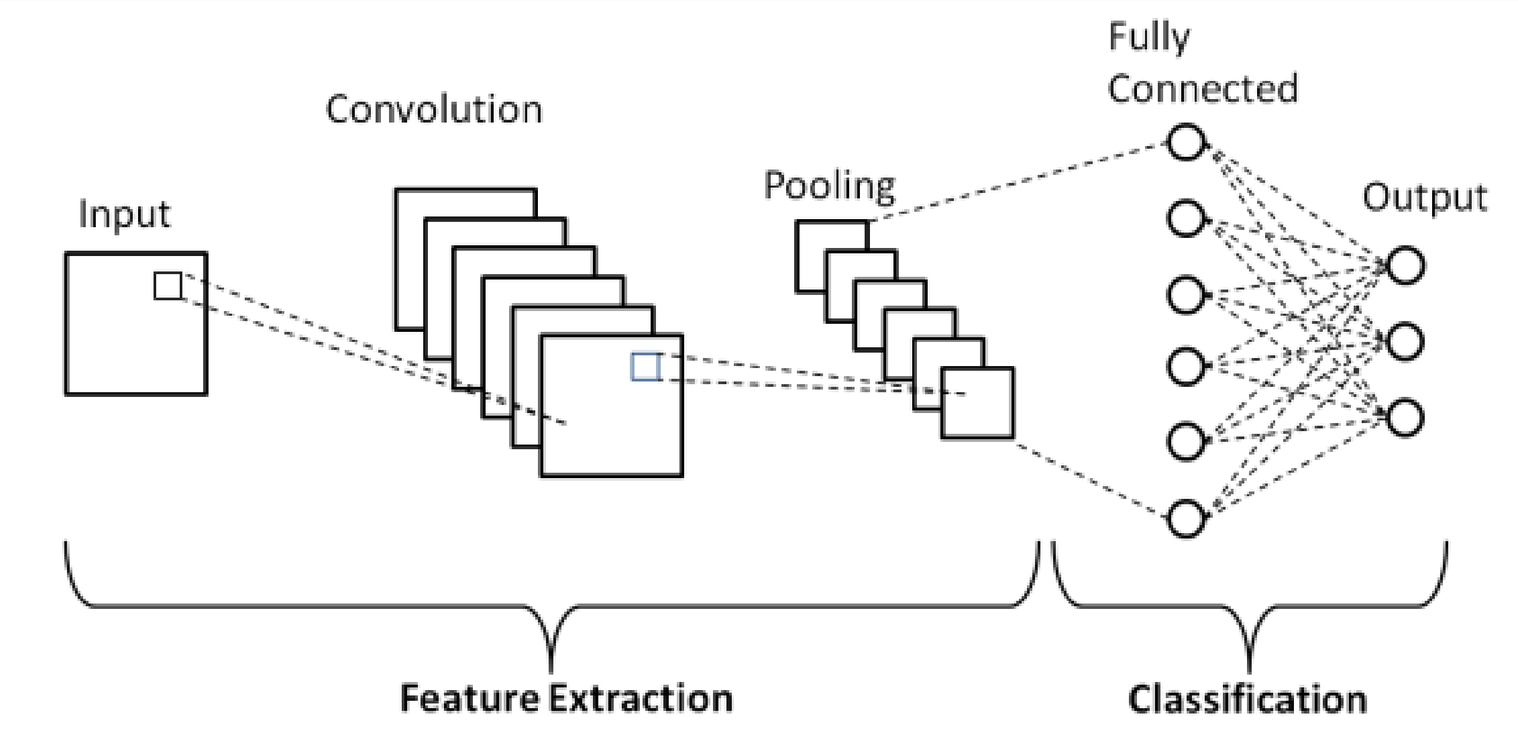
IV PROPOSED SYSTEM

Our proposed approach was directly inspired by and rooted from LeNet-5 [35], AlexNet [21], and GoogleNet . The original idea of CNN was inspired by the neuroscience model of primate visual cortex.

The key insights from the paper is how to make machine learning with multiple level neurons like the human mind. In the human brain, it's known that different neurons control different perception functionality. How to make the computer recognize and think in a human-like way has long been a topic for many artificial intelligence experts. In the article by LeCun et al they proposed the initial structure of LeNet-5, which is considered to be the first successful trial in deep learning. In their paper, a 7-layer network structure is proposed to represent human-written digital characters and used for digits recognition.

IV PROPOSED SYSTEM

The proposed system consists of four phases: Image pre-processing, Feature selection and extraction, Classification and Output.



A block diagram stating the design of the CNN Model

IV PROPOSED SYSTEM

MobileNet V2 which uses CNN technique for recognising image and the image thus recognised is again checked in our nutrition dataset for getting the calories,fat,protein and carbs present in 100g of given food item.
We also present a simple and user-friendly interface for the user to interact with the backend and to help them with their diet plans for weight-gain,weight-loose etc.

