Food Classification Model

Ismael Deka
Hoan Ngo

Introduction

- The primary goal of this project was to develop an accurate food classification model using a pre-trained Convolutional Neural Network.
- We used transfer learning on ResNet50 to classify 10 different types of foods such as cheeseburgers, cakes, cookies, fries, hotdogs, pizza, salads, shrimp, steak, and sushi.
- Our goal was to achieve high accuracy in classifying these foods, which could have various practical applications.

Motivations

Food classification is important for many different applications such as:

- Helping to figure out the nutritional content of meals based on pictures taken in-person or found online.
- Tracking the different types of food eaten throughout the day so one can track their eating habits.
- Can help food delivery services organize and deliver orders.

Dataset

The dataset this model was trained on is composed of between 3000 images of a 10 standalone food items scraped from various sources (Pinterest, tumblr, reddit, etc). Each category is composed of 300 images to keep the dataset balanced.

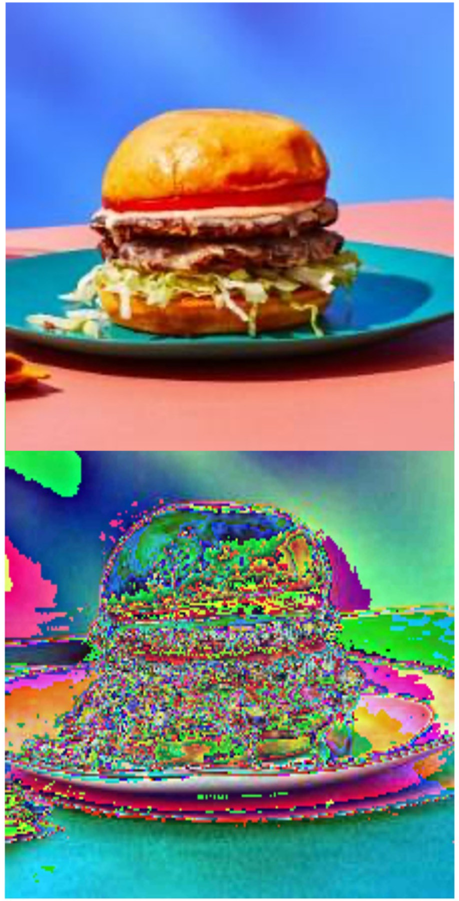
The Image dataset is available for download here.

Samples taken from our dataset

Pre-processing

In order to prepare our dataset, we must preprocess our images before we begin training. Preprocessing includes transforming the images into tensors, resizing their dimensions to 224x224, and normalizing the input data.

- Our images are represented by tensors of shape (C x H x W), where C is the number of channels (3 for RGB images, 1 for grayscale), H is the height of the image, and W is the width.
- We resize our images to 224 x 224 to make sure that they fit the input shape that ResNet50 was originally trained on.
- We normalize the input images using the same mean and standard deviation as ImageNet, which helps the model to perform better on the input images.

