greatlearning
Power Ahead

Great Learning

AIML

CAPSTONE PROJECT

COMPUTER VISION

OBJECT DETECTION/CLASSIFICATION - FOOD

Milestone 1 – Interim Report

The CV-3 Team

Batch details	AIML Online June 24 A
Team members	Poongudivanan Natarajan Anand Singh Prabhu Arumugam Rajesh kumar Dash Ranganath Deshpande
Domain of Project	Food Industry
Proposed project title	OBJECT DETECTION/CLASSI-FICATION - FOOD
Group Number	Group 5 - CV 3 - Food 101 De-tection Project
Mentor Name	Jyant Mahara

Table of Contents – Milestone 1 Interim Report

1) Introduction

1.1 Business Understanding

In industries such as food service, manufacturing, and retail, maintaining quality control and process compliance requires constant visual supervision. Traditionally, this has been done manually, which can be labour-intensive, inconsistent, and prone to human error.

With advancements in computer vision, visual tasks such as identifying food items, assessing their appearance, and detecting anomalies can be automated using image data. For example, cameras can recognize food types, check for correct ingredients, evaluate food colour and texture, and ensure presentation standards are met. If the system predicts an undesirable event (e.g., incorrect item or poor quality), it can trigger automated actions such as alerts or rejections.

This not only enhances accuracy and consistency but also significantly reduces supervision costs and operational inefficiencies.

The current solution automates the identification and classification of food images using AI techniques.

1.2 Business Objective

To design and implement an **automated computer vision system** that:

- Identifies food items from images or video feeds.
- Analyzes visual characteristics (type, color, ingredients, presentation).
- Predicts potential quality issues or mismatches.

- Triggers predefined actions (e.g., alerts, workflow changes) in response to predicted events.
- Improves quality control, reduces human supervision needs, and increases overall process efficiency.

2) Summary of Problem Statement, Data, and Findings

2.1 Business Problem Summary

The current solution automates the identification and classification of food images. This solution enhances accuracy and consistency but also significantly reduces supervision costs and operational inefficiencies.

2.2 Dataset Description

The given dataset contains 16256 images of 17 different types of food. The images are in JPG format with moderate quality of capture. The input is set of unclassified images.

2.3 Key Observations and Initial Findings

The **Food101** dataset contains a total of **16,256 images** categorized into **17 different food classes**. Each class represents a type of food, such as apple-pie, pizza, samosa, etc. Each food category has approximately **1,000 images**, except for apple_pie, which has 257 samples.

3) Exploratory Data Analysis (EDA) and Preprocessing

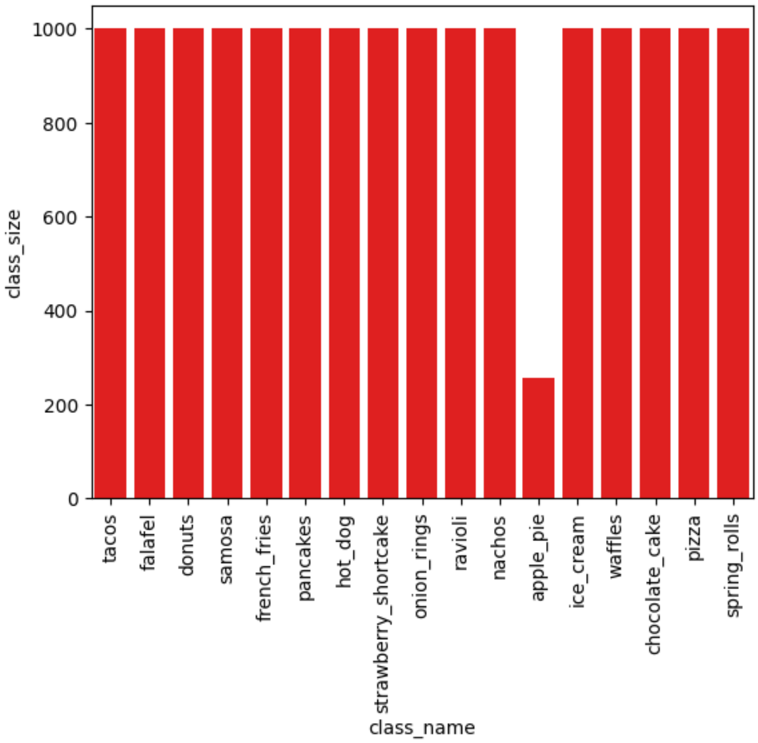
3.1 EDA Goals and Approach

The EDA was done to understand the classification, distribution and bias of different food classes. The classification and count of different food classes are outputted in below table.

- **Count of 17 food classes:**

	class_size
class_name	
tacos	1000
falafel	1000
donuts	1000
samosa	1000
french_fries	1000
pancakes	1000
hot_dog	1000
strawberry_shortcake	1000
onion_rings	1000
ravioli	1000
nachos	1000
apple_pie	257
ice_cream	1000
waffles	1000
chocolate_cake	1000
pizza	1000
spring_rolls	1000

- **Bar Plot of the food classification:**



➢ **Summary of Data Observation**

- There are in total **17 classes** of food items
- **16 classes** of food items have **1000 images** each.
- The Apple Pie class of food items has **257 images**
- This is a well-balanced data set. There is a uniform distribution across 16 classes.
- The Apple Pie class has 25% of images compared to other classes. This will afflict the classification of Apple Pie images.
- To address the class imbalance with Apple Pie, following mitigations are recommended.
    - **Data Augmentation**: Apply augmentation (rotation, flipping, color jittering) to synthetically increase the number of `apple_pie` images.
    - **Class Weights**: Use class weights in the loss function during training to emphasize learning on the minority class.
    - **SMOTE or Oversampling Techniques** (if suitable for images).
    - **Undersampling** other classes (less preferred due to loss of valuable data).
    - **Visual inspection** that highlight the region of interest of apple_pie images may help understand why the count is low (e.g., ambiguous visuals, data collection issue).

3.2 Data Visualization and Insights

The visual inspection of sample data is done with the bounding box annotations. This helps in highlighting the region of interest and accurate classification of food items.

➢ Purpose of this Visualization

- To visually verify the quality and accuracy of bounding box annotations.
- To understand how localized regions differ across classes and image samples.
- To evaluate whether bounding box information can aid in better feature learning for classification or detection tasks.

➢ Dataset annotation overview

How We Annotated the Data

We used a hybrid approach combining **automated tools** and **AI assistance** to efficiently generate high-quality annotations:

- **YOLOv8 and YOLOv11**: Employed for automatic object detection and initial bounding box generation.Annotation for classes like Pizza,Cake,Donuts, Hot dog can be generated by Yolo.
- **ChatGPT API**: Used to **pass images directly** and **receive bounding box predictions** for objects, especially helpful in cases requiring semantic understanding or where pre trained models struggled.
- **Roboflow**: Used as a platform for annotation review, correction, and dataset management.It helps where ever Yolo model failed to detect the food classes.