V Functional analysis

The proposed system works with the help of Convolutional Neural Network(CNN), uses the model MobileNetV2, steps included are :-

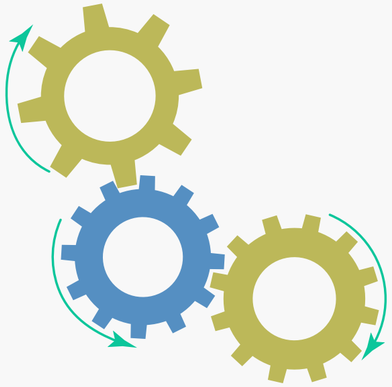
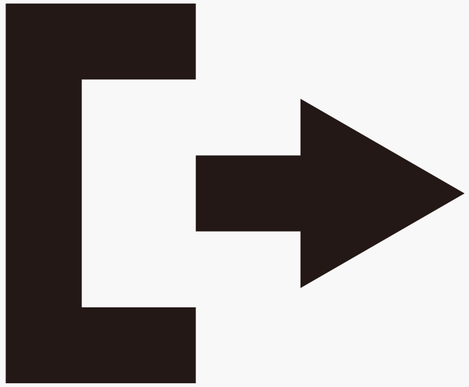


Image pre-processing

It is the operations on images at the lowest level of abstraction. These operations do not increase image information content but they decrease it if entropy is an information measure.

Feature selection,Extraction

Used for dimensionality reduction which is key to reducing model complexity and over fitting. dimensionality reduction is one of the most important aspects of training machine learning models.



Classification and output

The given image is then classified and the prediction is done

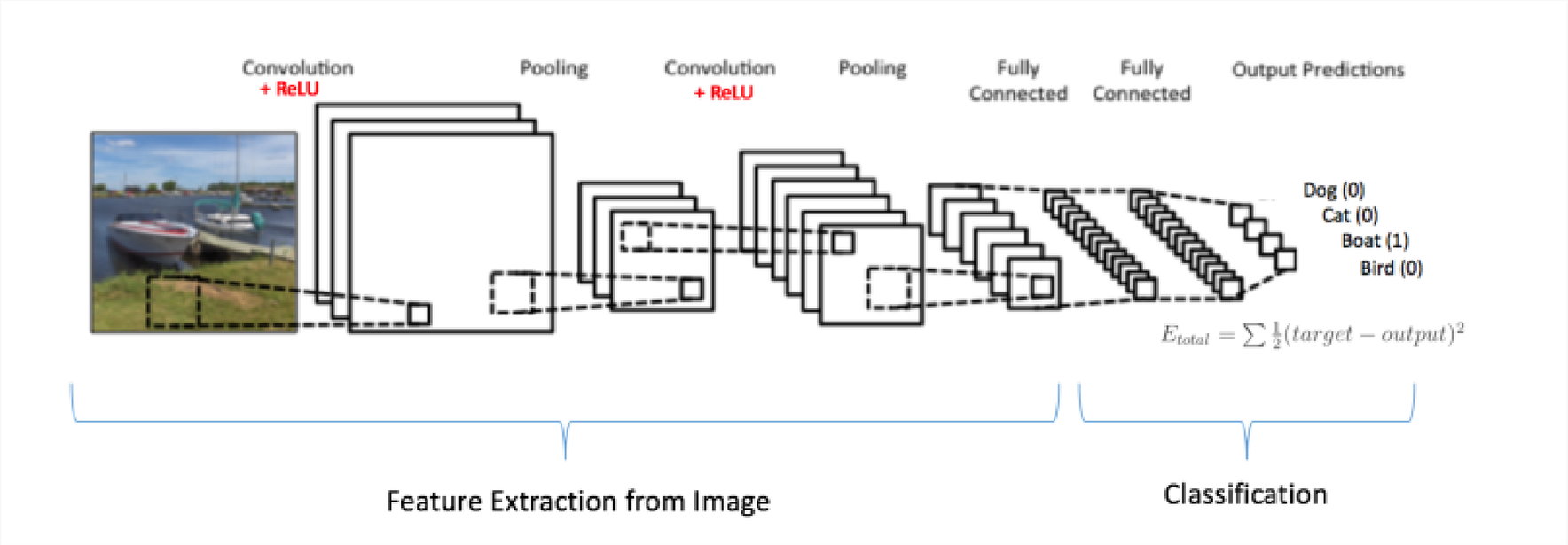
V Functional analysis

Image pre-processing

- Food 101 dataset contains 101 food items each class having 1000 food each
- each image is cropped into size 299 x 299
- 750 images is used for training purpose
- 250 image is used for testing purpose
- destination of testing and training image is specified in train and test file in meta dir
- All unwanted images are manually removed and new ones are added