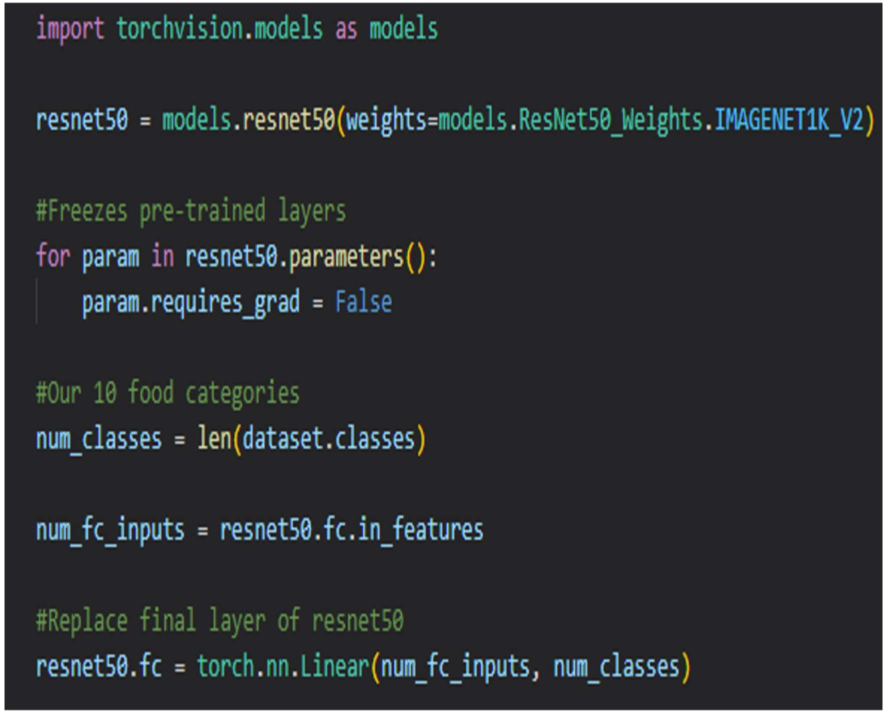

An example of what normalization does to an image.

Method

We used ResNet50 model pre-trained on the ImageNet

We then freezed all pre-trained layers of the original model,

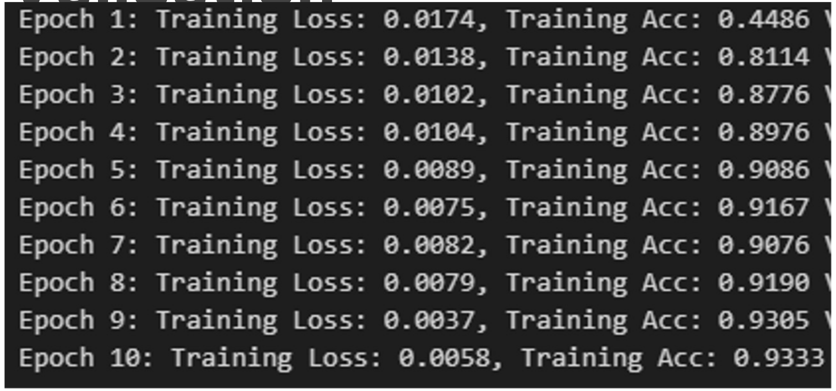
Finally, final fully-connected layer of resnet50 was replaced with a new layer that has the same number of inputs and 10 outputs for our food categories



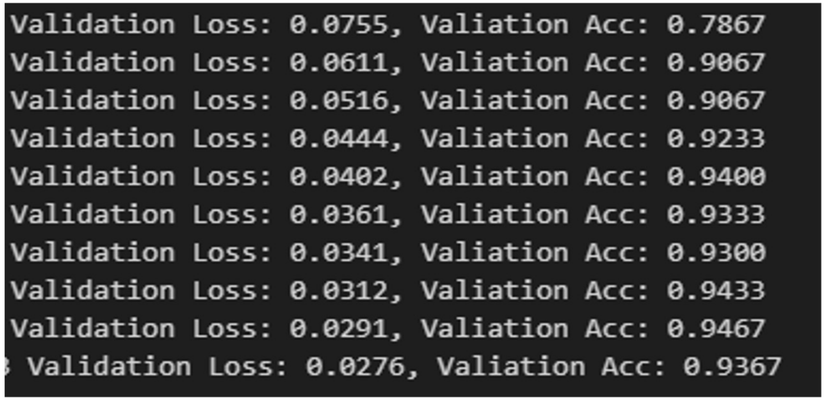
```
import torchvision.models as models

resnet50 = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V2)

#Freezes pre-trained layers
for param in resnet50.parameters():
    param.requires_grad = False

#Our 10 food categories
num_classes = len(dataset.classes)

num_fc_inputs = resnet50.fc.in_features

#Replace final layer of resnet50
resnet50.fc = torch.nn.Linear(num_fc_inputs, num_classes)
```

Method - Training and Validation

- Our dataset is then split into train (70%), validation (10%), and test (20%) sets which are randomly split to make sure that each set has a representative number of images for each food item.
- Dataloaders were created with a batch size of 32.
- We ran training and validation for 10 epochs.
- We used a stochastic gradient descent optimizer for final training layer, with a learn rate of 0.001
- Cross-entropy loss was used as our loss function



```
Epoch 1: Training Loss: 0.0174, Training Acc: 0.4486
Epoch 2: Training Loss: 0.0138, Training Acc: 0.8114
Epoch 3: Training Loss: 0.0102, Training Acc: 0.8776
Epoch 4: Training Loss: 0.0104, Training Acc: 0.8976
Epoch 5: Training Loss: 0.0089, Training Acc: 0.9086
Epoch 6: Training Loss: 0.0075, Training Acc: 0.9167
Epoch 7: Training Loss: 0.0082, Training Acc: 0.9076
Epoch 8: Training Loss: 0.0079, Training Acc: 0.9190
Epoch 9: Training Loss: 0.0037, Training Acc: 0.9305
Epoch 10: Training Loss: 0.0058, Training Acc: 0.9333
```