We have successfully annotated our image dataset and stored the results in a file named `annotation_csv`. This file adheres to the **YOLO format**, with the following columns:

- **`image_name`**: Name of the image file.
- **`class_name`**: Object category label.
- **`x_center`**, **`y_center`**, **`width`**, **`height`**: Bounding box coordinates **normalized to the [0, 1] range**.

The annotated data is at google collab drive.

/content/drive/MyDrive/Python Course_shared/computer Vision/annotation/Food-101-Annotated-V3.zip

The annotation file (annotation_refined.csv) contains important metadata such as image file names, bounding box coordinates, and corresponding food class labels. The annotation.csv will be used for:

- Data Preparation: Parsing annotations in YOLO format for image-label mapping.
- Model Training: Feeding normalized bounding boxes and class labels into the training pipeline.

We extract and display the list of **unique food classes** present in the dataset to understand the classification scope.

- ## Annotations Sample data

Sample of Annotations Data:

	image	class_name	x_center	y_center	width	height
0	1199754.jpg	french_fries	0.509375	0.522656	0.720313	0.771875
1	1232631.jpg	nachos	0.178125	0.717969	0.351562	0.439063
2	1232631.jpg	nachos	0.522656	0.511719	0.932813	0.954688
3	2616112.jpg	nachos	0.165625	0.526563	0.214844	0.465625
4	2616112.jpg	nachos	0.394531	0.482812	0.171875	0.379688

- ## Annotations data frame info

```
Annotations DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1090 entries, 0 to 1089
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   image       1090 non-null   object
 1   class_name  1090 non-null   object
 2   x_center    1090 non-null   float64
 3   y_center    1090 non-null   float64
 4   width       1090 non-null   float64
 5   height      1090 non-null   float64
dtypes: float64(4), object(2)
memory usage: 51.2+ KB
```

- ## Annotations class distribution

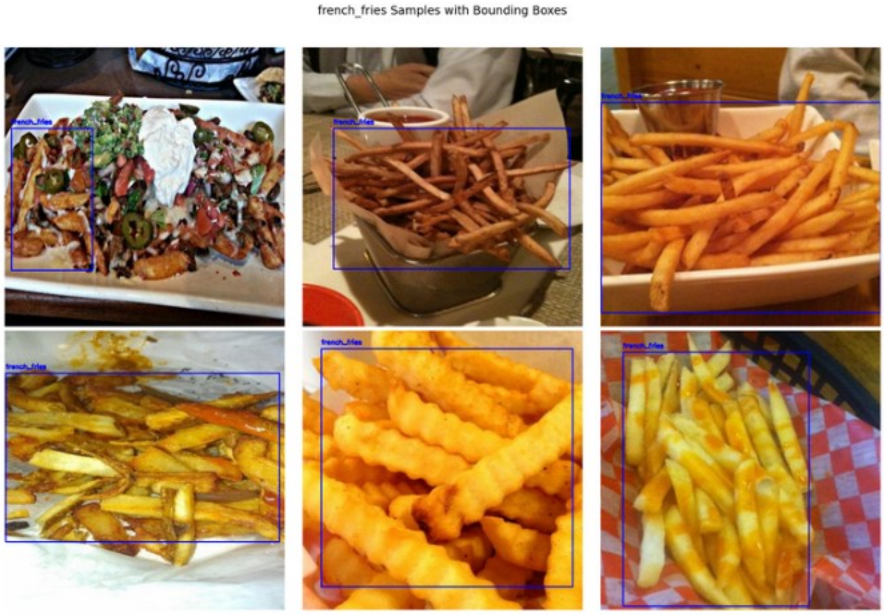
Class Distribution:

class_name	count
samosa	193
onion_ring	176
nachos	143
ice_cream	125
tacos	117
french_fries	81
pizza	68
strawberry_shortcake	67
waffle	65
chocolate_cake	55

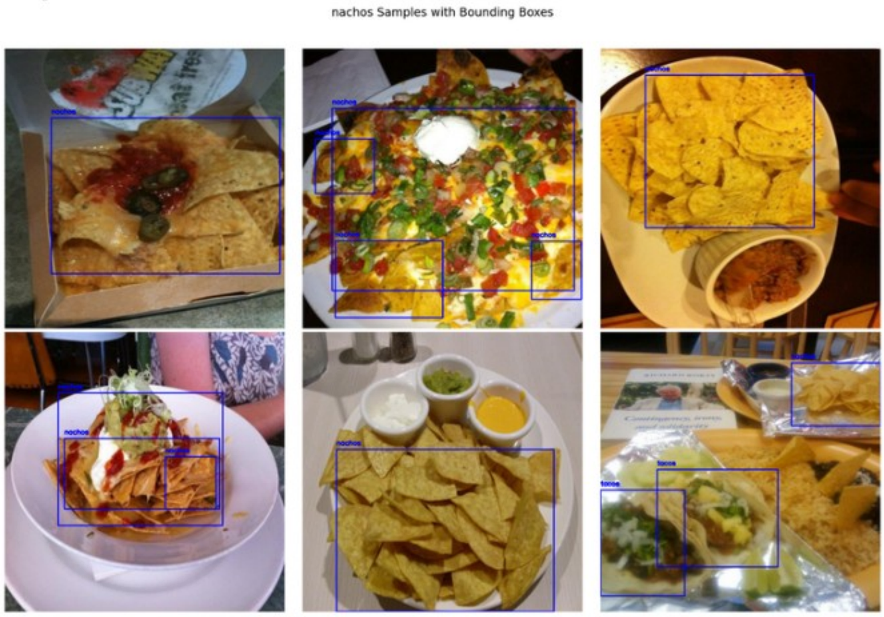
Class Distribution:

	x_center	y_center	width	height
count	1090.000000	1090.000000	1090.000000	1090.000000
mean	0.503608	0.485872	0.533300	0.502740
std	0.174767	0.152479	0.257276	0.234620
min	0.067187	0.102344	0.001563	0.002344
25%	0.397656	0.392383	0.327539	0.314844
50%	0.503125	0.492188	0.487109	0.463672
75%	0.603125	0.572656	0.753906	0.683594
max	0.953125	0.920312	1.000000	1.000000

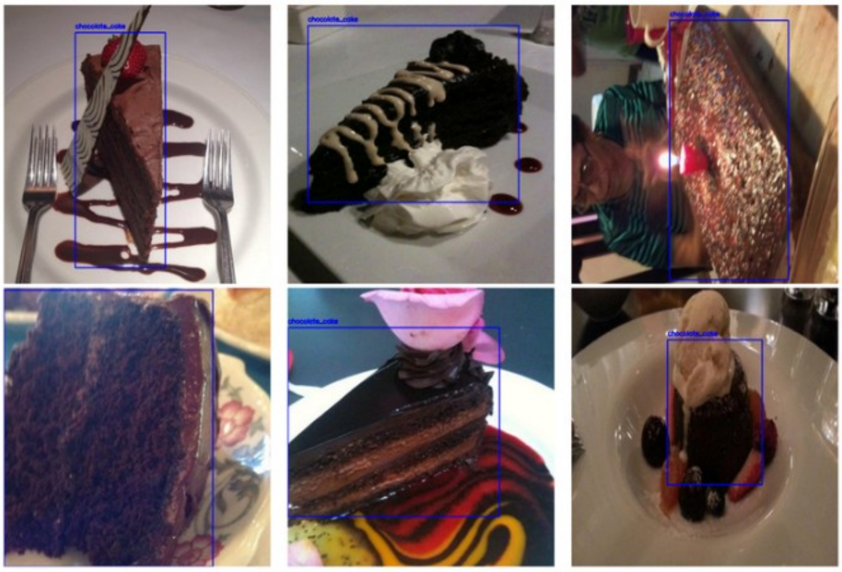
- Visualising classes of data with bounding boxes