V Functional analysis

Feature selection and Extraction



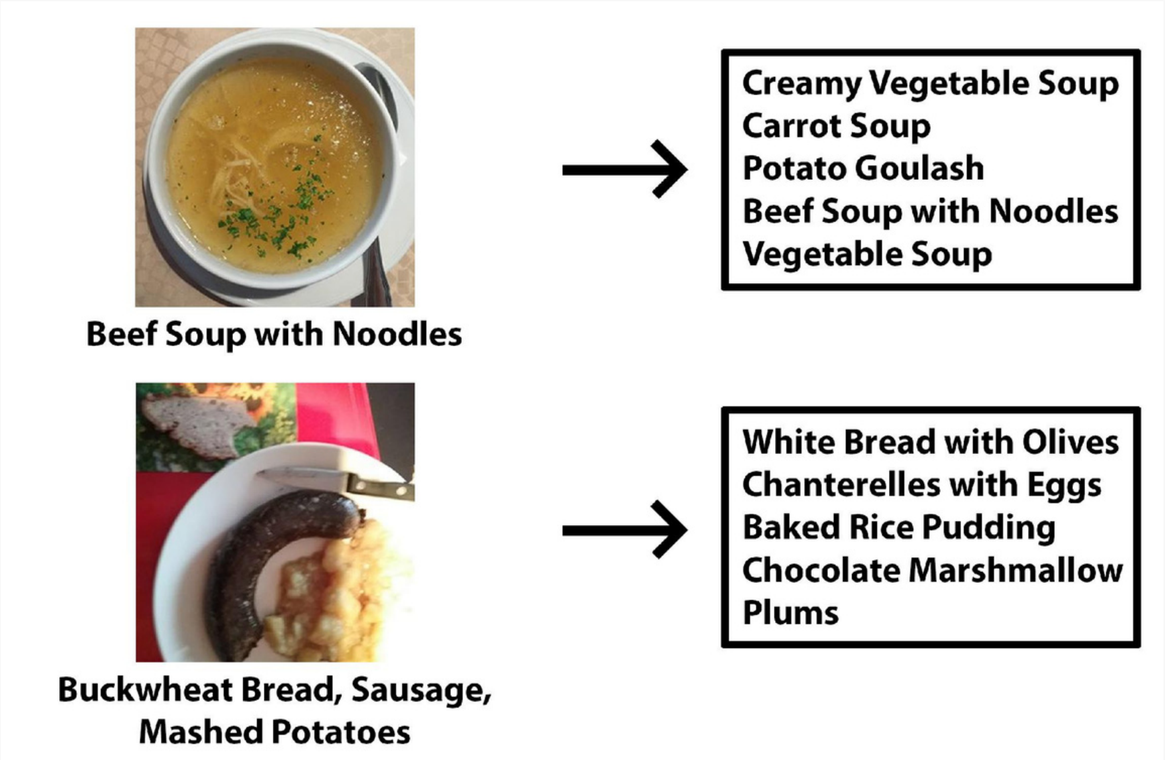
Feature Selection:- This module is used for feature selection/dimensionality reduction on given datasets. This is done either to improve estimators' accuracy scores or to boost their performance on very high-dimensional datasets.

Feature Extraction:- This module is used to extract features in a format supported by machine learning algorithms from the given datasets consisting of formats such as text and image.

V Functional analysis

Classification and output

Trained model generates an array which has 101 elements each element is a value and the index of the highest value is given and this index value is used to predict the name of the item



[[4.09880013e-04 1.75425346e-04 1.54017016e-05 3.78764776e-06
2.87252005e-05 2.39710539e-06 5.25783871e-05 3.64836865e-06
3.88090084e-05 7.13496820e-06 1.35127862e-04 2.01592570e-06
1.99040260e-05 1.79127005e-06 2.90905919e-05 9.19439208e-06
2.38178793e-04 7.02749749e-05 4.07751668e-06 3.73373700e-06
2.50982004e-04 1.34260990e-05 1.21229468e-05 8.05689301e-03
5.62445930e-05 1.17368437e-03 9.06098376e-06 6.06061221e-05
2.13950087e-04 3.52410484e-06 3.78020695e-06 1.35315233e-04
1.12188085e-04 7.78620451e-05 1.16673118e-05 2.21790970e-06
5.60731269e-06 1.69580871e-05 2.13032160e-02 4.91840765e-05
9.40432489e-01 3.62113533e-05 3.34404205e-04 1.72700034e-04
9.45541706e-06 2.07480389e-05 4.23588877e-04 2.75111524e-05
3.42381304e-06 8.27680808e-03 9.94165457e-05 1.06783928e-05
7.82694333e-05 8.76818958e-04 4.97202018e-05 9.49737895e-03
9.41947462e-07 2.65705730e-05 1.61375829e-05 1.65105248e-05
1.75364112e-04 1.02222036e-03 2.04949683e-04 5.82995744e-05
6.12389995e-05 4.50936795e-06 4.97111614e-06 1.53607034e-05
1.07561573e-04 2.08879942e-06 2.47372027e-05 1.86120394e-06
2.87787261e-05 5.59624868e-06 6.07840557e-05 1.29018936e-05
1.69637151e-05 5.06935430e-05 3.34563362e-03 1.08457880e-05
6.30733382e-04 1.87791993e-05 7.43245846e-06 8.09896301e-06
2.15964383e-06 8.71797693e-06 5.49734250e-06 1.58588282e-05
2.15117198e-05 6.92537185e-07 2.20174879e-05 2.66875195e-05
5.57132007e-04 7.04817649e-05 1.45889808e-05 2.12295345e-05
5.37838150e-06 4.04761022e-06 1.89066250e-05 9.04067292e-06
8.62938250e-05]]

VI MODEL TRAINING

MobileNet-v2 is a convolutional neural network that is 53 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database . The pretrained network can classify images into 1000 object categories The network has an image input size of 299-by-299.
To retrain the network on a new classification task:-

net = mobilenetv2 returns a MobileNet-v2 network trained on the ImageNet data set.
net = mobilenetv2('Weights','imagenet') returns a MobileNet-v2 network trained on the ImageNet data set. This syntax is equivalent to net = mobilenetv2.
lgraph = mobilenetv2('Weights','none') returns the untrained MobileNet-v2 network architecture. The untrained model.

