

Food detection & Calorie estimation using images

Group - Code Wars

Abhitej Thonupunoori
Manoj Kumar Madireddy
Aravind Kumar Gardas

Introduction

The main aim of the project is to build a food recognition model that predicts the food captured using the camera and also estimate its calorie, to help patients and dietitians in managing daily food intake.

Technology & Libraries

- Deep Learning - CNN
- Libraries used -
 1. Numpy, Pandas for preprocessing the data
 2. Open-CV, to read the image
 3. Keras, for training the data & visualizing the data
- Interface - Flask, Gradio

Data set

We have downloaded data from Kaggle,

<https://www.kaggle.com/datasets/aelchimminut/fruits262>

We have classified 6 food types from the data set, each category has 400 images uploaded to respective folders in google drive, which is mounted to google colab for project implementation.

Categories: Apple, Banana, Cherry, Pineapple, Lemon, Tomato

320 images from each category are used for training the model & 80 images from each category are used for testing the model.

Data pre-processing

- Converting the data types
- Image reshaping
- Image scaling between values 0 and 1
- Transforming the target data to binary class matrix

Model summary

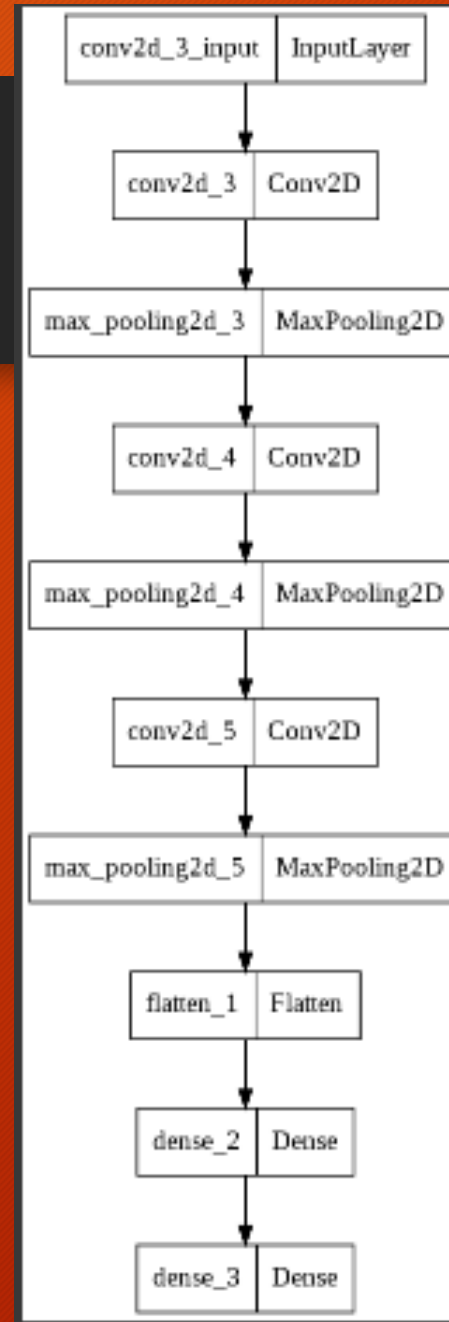
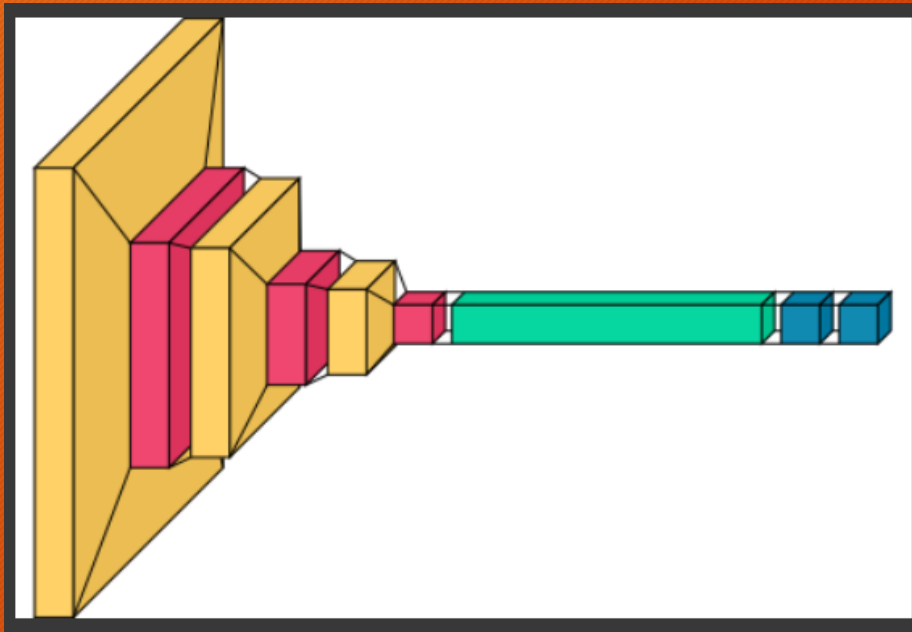
Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 58, 58, 32)	2432
max_pooling2d_3 (MaxPooling 2D)	(None, 29, 29, 32)	0
conv2d_4 (Conv2D)	(None, 27, 27, 64)	18496
max_pooling2d_4 (MaxPooling 2D)	(None, 13, 13, 64)	0
conv2d_5 (Conv2D)	(None, 11, 11, 64)	36928
max_pooling2d_5 (MaxPooling 2D)	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dense_2 (Dense)	(None, 128)	204928
dense_3 (Dense)	(None, 6)	774