Processing class: nachos

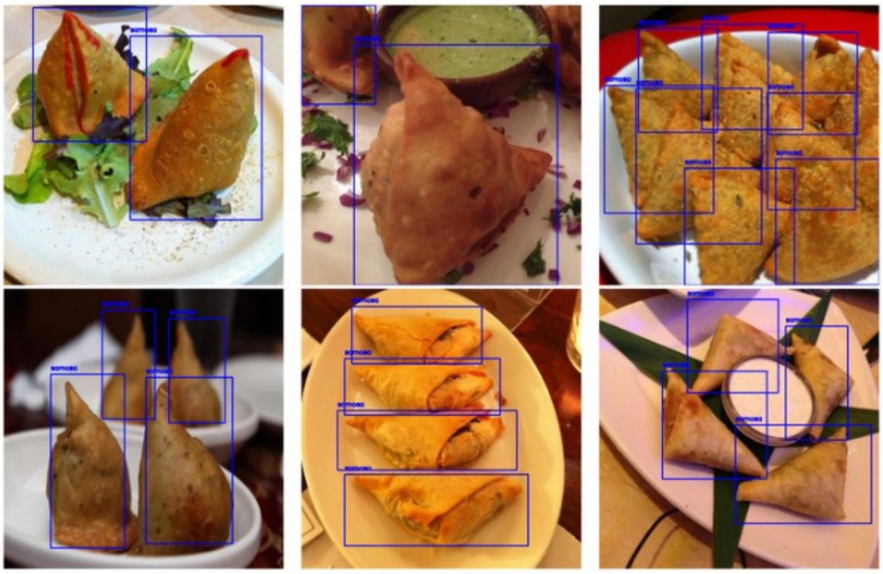


chocolate_cake Samples with Bounding Boxes



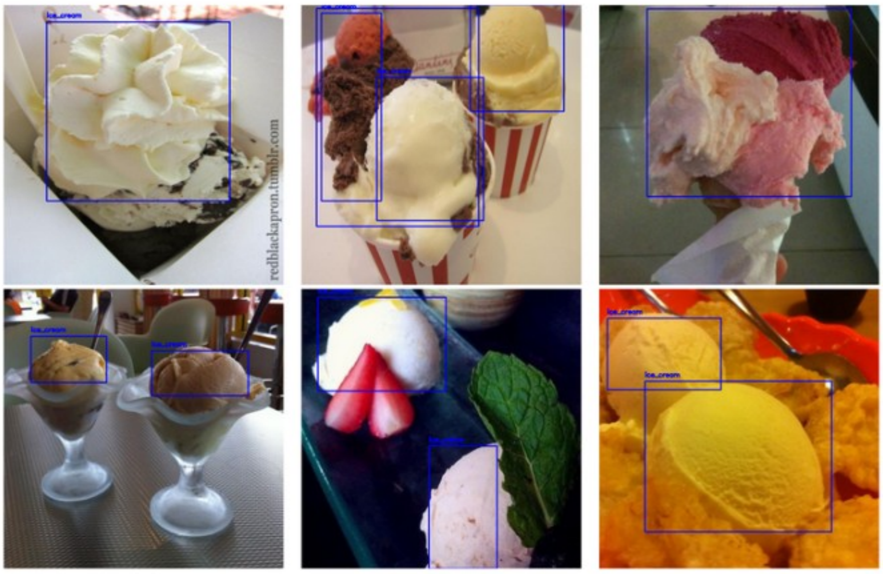
Processing class: samosa

samosa Samples with Bounding Boxes

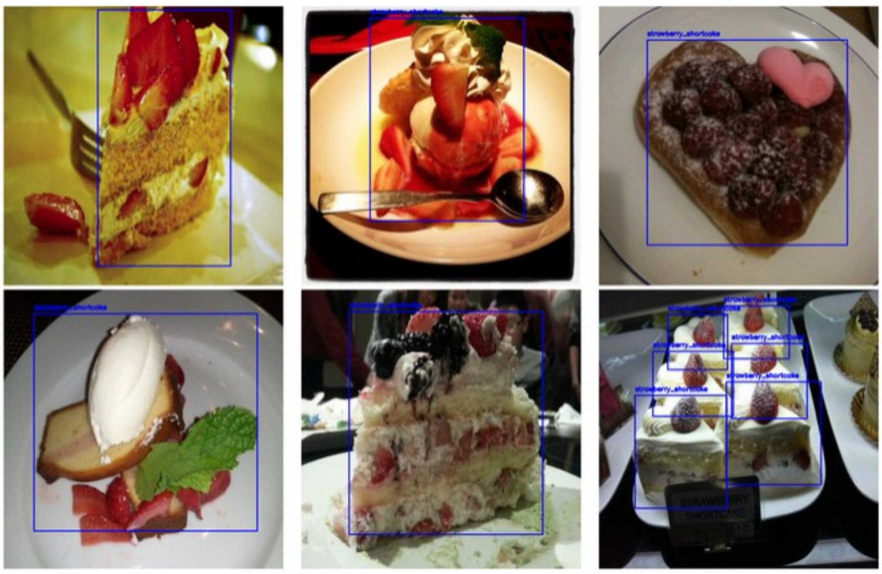


Processing class: ice_cream

ice_cream Samples with Bounding Boxes

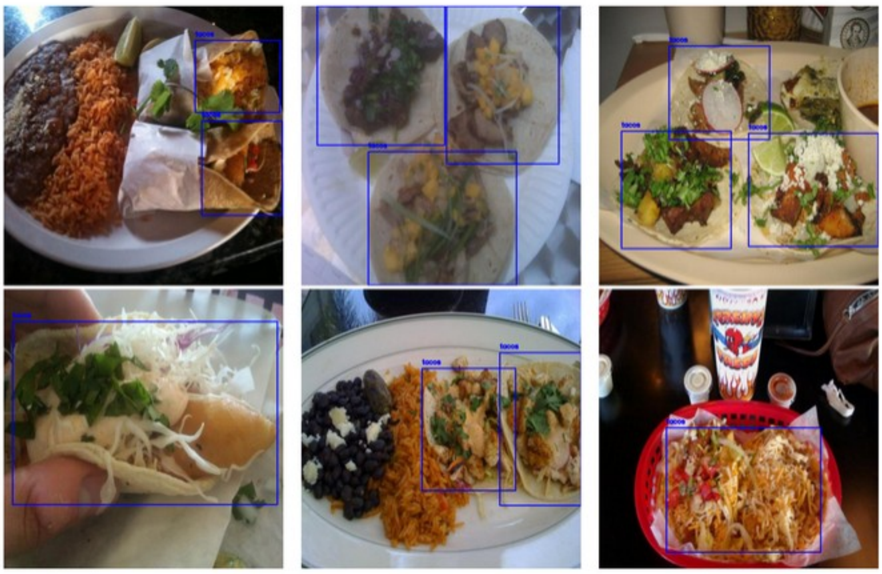


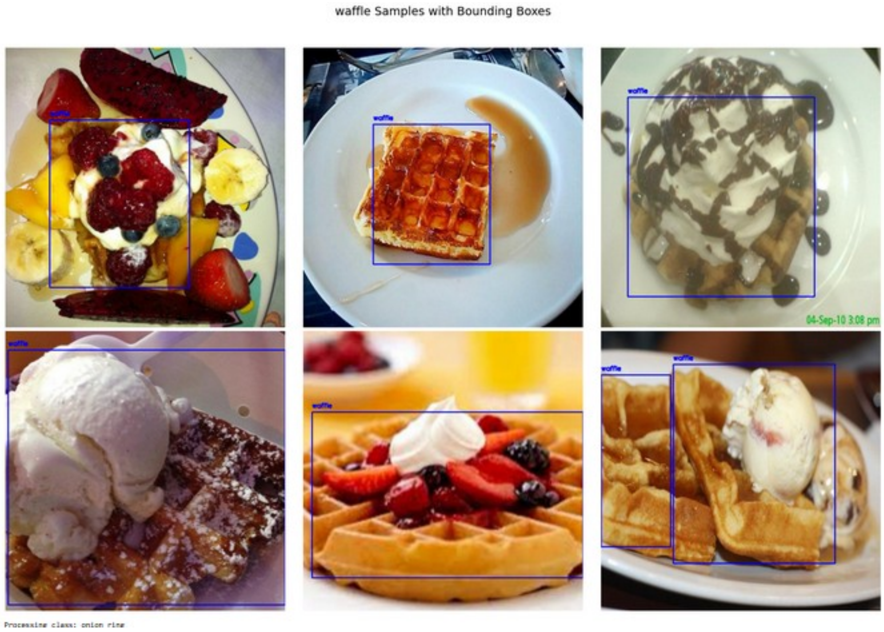
strawberry_shortcake Samples with Bounding Boxes

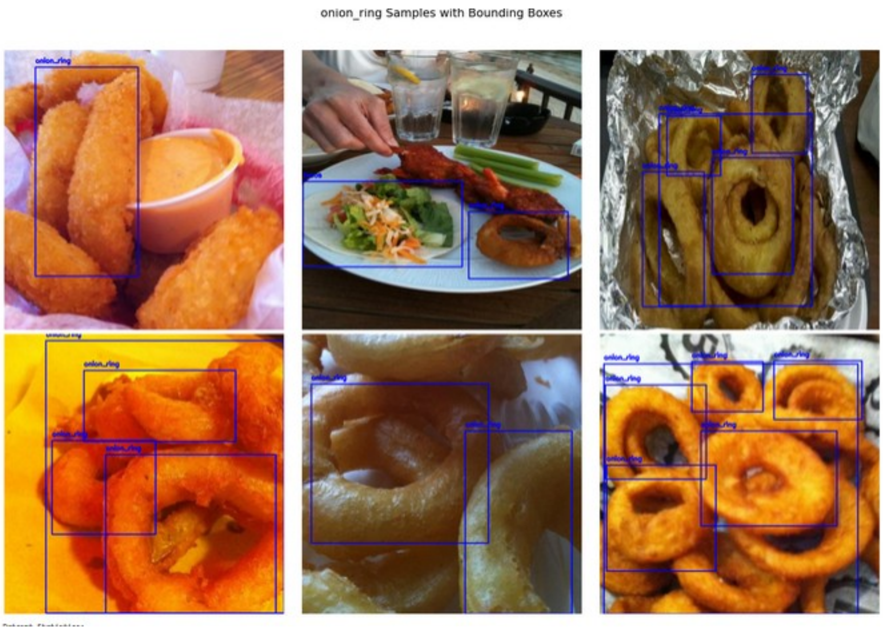


pizza Samples with Bounding Boxes

tacos Samples with Bounding Boxes







3.3 Data Cleaning and Preprocessing Techniques

We have annotated the datset and created a seperate annotation_csv file which contains imagename, classname and xcenter, ycenter, width, height (yolo format) all normalized 0 to 1.

We will be using the same dataset for the model data preparation and training.

- **Train and Test Split with Image Genrator**

Train Test Split With Image Generator

```
# Take first image only as there might have duplicate entryfor same image as same image mutiple instances of same food item
classification_df = annotation_df_model.groupby('image').first().reset_index()
# create a new volumn file name by concating the classname this will help us in using df for image data generator
classification_df['image_name'] = classification_df.apply(lambda row: f"{row['class_name']}/{row['image']}", axis=1)
# train test split with dividing the class in same propertion
train_df, temp_df = train_test_split(
    classification_df,
    test_size=0.2,
    stratify=classification_df['class_name'],  # ensures class distribution is preserved
    random_state=42)
valid_df, test_df = train_test_split(
    temp_df,
    test_size=0.5,
    stratify=temp_df['class_name'],  # ensures class distribution is preserved
    random_state=42)
```

```
print(f"Training Set ->{train_df.shape}",f"Validation Set ->{valid_df.shape}",f"Test Set ->{test_df.shape}")