Here a total of 75750 images is been trained with a total of 10 epochs and the model is saved as 'model_trained.h5'

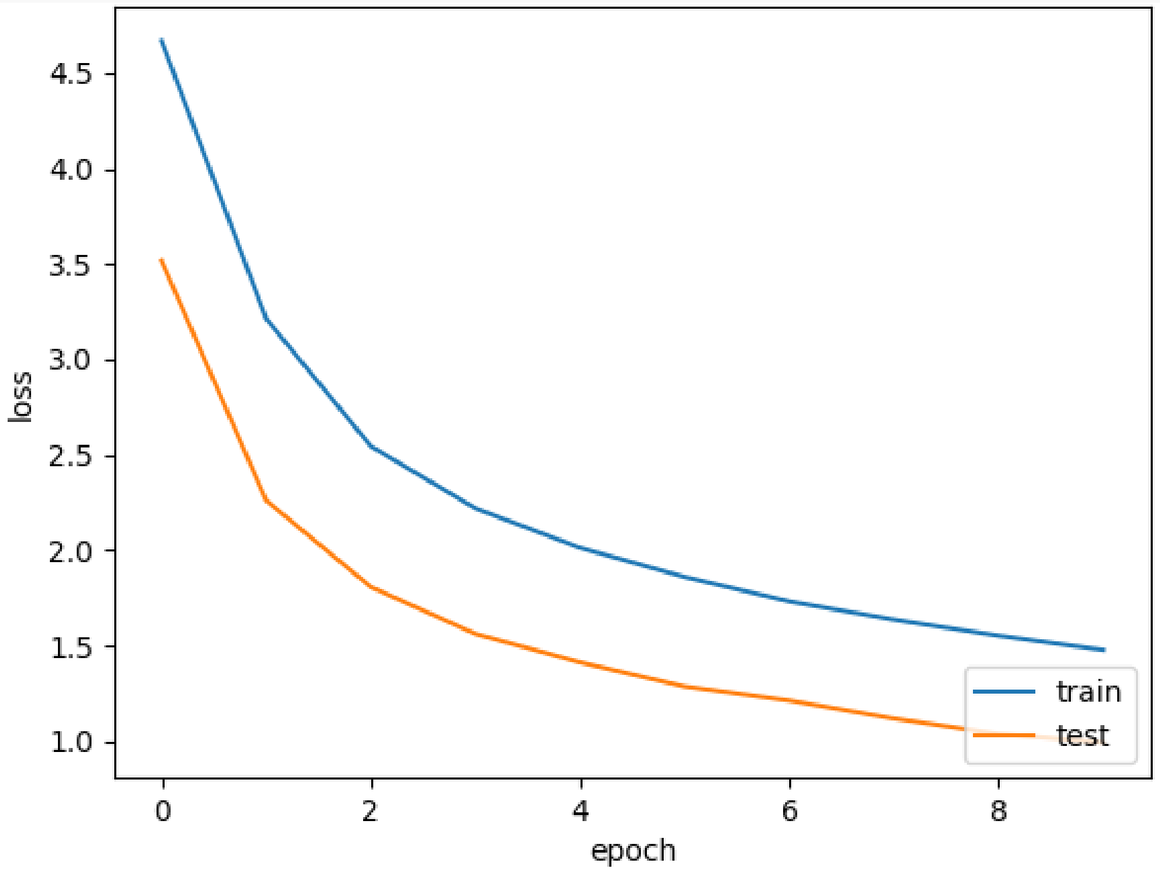
VI MODEL TRAINING

History log

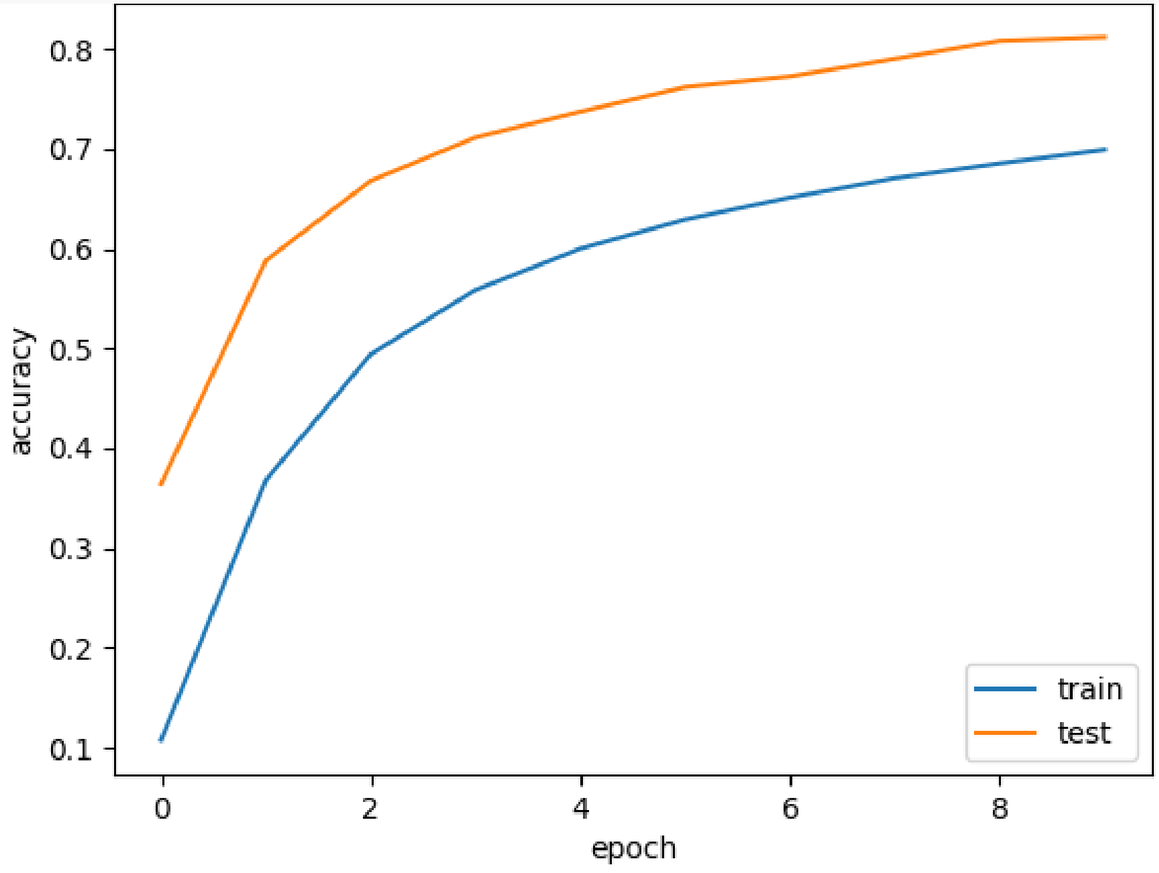
epoch	accuracy	loss	val_accuracy	val_loss
0	0.10765346884727478	4.6699652671813965	0.364421546459198	3.517435073852539
1	0.3682970404624939	3.213324546813965	0.5882725715637207	2.2608249187469482
2	0.49433663487434387	2.5452075004577637	0.6681457757949829	1.8078418970108032
3	0.5586930513381958	2.2194457054138184	0.7115689516067505	1.5617460012435913
4	0.6000891327857971	2.0130374431610107	0.7373613119125366	1.4127206802368164
5	0.6293069124221802	1.8600205183029175	0.7619651556015015	1.2862021923065186
6	0.6512871384620667	1.732200026512146	0.7727020382881165	1.2122372388839722