=====
Total params: 263,558
Trainable params: 263,558
Non-trainable params: 0

None

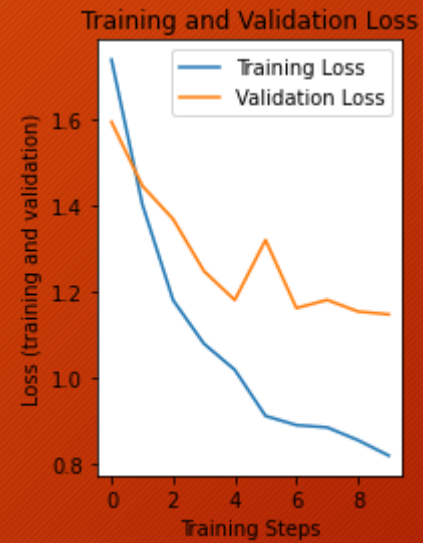
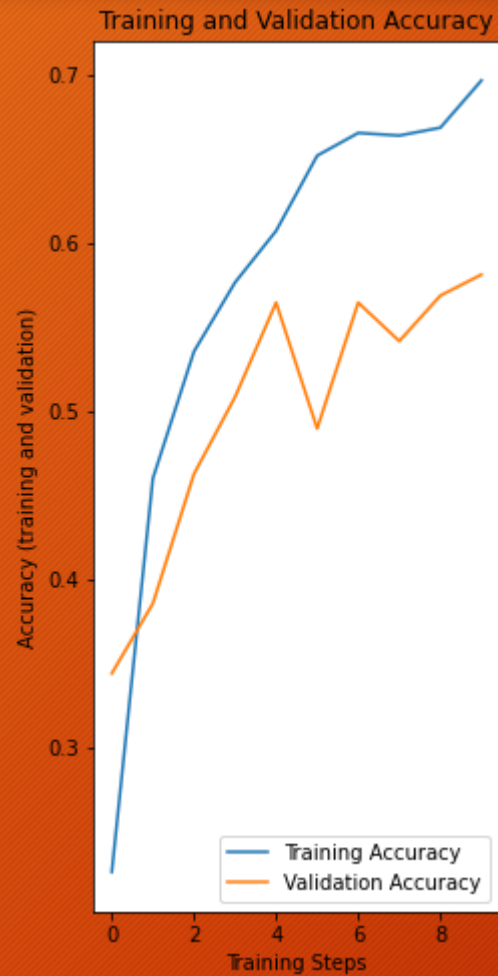
Model flow diagram