```

```
Training Set ->(447, 7) Validation Set ->(56, 7) Test Set ->(56, 7)
```

	class_name	total_count	train_count
0	chocolate_cake	51	41
1	french_fries	78	62
2	ice_cream	54	43
3	nachos	48	38
4	onion_ring	50	40
5	pizza	58	46
6	samosa	71	57
7	strawberry_shortcake	50	40
8	tacos	52	42
9	waffle	47	38

- We can see we have 447 for training 56 for testing and 56 will be for validation
- We can see each class is equally distributed 80% of total image set for the class from the dataframe

➢ **Defining the image data generator for train and test data validation**

```python
train_datagen =ImageDataGenerator(
                horizontal_flip=True,
                vertical_flip=True,
                rotation_range=15,
                width_shift_range=0.1,
                height_shift_range=0.1,
                zoom_range=0.2,
                shear_range=0.1,
                fill_mode='nearest',
                rescale=1/255) #rescale to [0-1], add zoom range of 0.2x and horizontal flip
test_valid_datagen = ImageDataGenerator(rescale=1./255)


# Create training image gen
train_gen = train_datagen.flow_from_dataframe(
    train_df,
    directory=base_path,  # base path
    x_col='image_name',
    y_col='class_name',
    target_size=(128, 128),
    class_mode='categorical',
    batch_size=32,
    shuffle=True
)

# Create test image gen
test_gen = test_valid_datagen.flow_from_dataframe(
    test_df,
    directory=base_path,
    x_col='image_name',
    y_col='class_name',
    target_size=(128, 128),
    class_mode='categorical',
    batch_size=32,
    shuffle=False
)
# Create valid image gen

valid_gen = test_valid_datagen.flow_from_dataframe(
    valid_df,
    directory=base_path,
    x_col='image_name',
    y_col='class_name',
    target_size=(128, 128),
    class_mode='categorical',
    batch_size=32,
    shuffle=False
)
```

➢ Observations:

Found 446 validated image filenames belonging to 10 classes.