7	0.6707326769828796	1.6361523866653442	0.7903724312782288	1.118086576461792
8	0.6855445504188538	1.5524297952651978	0.8083597421646118	1.0376791954040527
9	0.6992475390434265	1.4774245023727417	0.8121235966682434	0.9927214980125427

VI MODEL TRAINING

Model loss



model accuracy

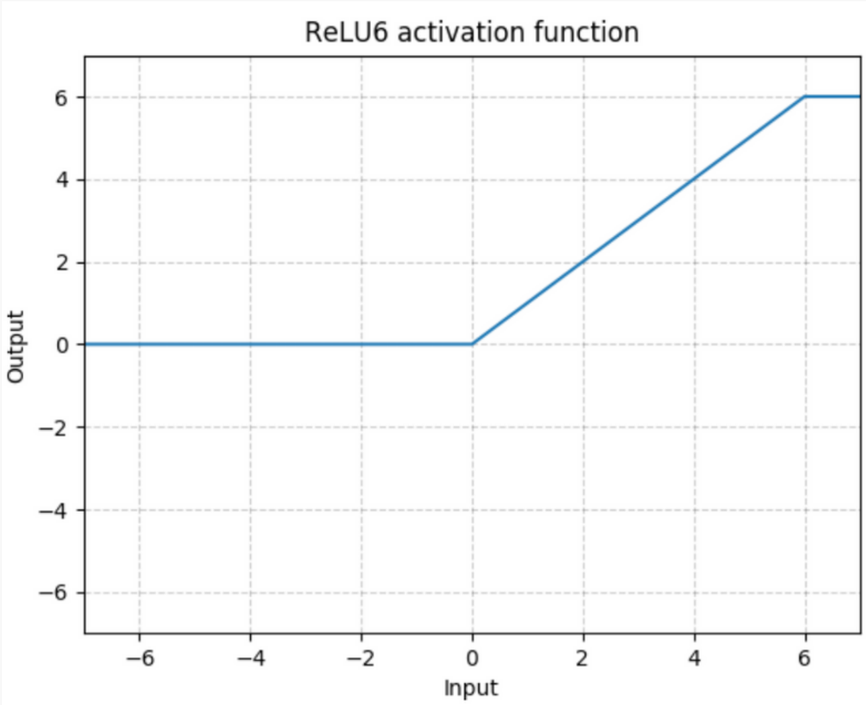


VII MobileNet Model

Mobilenet v1

In MobileNetV1, there are 2 layers.

- The first layer is called a depth wise convolution, it performs lightweight filtering by applying a single convolutional filter per input channel.
- The second layer is a 1×1 convolution, called a pointwise convolution, which is responsible for building new features through computing linear combinations of the input channels.
- ReLU6 is used here for comparison.
- ReLU6 is used due to its robustness when used with low-precision computation, based on MobileNetV1.