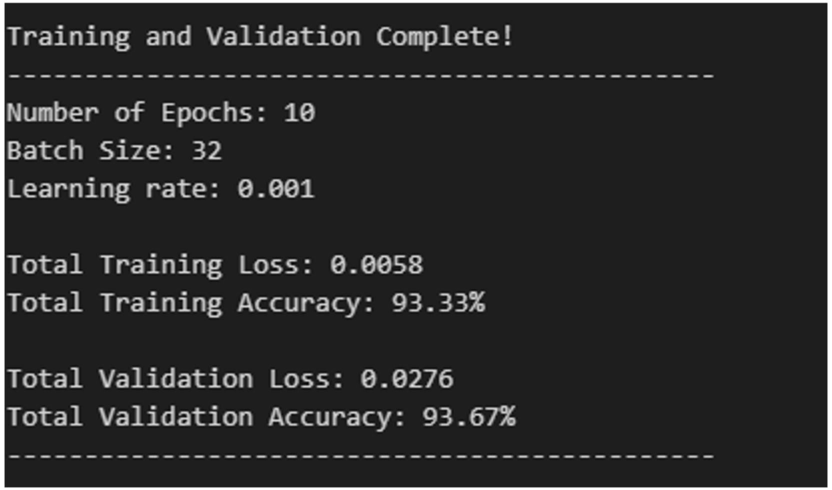


```
Validation Loss: 0.0755, Valiation Acc: 0.7867
Validation Loss: 0.0611, Valiation Acc: 0.9067
Validation Loss: 0.0516, Valiation Acc: 0.9067
Validation Loss: 0.0444, Valiation Acc: 0.9233
Validation Loss: 0.0402, Valiation Acc: 0.9400
Validation Loss: 0.0361, Valiation Acc: 0.9333
Validation Loss: 0.0341, Valiation Acc: 0.9300
Validation Loss: 0.0312, Valiation Acc: 0.9433
Validation Loss: 0.0291, Valiation Acc: 0.9467
 Validation Loss: 0.0276, Valiation Acc: 0.9367
```

Results - Training and Validation

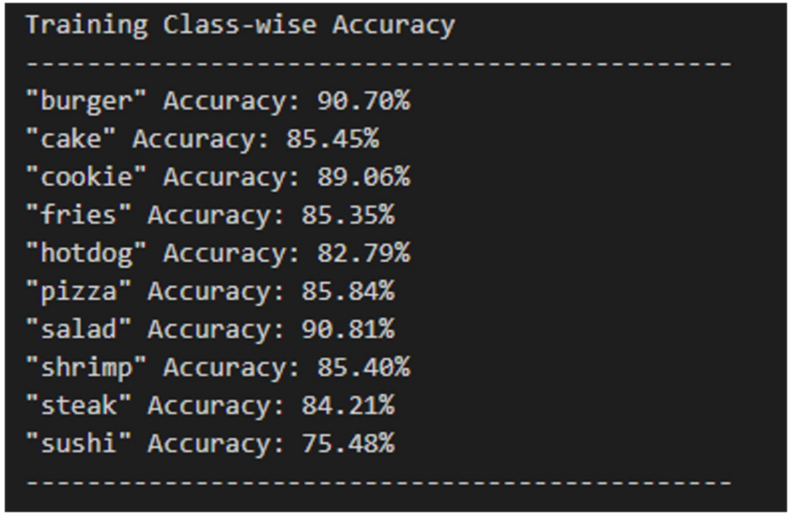
- We achieved a total accuracy of 93.33% and loss of 0.0058 for the training set.

- Accuracy and Loss for the validation set were similar at 93.67% and 0.0276 respectively.