Found 56 validated image filenames belonging to 10 classes.

Found 56 validated image filenames belonging to 10 classes.

```python
print(train_gen.class_indices)
print(valid_gen.class_indices)
print(test_gen.class_indices)
```

```
{'chocolate_cake': 0, 'french_fries': 1, 'ice_cream': 2, 'nachos': 3, 'onion_ring': 4, 'pizza': 5, 'samosa': 6, 'strawberry_shortcake': 7, 'tacos': 8, 'waffle': 9}
{'chocolate_cake': 0, 'french_fries': 1, 'ice_cream': 2, 'nachos': 3, 'onion_ring': 4, 'pizza': 5, 'samosa': 6, 'strawberry_shortcake': 7, 'tacos': 8, 'waffle': 9}
{'chocolate_cake': 0, 'french_fries': 1, 'ice_cream': 2, 'nachos': 3, 'onion_ring': 4, 'pizza': 5, 'samosa': 6, 'strawberry_shortcake': 7, 'tacos': 8, 'waffle': 9}
```

➢ Class Counts:

```python
from collections import Counter

# Count how many instances per class
class_counts = Counter(test_gen.classes)

# Map class indices back to names
index_to_class = {v: k for k, v in test_gen.class_indices.items()}
class_distribution = {index_to_class[i]: count for i, count in class_counts.items()}

# Print result
for class_name, count in class_distribution.items():
    print(f"{class_name}: {count}")
```

```
ice_cream: 6
french_fries: 8
chocolate_cake: 5
onion_ring: 5
strawberry_shortcake: 5
samosa: 7
tacos: 5
nachos: 5
pizza: 6
waffle: 4
```

- **Train and Test Split with Image Genrator**

```python
# Convert lists to arrays
X = np.array(images)
y = np.array(labels)

# Encode string labels to integers
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
y_categorical = to_categorical(y_encoded)  # one-hot encoding

# Train-test split
X_train, X_temp, y_train, y_temp = train_test_split(X, y_categorical, test_size=0.2, random_state=42, stratify=y_categorical)
X_valid, X_test, y_valid, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42, stratify=y_temp)

# Shapes
print(f"Train: {X_train.shape}, {y_train.shape}")
print(f"Valid: {X_valid.shape}, {y_valid.shape}")
print(f"Test: {X_test.shape}, {y_test.shape}")
```

```
Train: (452, 128, 128, 3), (452, 10)
Valid: (56, 128, 128, 3), (56, 10)
Test: (57, 128, 128, 3), (57, 10)
```

5) Model Performance Improvement Strategy

5.1 Identified Challenges

We tried with the Train/Test set created without Image generator.

Model: "sequential"

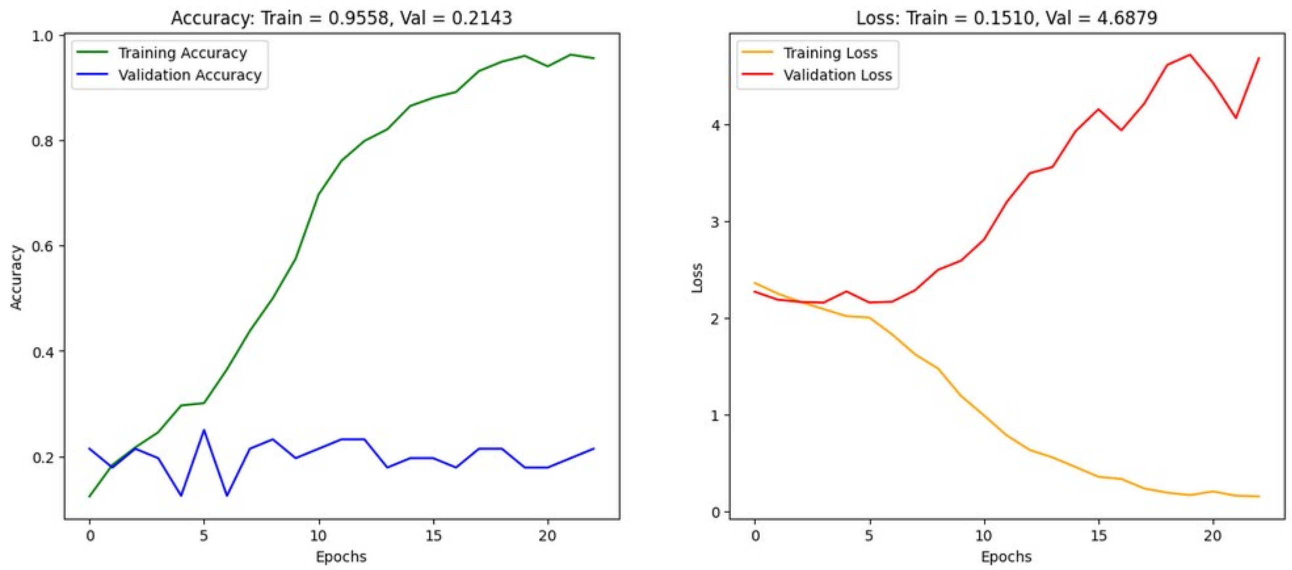
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 128)	3,211,392
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

Total params: 3,305,930 (12.61 MB)
Trainable params: 3,305,930 (12.61 MB)
Non-trainable params: 0 (0.00 B)

➢ **MODEL ACCURACY**

Model is overfitting

```
plot_model_accuracy(history,cnn_model)
```



Conclusion:
The training and validation curve is diverging. This model is **NOT** a good for further analysis

