Future of Food Tech Hackathon

Nithish P
Padhma Prakash A

Problem Statement - 1

To Create a model for automatically categorizing items added to the kitchen.

Data Preperation

Dataset used:

https://www.kaggle.com/datasets/theeyeschico/indian-food-classification

Model Architecture

- ResNet (Residual Networks)
- EfficientNet

Model Training

The introduction of residual connections helps mitigate the vanishing gradient problem, making it easier to train deep networks.

ResNet is employed as a feature extractor for image classification. The pre-trained ResNet model, obtained from TensorFlow Hub, provides a powerful foundation for extracting hierarchical features from images.

EfficientNet introduces a compound scaling method to optimize model parameters, providing a favorable trade-off between accuracy and computational resources

EfficientNet is used for a similar purpose which offers a different approach to model efficiency, potentially achieving comparable or better performance with fewer parameters

Why two models ?

- Using two models introduces diversity in the features extracted. Each model may capture different aspects of the input images, enhancing the overall representation.

- Combining predictions from multiple models (ensemble learning) can often lead to improved performance compared to using a single model.

Categories

- Ingredients
- Cuisine
- Veg/NonVeg
- Description

```python
print("\n>> ADD ITEM TO THE KITCHEN\n")
food=pred_and_plot(efficientnet_model, "i1.jpg", class_names)
print(food)
```

```
>> ADD ITEM TO THE KITCHEN

1/1 [==============================] - 0s 298ms/step
Idli
```

Idli: South Indian Vegetarian Diabetic-Friendly

Problem Statement - 2

Last Mile deliver batching :
To design a program that intelligently batches multiple delivery orders for a more streamlined and economical delivery process.

```
C1:
Do You want to order(y/n): y
Enter Kitchen to order from: k1
['item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', 'item8', 'item9', 'item10']
Enter item no.s to order : 1 2 4
[0, 1, 3]
Do You want to order(y/n): y
Enter Kitchen to order from: k1
['item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', 'item8', 'item9', 'item10']
Enter item no.s to order : 2 5
[1, 4]
Do You want to order(y/n): n


C2:
Do You want to order(y/n): n
```

```
C3:
Do You want to order(y/n): y
Enter Kitchen to order from: k3
['item21', 'item22', 'item23', 'item24', 'item25', 'item26', 'item27', 'item28', 'item29', 'item30']
Enter item no.s to order : 22 24
[1, 3]
Do You want to order(y/n): y
Enter Kitchen to order from: k2
['item11', 'item12', 'item13', 'item14', 'item15', 'item16', 'item17', 'item18', 'item19', 'item20']
Enter item no.s to order : 11 12 15
[0, 1, 4]
Do You want to order(y/n): n
```

```
C4:
Do You want to order(y/n): y
Enter Kitchen to order from: k2
['item11', 'item12', 'item13', 'item14', 'item15', 'item16', 'item17', 'item18', 'item19', 'item20']
[0, 3, 5]
Do You want to order(y/n): n
```

```
 ['C1', 'C1', 'C3', 'C3', 'C4'] [[0, 1, 3], [1, 4], [1, 3], [0, 1, 4], [0, 3, 5]]
 ['k1', 'k1', 'k3', 'k2', 'k2']
 [14, 9, 12, 11, 13]
 {0, 1} {2, 3, 4}
```