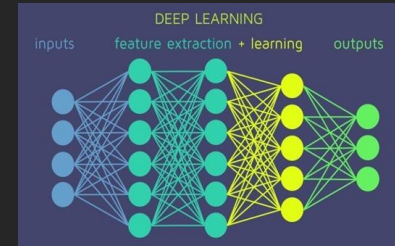
Model performance

```
Epoch 1/10
15/15 [=====] - 13s 822ms/step - loss: 1.7382 - accuracy: 0.2255 - val_loss: 1.5936 - val_accuracy: 0.3438
Epoch 2/10
15/15 [=====] - 12s 820ms/step - loss: 1.4024 - accuracy: 0.4599 - val_loss: 1.4452 - val_accuracy: 0.3854
Epoch 3/10
15/15 [=====] - 12s 805ms/step - loss: 1.1796 - accuracy: 0.5354 - val_loss: 1.3678 - val_accuracy: 0.4625
Epoch 4/10
15/15 [=====] - 12s 809ms/step - loss: 1.0785 - accuracy: 0.5766 - val_loss: 1.2473 - val_accuracy: 0.5083
Epoch 5/10
15/15 [=====] - 12s 804ms/step - loss: 1.0180 - accuracy: 0.6073 - val_loss: 1.1799 - val_accuracy: 0.5646
Epoch 6/10
15/15 [=====] - 12s 807ms/step - loss: 0.9108 - accuracy: 0.6521 - val_loss: 1.3194 - val_accuracy: 0.4896
Epoch 7/10
15/15 [=====] - 12s 807ms/step - loss: 0.8895 - accuracy: 0.6656 - val_loss: 1.1613 - val_accuracy: 0.5646
Epoch 8/10
15/15 [=====] - 12s 804ms/step - loss: 0.8842 - accuracy: 0.6641 - val_loss: 1.1802 - val_accuracy: 0.5417
Epoch 9/10
15/15 [=====] - 12s 822ms/step - loss: 0.8544 - accuracy: 0.6687 - val_loss: 1.1532 - val_accuracy: 0.5688
Epoch 10/10
15/15 [=====] - 12s 820ms/step - loss: 0.8186 - accuracy: 0.6969 - val_loss: 1.1467 - val_accuracy: 0.5813
```


Accuracy & Loss



Predictions



Visual Studio Code interface showing a Python project for video processing and object detection.

EXPLORER: NO FOLDER OPENED

OUTLINE: categories, bg, run_avg, segment, loadModel, predict, capture_image

segment.py - Visual Studio Code

```
112 # so that our running average model gets calibrated
113 if num_frames < 30:
114     run_avg(gray, aweight)
115 else:
116     # segment the food region
117     food = segment(gray, 25)
118
119 # check whether food region is segmented
120 if food is not None:
121     # if yes, unpack the thresholded image and
122     # segmented region
123     (thresholded, segmented) = food
124
125 # draw the segmented region and display the frame
126 cv2.drawContours(clone, [segmented + (right, top),
127 cv2.imshow("Thresholded", thresholded)
128
129 if(num_frames%100 == 0):
130
131     print("food type")
132     print(type(thresholded))
133     print(thresholded.shape)
134     print(type(food))
135
136     print("capturing the food region")
137     filename1 = 'capturedImageAtFrame-' + str(num_frames) + '.jpg'
```

Video Feed: A video frame showing a person holding a lemon. The lemon is highlighted with a green bounding box, and a red contour is drawn on its surface.