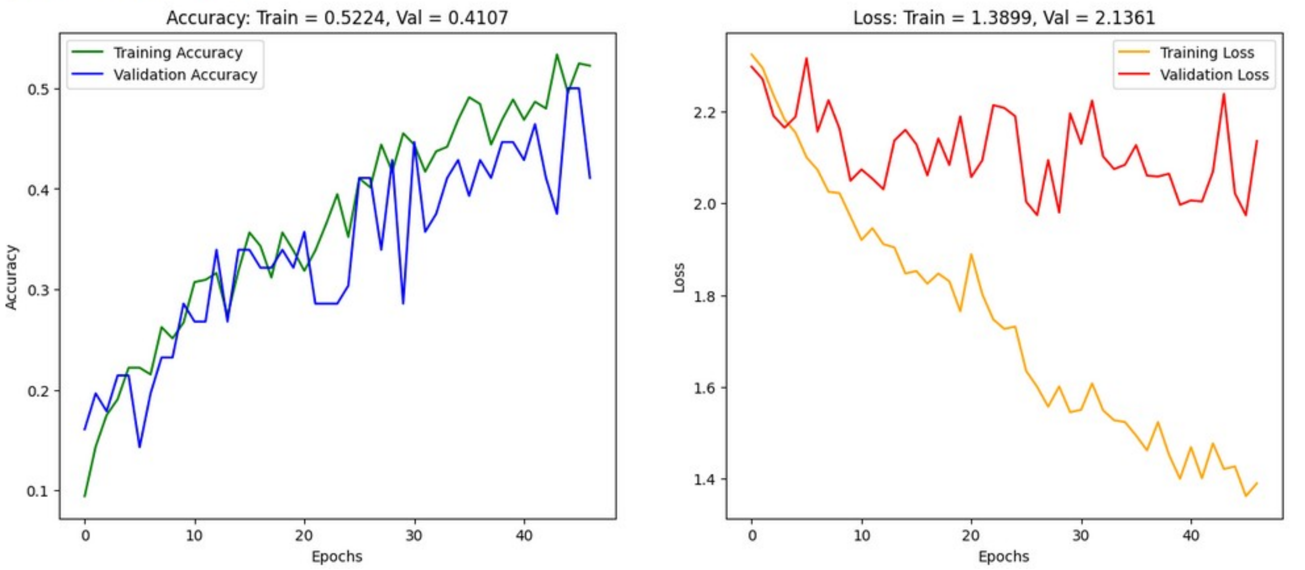
5.2 IMPROVEMENT TECHNIQUES (E.G., DATA AUGMENTATION, HYPERPARAMETER TUNING)

a. Model Creation with train/test data enhanced with image generator and balance train set

As an improvement we tried model building using TRAIN/Test split with Image generator and balanced train set (ADAM). Observations as below.

- ➢ **MODEL ACCURACY**

```
plot_model_accuracy(history,model_2)
```



- Higher noise observed in training and validation
- We can see its going up an down significantly and model training accuracy of 52% tells that model is failing to learn nuance features

```
model_2.evaluate(test_gen)
```

Observations:

- Model failed to generalise as training accuracy is around 52% with validation accuracy is at 41%.
- Test acccuracy is around 50%
- We can see both validation and train curve has noises .

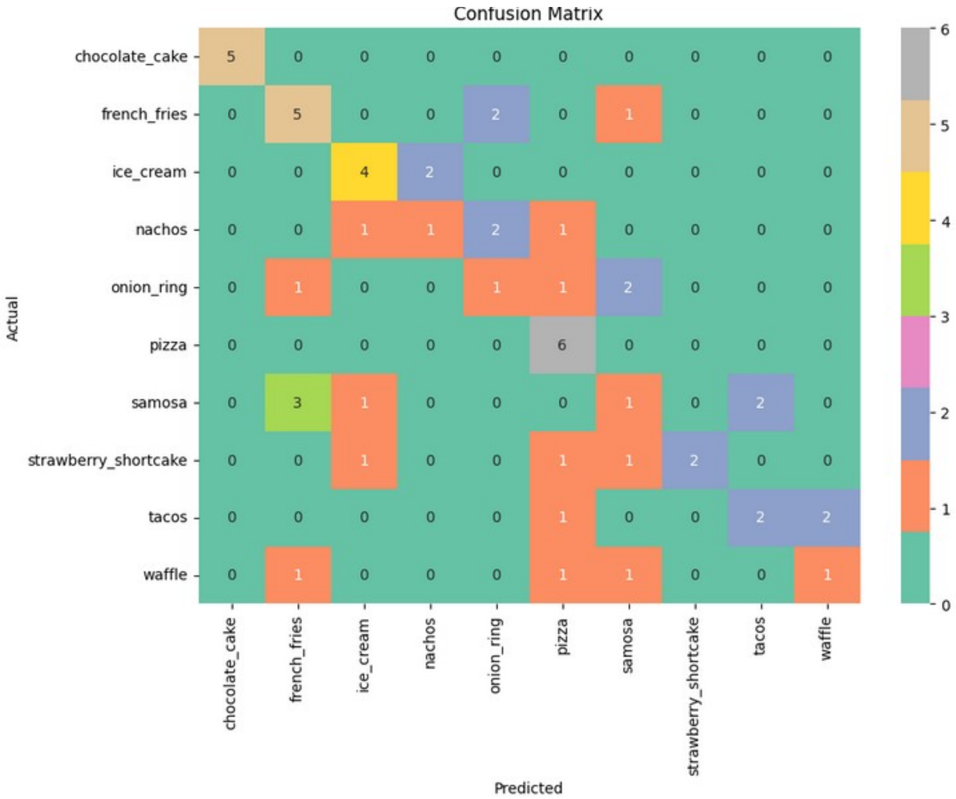
```
# SHow classfication report
generate_classification_report(model_2, test_gen)
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 309ms/step
                    precision    recall  f1-score   support

    chocolate_cake       1.00      1.00      1.00         5
      french_fries       0.50      0.62      0.56         8
         ice_cream       0.57      0.67      0.62         6
            nachos       0.33      0.20      0.25         5
        onion_ring       0.20      0.20      0.20         5
             pizza       0.55      1.00      0.71         6
            samosa       0.17      0.14      0.15         7
strawberry_shortcake     1.00      0.40      0.57         5
             tacos       0.50      0.40      0.44         5
            waffle       0.33      0.25      0.29         4

          accuracy                           0.50        56
         macro avg       0.52      0.49      0.48        56
      weighted avg       0.51      0.50      0.48        56
```

