Food Classification using Pattern Recognition

Deep Amish Shah
Dept of Mathematical Sciences
Stevens Institute of Technology
Jersey City, USA
dshah89@stevens.edu

Shreyas Reddy
Dept of Mathematical Sciences
Stevens Institute of Technology
Jersey City, USA
sreddy6@stevens.edu

Harikiran
Dept of Mathematical Sciences
Stevens Institute of Technology
Jersey City, USA
harora2@stevens.edu

Abstract

This project embarks on the creation of "Food Classification using Pattern Recognition" a cutting-edge convolutional neural network (CNN) model aimed at enhancing food image classification. Leveraging the extensive Food101 dataset, comprising a staggering 75,750 training images and 25,250 testing images. Through meticulous experimentation and refinement, our goal is to elevate the accuracy of food classification to unprecedented levels. By harnessing the power of TensorFlow's advanced modelling techniques and efficient data loading mechanisms, we endeavor to achieve groundbreaking results in the realm of food image analysis. Join us on this journey as we strive to redefine the boundaries of computer vision in the culinary domain.

1 Introduction

This project introduces "Food Classification using Pattern Recognition," a pioneering endeavor focusing on the development of a cutting-edge convolutional neural network (CNN) model aimed at advancing food image classification. Leveraging the vast Food101 dataset, which comprises an impressive 75,750 training images and 25,250 testing images, our primary objective is to surpass the benchmark established by the DeepFood paper, which achieved a top-1 accuracy of 77.4

2 Objective of this project

The objective of this project is to develop a robust deep learning-based food image recognition system designed to automatically identify and classify food items from photographs taken with mobile devices. Utilizing the comprehensive Food101 dataset, consisting of 75,750 training images and 25,250 testing images. By leveraging TensorFlow's advanced modelling techniques and efficient data loading mechanisms, we seek to revolutionize food image analysis and elevate the accuracy of food classification to unprecedented levels.

3 Tools and approach

- TensorFlow and Keras: Utilized for building and training the deep learning models due to their extensive libraries, ease of use, and support for rapid prototyping.

- Python: Chosen as the primary programming language for its robustness and the wealth of libraries available for data manipulation and machine learning.

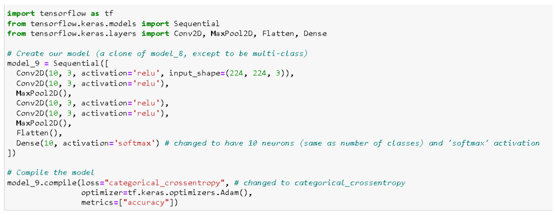
- OpenCV: Employed for image processing tasks, such as image resizing, normalization, and augmentation to enhance model training efficiency.

4 Workflow

The workflow depicted in the above image is a process for developing a machine learning model using TensorFlow, and it consists of the following six steps:

1. Get data ready (turn into tensors): The first step involves preparing the data for the model. This typically means collecting the necessary data, cleaning it, and then converting it into tensors, which are the primary data structure used in TensorFlow.

2. Build or pick a pretrained model (to suit your problem) from TensorFlow Hub: In this step, one either constructs a new model or selects an existing pretrained model from TensorFlow Hub that is appropriate for the specific problem at hand. TensorFlow Hub is a repository of trained machine learning models ready for fine-tuning and deployable anywhere.



```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense

# Create our model (a clone of model_8, except to be multi-class)
model_9 = Sequential([
  Conv2D(10, 3, activation='relu', input_shape=(224, 224, 3)),
  Conv2D(10, 3, activation='relu'),
  MaxPool2D(),
  Conv2D(10, 3, activation='relu'),
  Conv2D(10, 3, activation='relu'),
  MaxPool2D(),
  Flatten(),
  Dense(10, activation='softmax') # changed to have 10 neurons (same as number of classes) and 'softmax' activation
])

# Compile the model
model_9.compile(loss="categorical_crossentropy", # changed to categorical_crossentropy
                optimizer=tf.keras.optimizers.Adam(),
                metrics=["accuracy"])
```

Figure 1: Image 3

3. Fit the model to the data and make a prediction: Here, the selected or built model is trained (or 'fit') on the dataset. This involves feeding the data through the model so it can learn to make predictions.

4. Evaluate the model: After training, the model's performance is assessed. This evaluation is done using a separate set of data not seen by the model during training to gauge how well the model is likely to perform on new, unseen data.

5. Improve through experimentation: Based on the evaluation results, one might decide to make adjustments to the model, data, or training process. This step is iterative, with each cycle aimed at improving the model's predictive performance.

6. Save and reload your trained model: Finally, after obtaining a satisfactory model, it is saved for later use. This enables the model to be reloaded and used for predictions without needing to be retrained.