VII MobileNet Model

Mobilenet v2

- In MobileNetV2, there are two types of blocks. One is residual block with stride of 1. Another one is block with stride of 2 for downsizing.
- There are 3 layers for both types of blocks.
- This time, the first layer is 1×1 convolution with ReLU6.
- The second layer is the depthwise convolution.
- The third layer is another 1×1 convolution but without any non-linearity. It is claimed that if ReLU is used again, the deep networks only have the power of a linear classifier on the non-zero volume part of the output domain

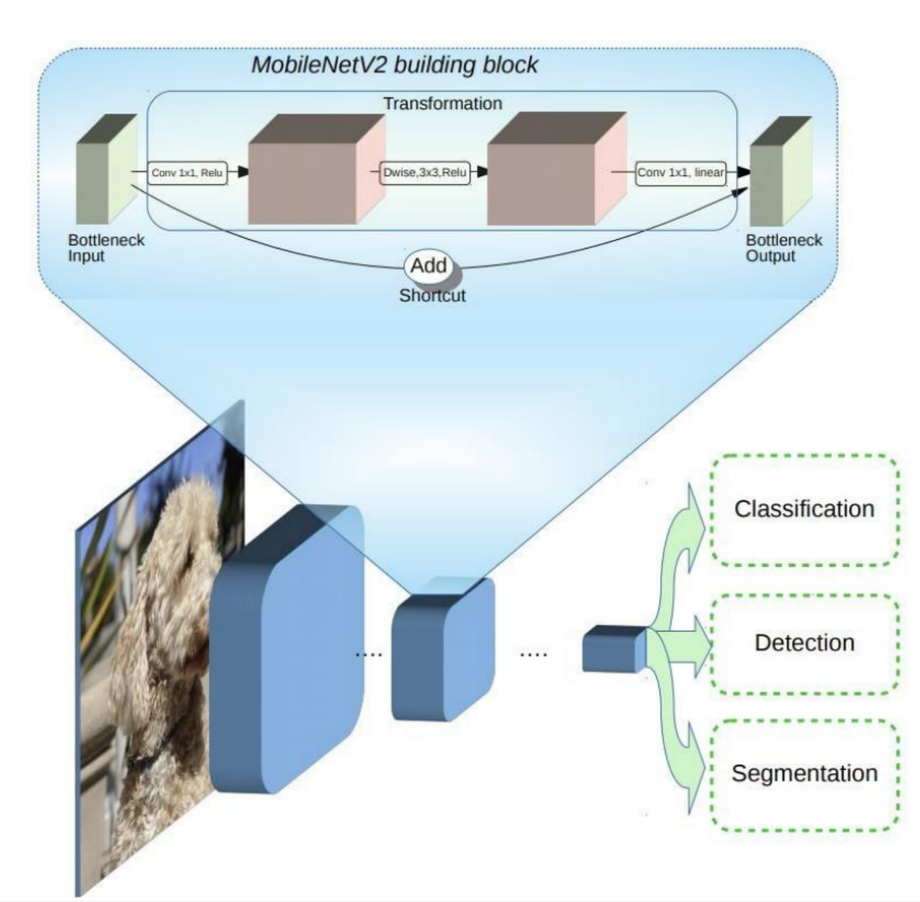
Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

VII MobileNet Model

Mobilenet v2

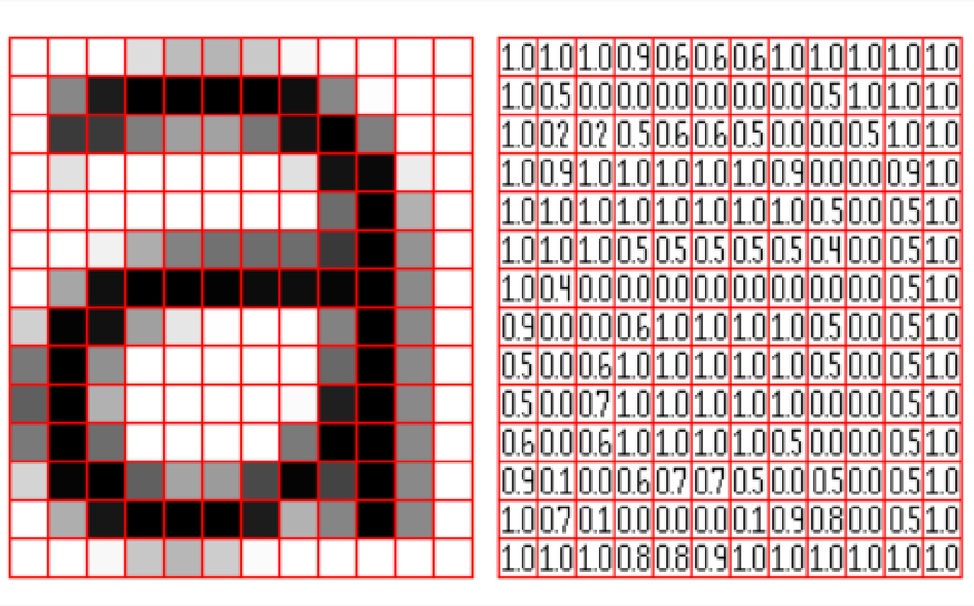
- And there is an expression factor t. and t=6 for all main experiments.
- If tht got input got 64 channels, the internal output would get