- ➢ **CONFUSION MATRIX**



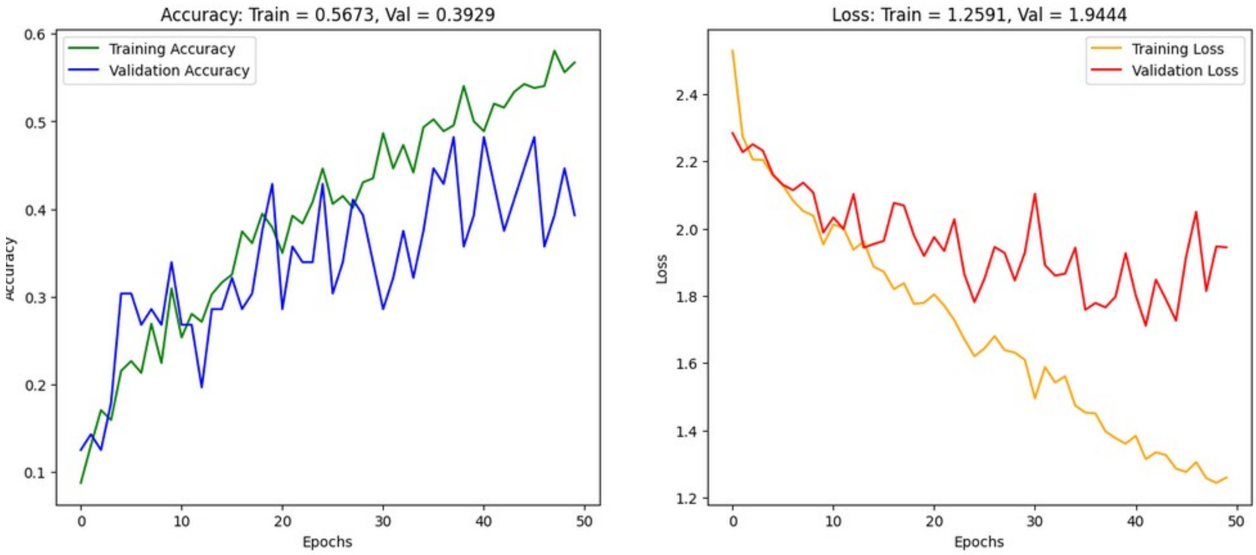
- **Model Summary**

  - **OVERALL TEST ACCURACY: 50%**
  - **OVERALL ACCURACY** IS AT **52%**
  - **HIGH PERFORMING CLASSES: CHOCOLATE_CAKE, PIZZA, ICE_CREAM SHOW GOOD PRECISION AND RECALL.**
  - **POORLY LEARNED CLASSES: SAMOSA, ONION_RING, STRAWBERRY_SHORTCAKE — EITHER COMPLETELY MISCLASSIFIED OR LOW RECALL.**

b. Model Creation with the normal train/test data and more layers

➢ MODEL ACCURACY

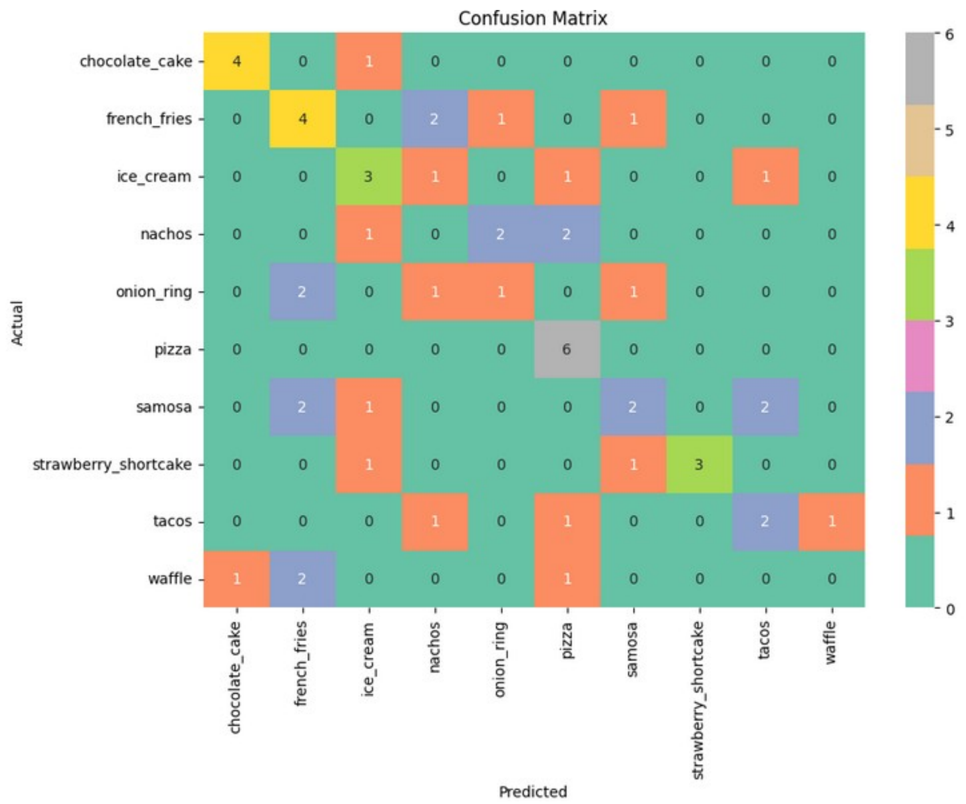
```
plot_model_accuracy(history,cnn_model_2)
```



➢ CLASSIFICATION REPORT

	precision	recall	f1-score	support
chocolate_cake	0.80	0.80	0.80	5
french_fries	0.40	0.50	0.44	8
ice_cream	0.43	0.50	0.46	6
nachos	0.00	0.00	0.00	5
onion_ring	0.25	0.20	0.22	5
pizza	0.55	1.00	0.71	6
samosa	0.40	0.29	0.33	7
strawberry_shortcake	1.00	0.60	0.75	5
tacos	0.40	0.40	0.40	5
waffle	0.00	0.00	0.00	4
accuracy			0.45	56
macro avg	0.42	0.43	0.41	56
weighted avg	0.43	0.45	0.42	56

➢ CONFUSION MATRIX

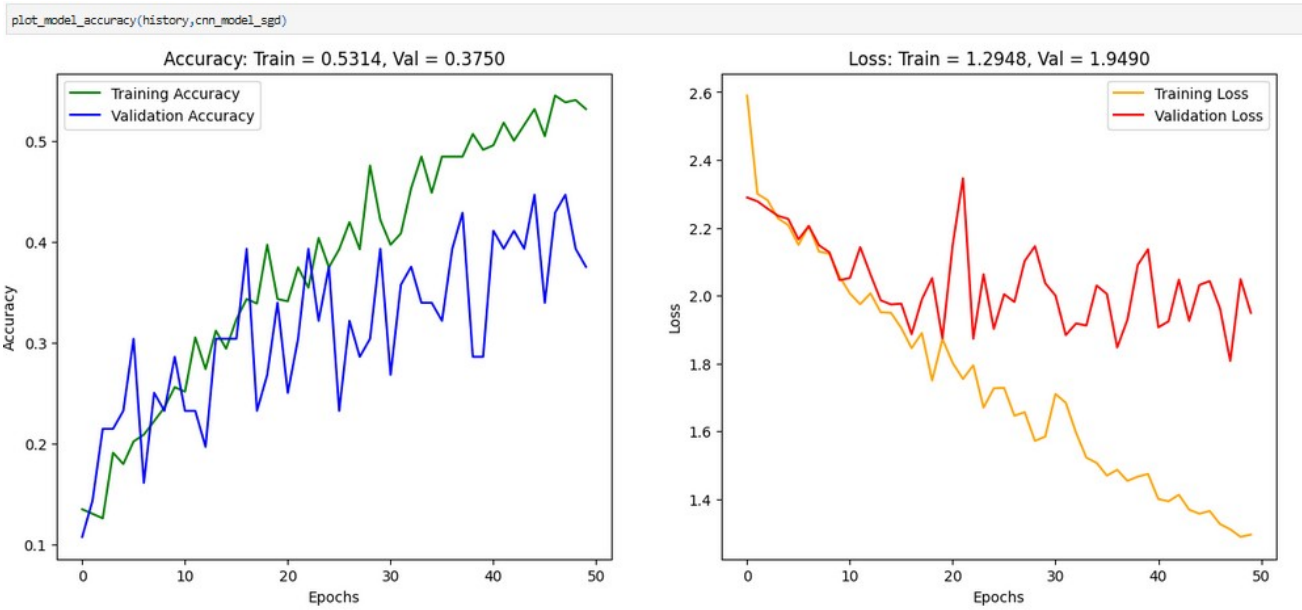


➢ MODEL SUMMARY

- **Test accuracy** is at 45%
- **Overall Accuracy**: 43% — lesser from previous model which is at 50% .
- **Well-Classified Classes**:
  - chocolate_cake, pizza, ice_cream: Good precision and recall.
- **Moderate Performance**:
  - french_fries, tacos, onion_ring: Acceptable but need improvement.
- **Poorly Classified Classes**:
  - samosa, nachos, waffle, strawberry_shortcake: Low recall or F1-score; waffle has 0% recall.