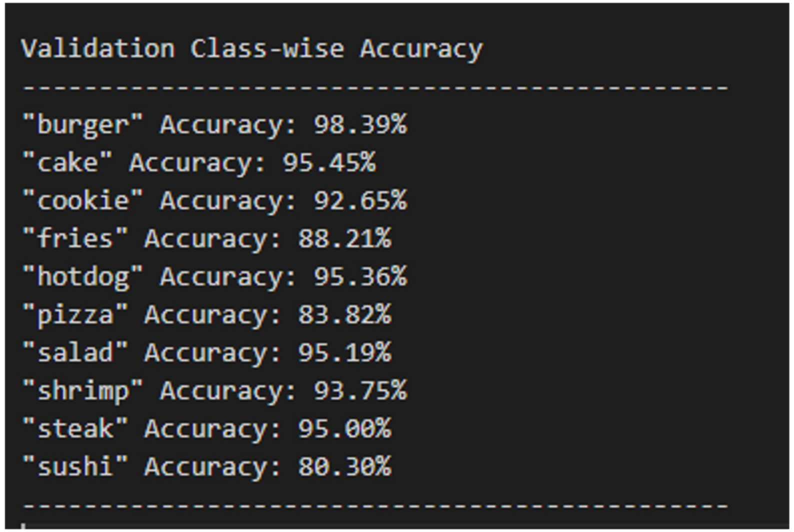


```
Training and Validation Complete!
---------------------------------------------
Number of Epochs: 10
Batch Size: 32
Learning rate: 0.001

Total Training Loss: 0.0058
Total Training Accuracy: 93.33%

Total Validation Loss: 0.0276
Total Validation Accuracy: 93.67%
---------------------------------------------
```