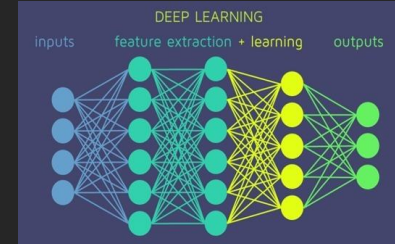
TERMINAL:

```
2--Cherry
food type
<class 'numpy.ndarray'>
(215, 240)
<class 'tuple'>
capturing the food region
<class 'numpy.ndarray'>
predicting6300
3
3--Lemon
food type
```

STATUS BAR: Ln 116, Col 38 Spaces: 4 UTF-8 CRLF Python 3.10.2 64-bit

Taskbar: Windows Start button, Search bar, Task View, and various application icons (File Explorer, Chrome, etc.). System tray shows 30°C, 10:12 PM, 08-05-2022.

Predictions



```
File Edit Selection View Go Run Terminal Help
segment.py - Visual Studio Code

EXPLORER
  NO FOLDER OPENED
  OUTLINE
    categories
    bg
    run_avg
    segment
    loadModel
    predict
    capture_image

G: > UMKC > Python Project > segment.py > ...
3 import imutils
4 import numpy as np
5 import keras
6
7
8 categories = {
9     0: 'Apple',
10    1: 'Banana',
11    2: 'Cherry',
12    3: 'Lemon',
13    4: 'Pineapple',
14    5: 'Tomato'
15 }
16
17 # global variables
18 bg = None
19
20 #-----
21 # To find the running average over the background
22 #-----
23 def run_avg(image, aWeight):
24     global bg
25     # initialize the background
26     if bg is None:
27         bg = image.copy().astype("float")
28     return
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Video Feed

The screenshot shows a video feed window titled 'Video Feed'. It displays a person holding a small, dark object (likely a cherry) in their hand. A green bounding box is drawn around the object, indicating the predicted region of interest. The background is a plain, light-colored wall.

TERMINAL

```
2--Cherry
food type
<class 'numpy.ndarray'>
(215, 240)
<class 'tuple'>
capturing the food region
<class 'numpy.ndarray'>
predicting3800
2
2--Cherry
```

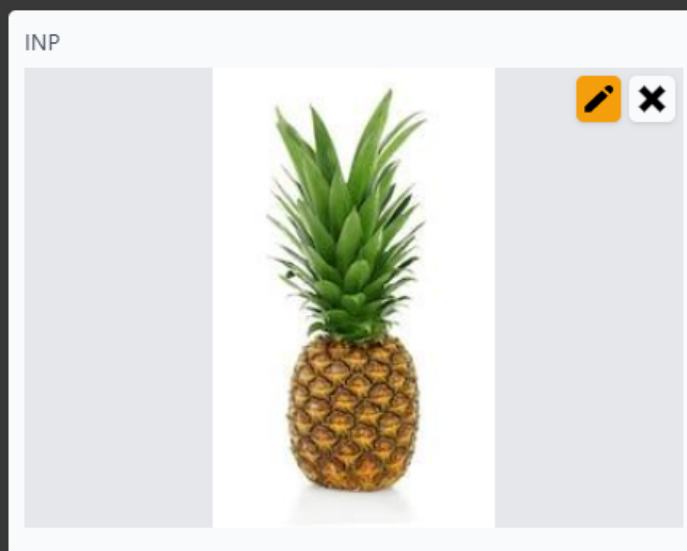
Ln 165, Col 16 Spaces: 4 UTF-8 CRLF Python 3.10.2 64-bit

09:34 PM 08-05-2022

Interface

Colab notebook detected. To show errors in colab notebook, set ``debug=True`` in ``launch()``
Running on public URL: <https://55319.gradio.app>

This share link expires in 72 hours. For free permanent hosting, check out Spaces (<https://huggingface.co/spaces>)



Clear

Submit

OUTPUT

0.1s

Pineapple - 452 cal

Pineapple - 452 cal

100%

Apple - 95 cal

0%

Banana - 105 cal

0%

Flag

Future Scope

Now that we are able to predict the food given by the end user using our application, we can extend the functionality so to predicting multiple types of food in a single image, calculate quantity, derive nutrition tables etc.



Thank you!! Any Questions???