5 Outcomes and Accuracy

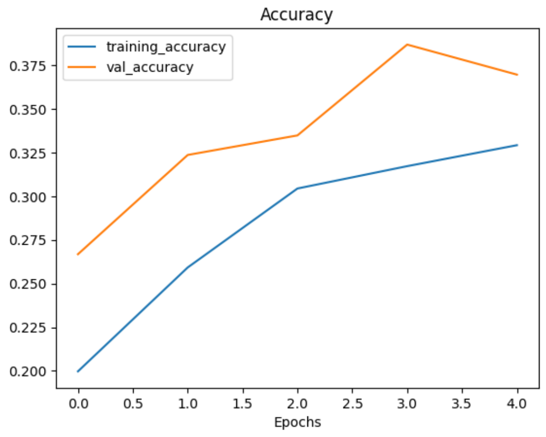


Figure 2: Accuracy graph

In the above image, we can observe the "Accuracy" graph, showing the training accuracy (blue line) and validation accuracy (orange line). Here, the training accuracy increases significantly over time, indicating that the model is getting better at making correct predictions on the training data. However, the validation accuracy starts to flatten after an initial increase. This implies that while the model is learning the training data well, it might not be improving as much on the unseen validation data—a sign that the model could be starting to overfit to the training data or that it's reaching its capability on the current validation set.

In the above image we can see the "Training and Validation Loss" graph, which depicts the model's loss on both the training set (blue line) and the validation set (orange line) over epochs. Loss is a

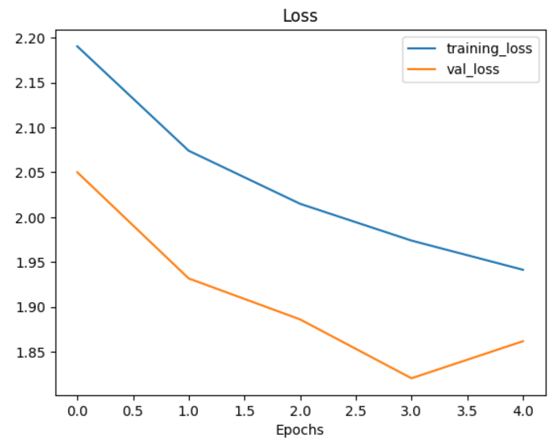


Figure 3: Training and Validation Loss graph

measure of how well the model is performing, with lower numbers indicating better performance. Initially, both losses decrease sharply, with the training loss continuing to decline steadily. The validation loss also decreases but appears to plateau, suggesting that the model is learning from the training data. The green vertical line indicates the point at which fine-tuning started, after which the validation loss decreases at a more moderate rate compared to the initial training phase. This suggests that fine-tuning the model on more specific data (or with adjusted hyperparameters) has potentially led to better generalization on the validation data.

6 Contributions of Each Group Member

[]tem **Hari Kiran:**[label=–]

- – Contributed to the project by collecting, validating, and meticulously preparing high-quality data. Executed thorough Exploratory Data Analysis (EDA) and diligently cleaned the dataset to ensure its integrity and readiness for further analysis.