Results - Training and Validation

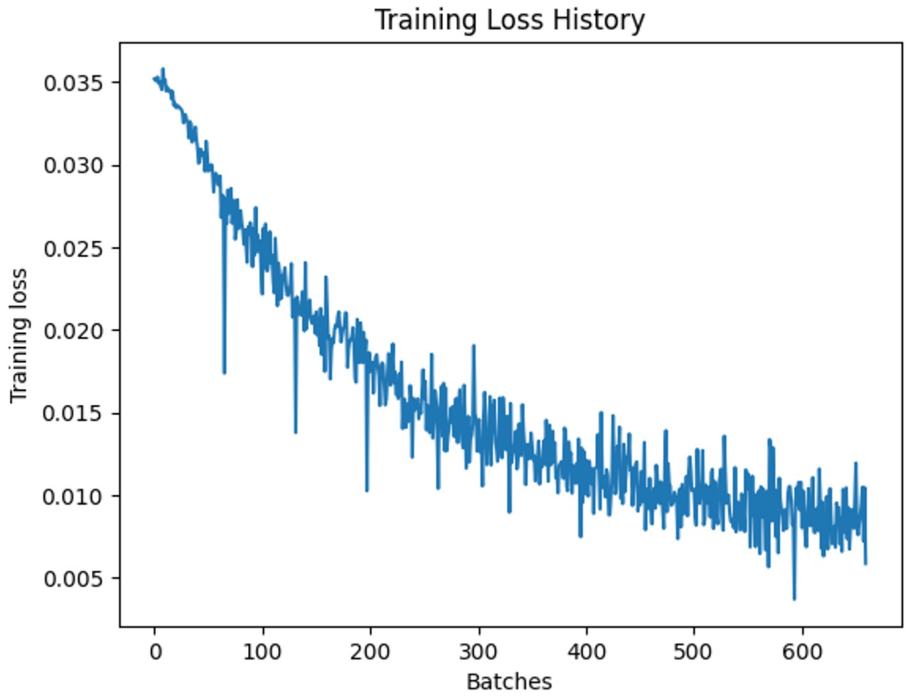


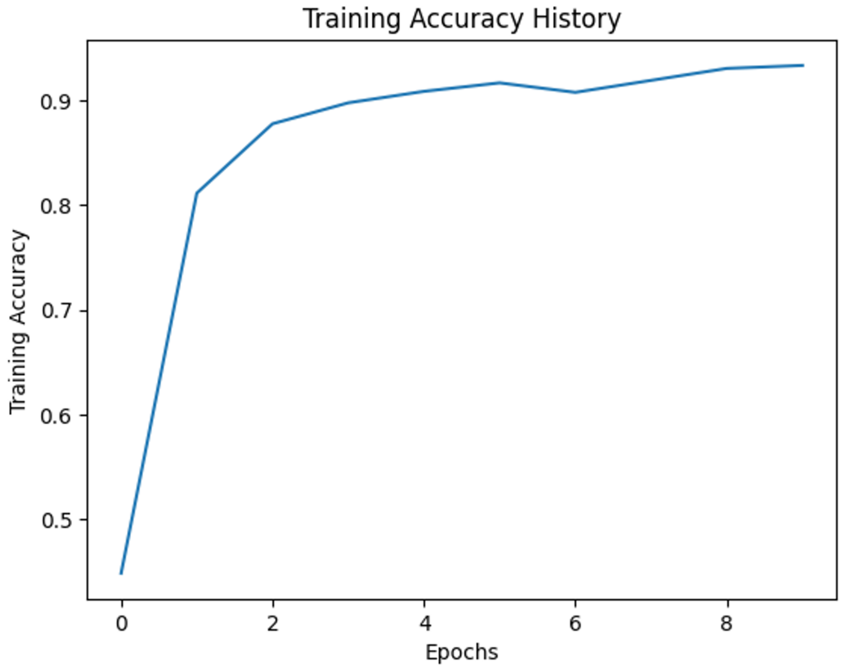
```
Training Class-wise Accuracy
---------------------------------------------
"burger" Accuracy: 90.70%
"cake" Accuracy: 85.45%
"cookie" Accuracy: 89.06%
"fries" Accuracy: 85.35%
"hotdog" Accuracy: 82.79%
"pizza" Accuracy: 85.84%
"salad" Accuracy: 90.81%
"shrimp" Accuracy: 85.40%
"steak" Accuracy: 84.21%
"sushi" Accuracy: 75.48%
---------------------------------------------
```



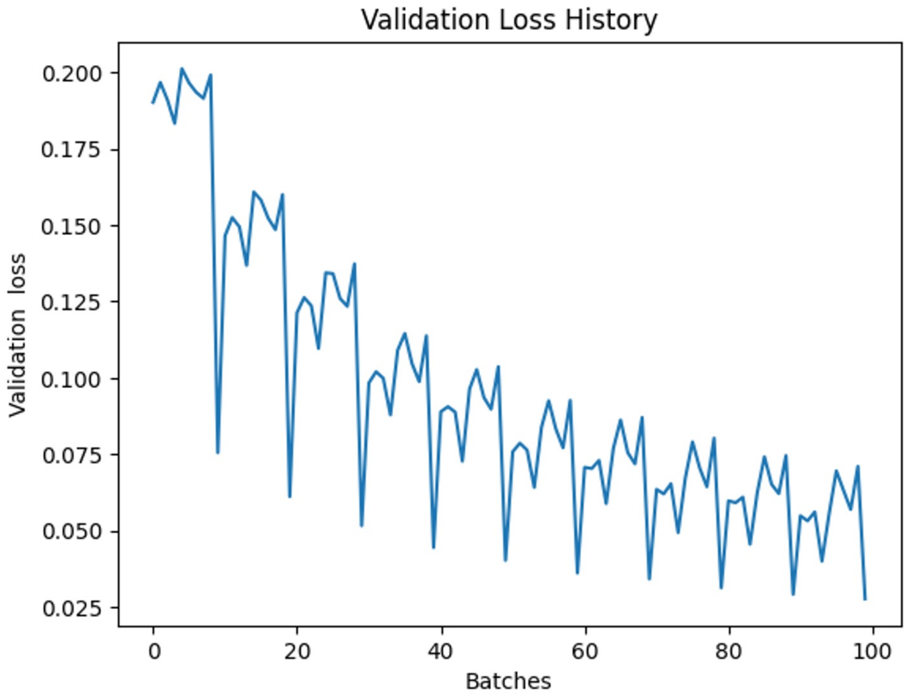
```
Validation Class-wise Accuracy
---------------------------------------------
"burger" Accuracy: 98.39%
"cake" Accuracy: 95.45%
"cookie" Accuracy: 92.65%
"fries" Accuracy: 88.21%
"hotdog" Accuracy: 95.36%
"pizza" Accuracy: 83.82%
"salad" Accuracy: 95.19%
"shrimp" Accuracy: 93.75%
"steak" Accuracy: 95.00%
"sushi" Accuracy: 80.30%
---------------------------------------------
```