  64×t=64×6=384 channels



IX convNet architecture

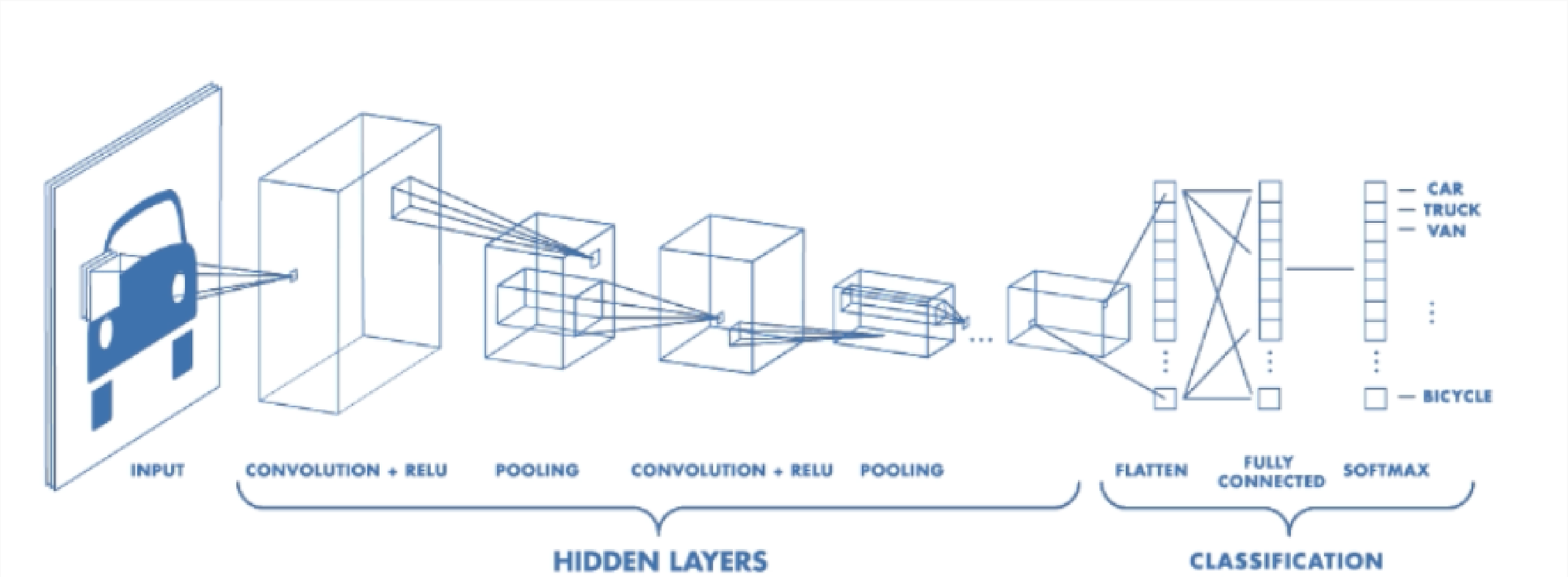
The human brain processes a huge amount of information the second we see an image. Each neuron works in its own receptive field and is connected to other neurons in a way that they cover the entire visual field. Just as each neuron responds to stimuli only in the restricted region of the visual field called the receptive field in the biological vision system, each neuron in a CNN processes data only in its receptive field as well. The layers are arranged in such a way so that they detect simpler patterns first (lines, curves, etc.) and more complex patterns (faces, objects, etc.) further along. By using a CNN, one can enable sight to computers.