c. Model Creation with the normal train/test data and SGD

➢ MODEL ACCURACY

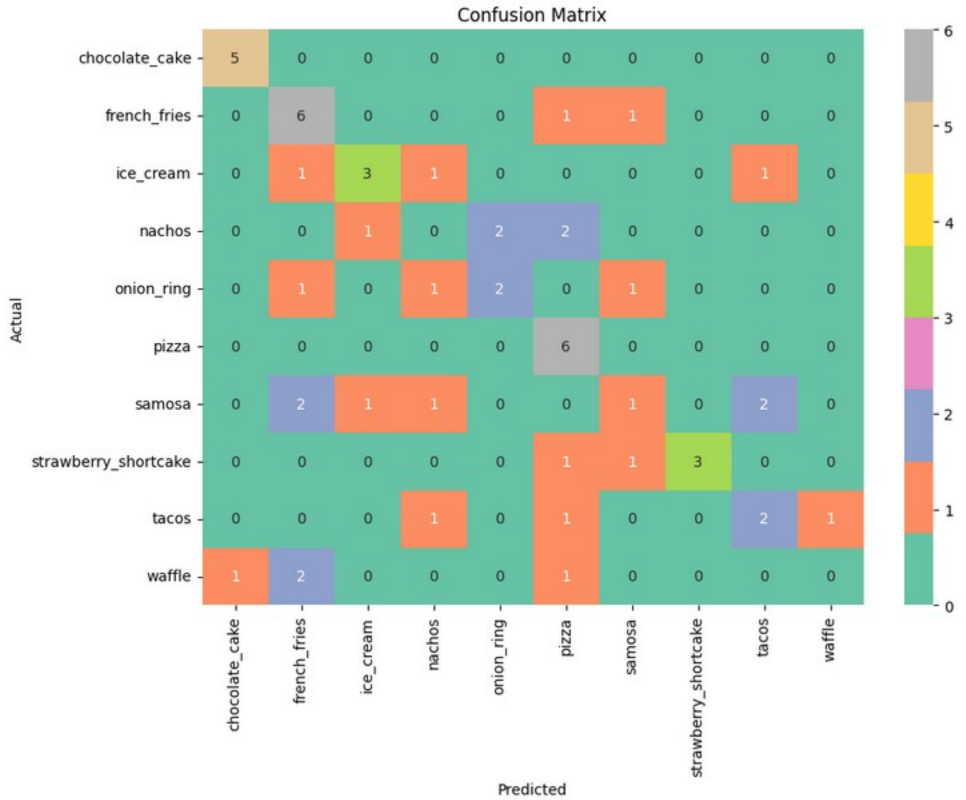


Not Improvement compared to previous models . SGD did not make the curve smooth. Though test score is same as first model but the training and validation score are lesser then first model.

➢ CLASSIFICATION REPORT

	precision	recall	f1-score	support
chocolate_cake	0.83	1.00	0.91	5
french_fries	0.50	0.75	0.60	8
ice_cream	0.60	0.50	0.55	6
nachos	0.00	0.00	0.00	5
onion_ring	0.50	0.40	0.44	5
pizza	0.50	1.00	0.67	6
samosa	0.25	0.14	0.18	7
strawberry_shortcake	1.00	0.60	0.75	5
tacos	0.40	0.40	0.40	5
waffle	0.00	0.00	0.00	4
accuracy			0.50	56
macro avg	0.46	0.48	0.45	56
weighted avg	0.46	0.50	0.46	56

➢ Confusion Matrix



➢ Model Summary

- **Test Accuracy**: 50%
- **Overall Accuracy**: 46% less then 1st model (without he_normal)
- **High Performing Classes**: chocolate_cake, pizza, ice_cream show good precision and recall.
- **Poorly Learned Classes**: samosa, onion_ring, strawberry_shortcake — either completely mis-classified or low recall.

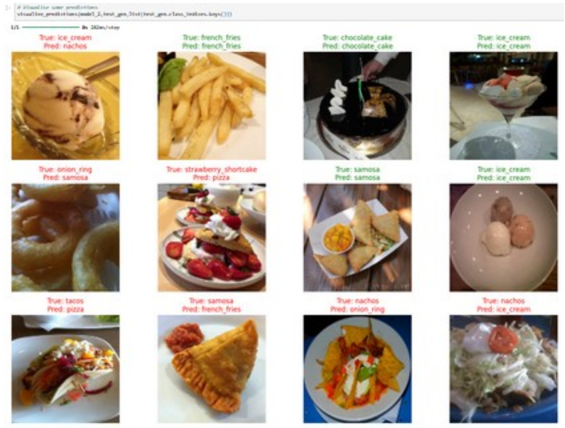
Model performances is almost similar to 1st cnn model without he_normal and sgd(used Adam)

➢ Class-wise observations

Class	Best Model (based on F1)	Comments
chocolate_cake	Model 3(SGD+he_normal) (0.91)	Excellent in all models.
french_fries	Model 3(SGD+he_normal)) (0.60)	Improved with SGD + He Normal.
ice_cream	Model 1(Adam) (0.62)	Slight drop in Model 3.
nachos	Model 1(Adam) (0.25)	Still poor in all.
onion_ring	Model 3(SGD+he_normal) (0.44)	He Normal helped.
pizza	Tie (1.00 recall in all)	Always predicted correctly.
samosa	Model 2(Adam+henormal) (0.33)	Still weak prediction.
strawberry_shortcake	Model 1 (0.57) / Model 2 & 3 (0.75)	Higher precision and recall with He Normal.
tacos	Tie (~0.40 F1 across)	Consistent poor performance.
waffle	Model 1(Adam) (0.29)	Missed completely in Model 2 & 3.

d. Predictions

➢ **Visualizing the prediction (Model-1 Default Initializer and Adam)**



➢ **Summary**

- Strongest classes: chocolate, cake, pizza, ice_cream, French Fries
- Performs better on visually distinct items.

Pitfalls:

- Fails on subtle or similar-looking items: samosa, waffle, nachos, and strawberry, short-cake.
- Some cases if there is some mayo like item in image it predict it as ice cream might be due to similarity with ice cream and mayo look.

5.3 Future Implementation Plan

➢ We can try adding more training data for weak classes.
➢ Use of data augmentation or fine-tuning with class weighting to improve class balance.
➢ Use transfer leraning from model like Efficientnet, mobilenet