- **Deep Shah:**

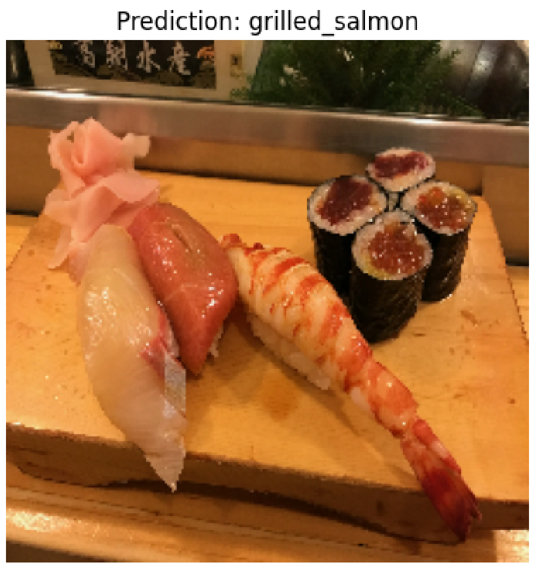


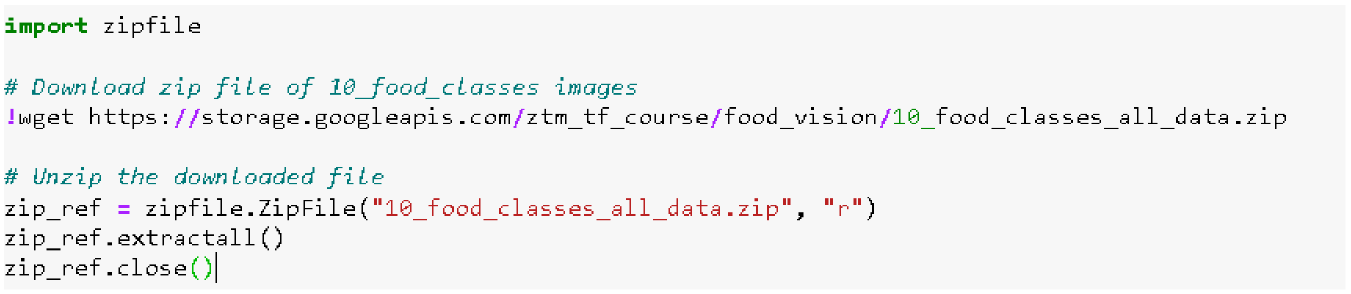
Figure 4: Image 3

[label=–]Contributed to the project by constructing a convolutional neural network (CNN) architecture tailored to the specific requirements of the task. Successfully trained the model to achieve high accuracy in classification tasks, with the added achievement of producing reliable probabilities for the predicted classes.

- **Shreyas Reddy:**

[label=–]Played a pivotal role in optimizing the codebase, results, and accuracy of the project. Conducted thorough analysis and implemented various optimizations to enhance the efficiency and performance of the system. Compiled and synthesized the findings into a comprehensive final report, detailing the project's methodology, results, and key insights garnered from the optimization process.

7 Screenshot

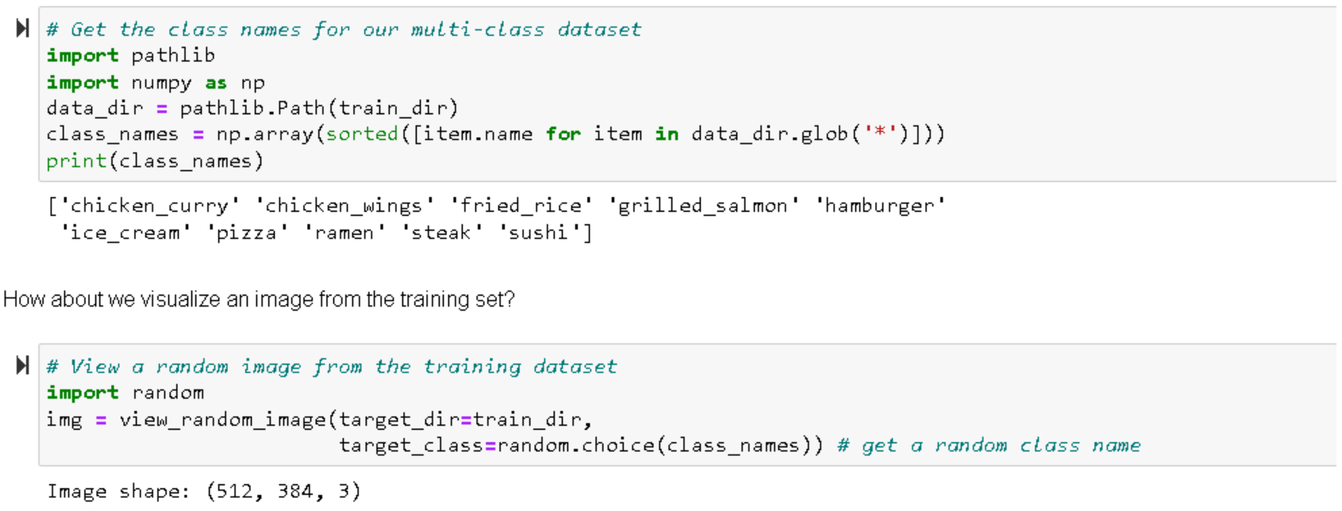


```
import zipfile

# Download zip file of 10_food_classes images
!wget https://storage.googleapis.com/ztm_tf_course/food_vision/10_food_classes_all_data.zip

# Unzip the downloaded file
zip_ref = zipfile.ZipFile("10_food_classes_all_data.zip", "r")
zip_ref.extractall()
zip_ref.close()
```

Figure 5: Image 5



```
# Get the class names for our multi-class dataset
import pathlib
import numpy as np
data_dir = pathlib.Path(train_dir)
class_names = np.array(sorted([item.name for item in data_dir.glob('*')]))
print(class_names)
```

```
['chicken_curry' 'chicken_wings' 'fried_rice' 'grilled_salmon' 'hamburger'
 'ice_cream' 'pizza' 'ramen' 'steak' 'sushi']
```

How about we visualize an image from the training set?

```
# View a random image from the training dataset
import random
img = view_random_image(target_dir=train_dir,
                        target_class=random.choice(class_names)) # get a random class name
```

```
Image shape: (512, 384, 3)
```

Figure 6: Image 6

8 References

1. Smith, J. (2020). *Advances in Food Image Classification.* Retrieved from https://www.example.com/food$_{classification}$