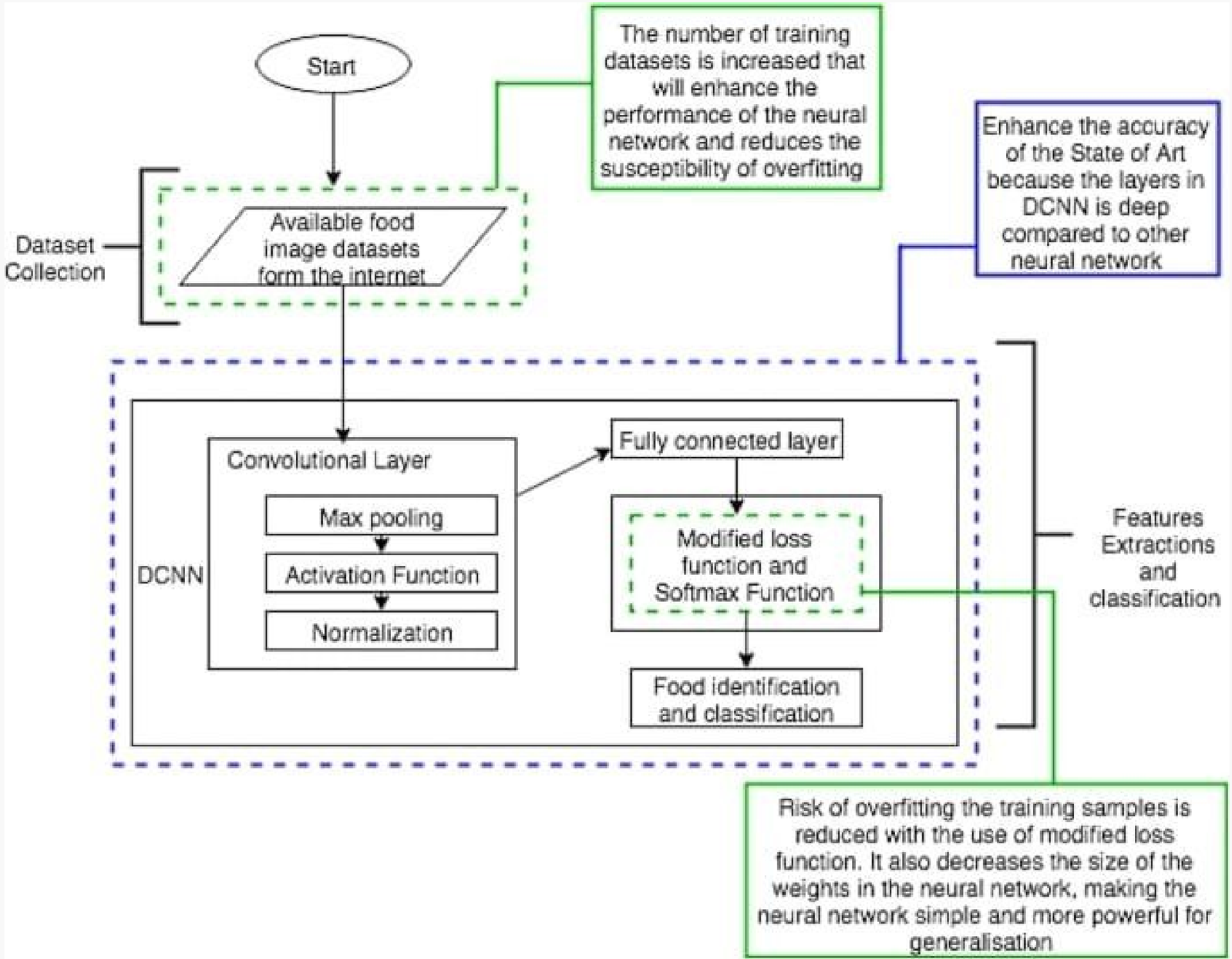
IX convNet architecture

CNN Architecture

A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.



X DATA FLOW DIAGRAM



XI DATASET

A total of 101,000 images from 101 classes of food were used from the Food-101 dataset, with 1000 images for each class. Of the 1000 images for each class, 250 were manually reviewed test images, and 750 were intentionally noisy training images, for a total training data size of 75,750 training images and 25,250 test images.

XII implementation

Libraries and packages used

The initial step that we demand to do is to import libraries. We use TensorFlow, NumPy, os, and pandas.

Some important packages includes :-

- KERAS
- CARET
- SHUTIL
- FRAMER MOTION
- RECHARTS
- COREUI
- STYLED-COMPONENTS
- REDUX

XII implementation

Preparing The Data

After you arrange the libraries, the following step is to fix our dataset. In this example, we will apply a dataset named Food-101.This dataset consists of 100000 pictures with 101 categories, FOOD-101 is partitioned into training, validation, and a test collection of data. Every folder consists of pictures. Each picture filename includes the category and the identifier of it broken by an underscore. We need to produce the dataframe with columns as the image filename and the label with that folder arrangement. The subsequent step is to make an object to put the pictures into the model. We will practice the ImageDataGenerator object of tf.keras.preprocessing.image library. With this object, we will produce image batches. Also, we can augment our picture to largen the product of the dataset. Because we also extend those pictures, we further set parameters for the image augmentations technique.
Also, because we apply a dataframe as the knowledge about the dataset, we will exercise the flow_from_dataframe method to produce batches and augment the pictures.

XII implementation

Training the model

We use MobileNet V2 for training our model, we use transfer learning technique mobileNet V2 is already trained in imagenet dataset and we use the weight of this model.
Implementing CNN from scratch is not recommended.
We have 10 epochs and accuracy in each epochs are:-

epoch	accuracy	loss	val_accuracy	val_loss
0	0.10765346884727478	4.6699652671813965	0.364421546459198	3.517435073852539
1	0.3682970404624939	3.213324546813965	0.5882725715637207	2.2608249187469482
2	0.49433663487434387	2.5452075004577637	0.6681457757949829	1.8078418970108032
3	0.5586930513381958	2.2194457054138184	0.7115689516067505	1.5617460012435913
4	0.6000891327857971	2.0130374431610107	0.7373613119125366	1.4127206802368164
5	0.6293069124221802	1.8600205183029175	0.7619651556015015	1.2862021923065186
6	0.6512871384620667	1.732200026512146	0.7727020382881165	1.2122372388839722
7	0.6707326769828796	1.6361523866653442	0.7903724312782288	1.118086576461792

XII implementation

Front-end

- ReactJS, Express framework, NodeJS are used in the front end.
- An API call is used to connect the front-end to the back-end.
- We can run the front-end and the backend in different servers if needed.
- React frontend runs in port 3000 of the local system.
- Axios is used for the API call

XII implementation

Backend

- Flask 2.2.2 is used in the backend.
- It runs at port 5000 of the local system.
- All the model calculations and predictions are done in the backend.
- A string value is returned to the frontend which has the predicted name

Api documentation

address=http://localhost:5000

Get predicted food item
Api: address+ ‘/Upload’
Input: ‘filename’(string), ‘file’(blob)
Output: ‘response.data’(String)

Thank you