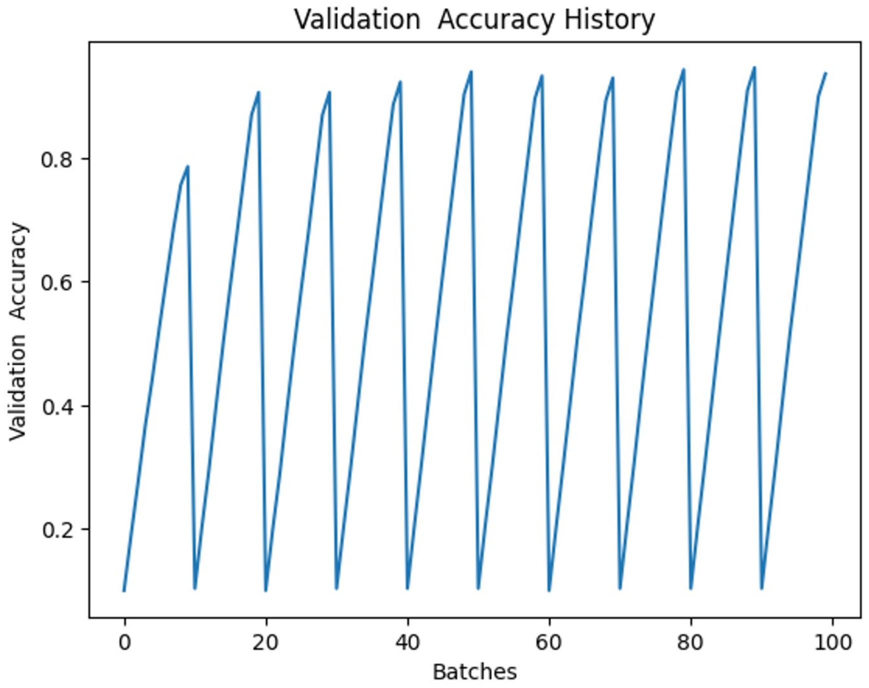
Results - Training



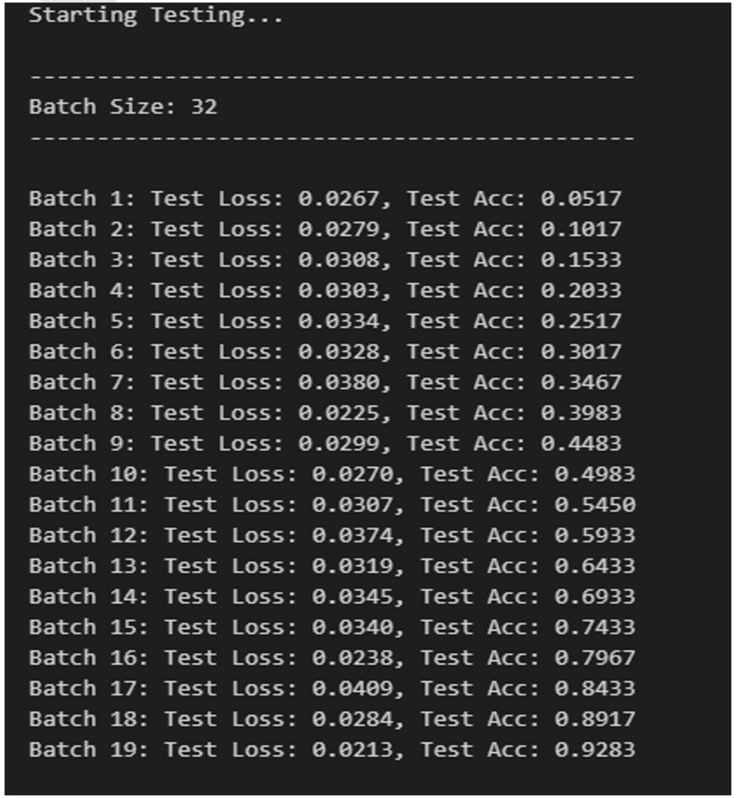


Results - Validation

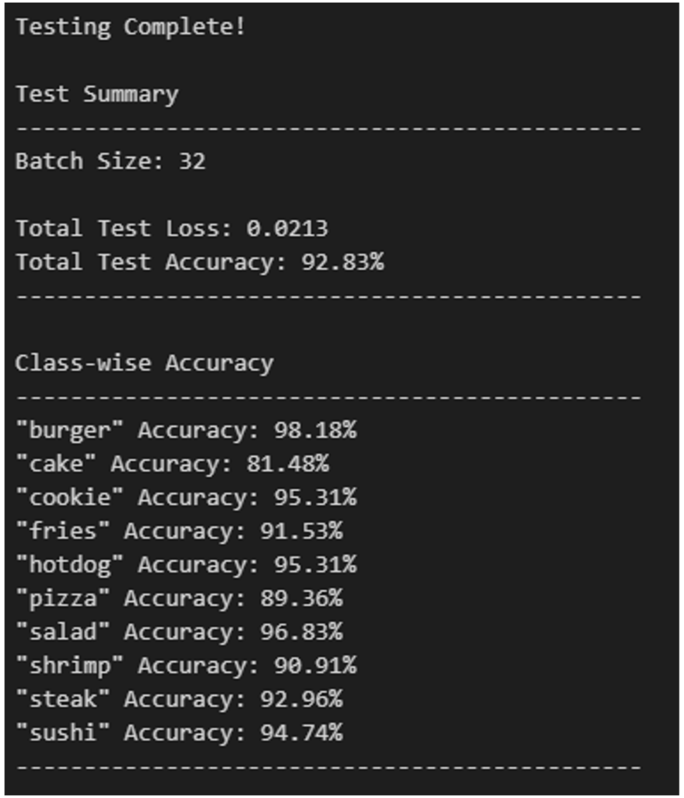




Results - Testing

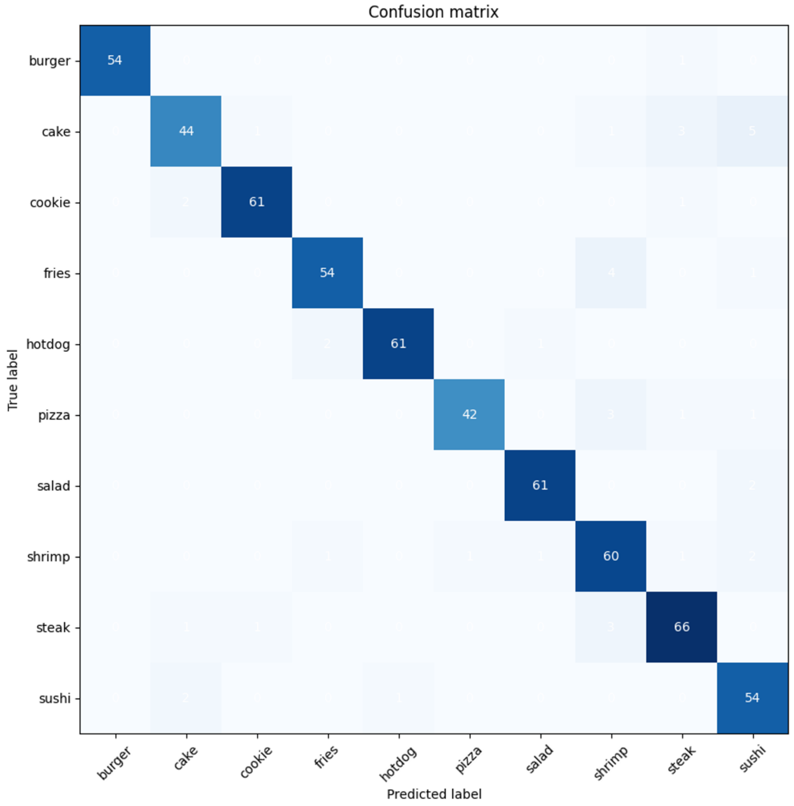


```
Starting Testing...

---------------------------------------------
Batch Size: 32
---------------------------------------------

Batch 1: Test Loss: 0.0267, Test Acc: 0.0517
Batch 2: Test Loss: 0.0279, Test Acc: 0.1017
Batch 3: Test Loss: 0.0308, Test Acc: 0.1533
Batch 4: Test Loss: 0.0303, Test Acc: 0.2033
Batch 5: Test Loss: 0.0334, Test Acc: 0.2517
Batch 6: Test Loss: 0.0328, Test Acc: 0.3017
Batch 7: Test Loss: 0.0380, Test Acc: 0.3467
Batch 8: Test Loss: 0.0225, Test Acc: 0.3983
Batch 9: Test Loss: 0.0299, Test Acc: 0.4483
Batch 10: Test Loss: 0.0270, Test Acc: 0.4983
Batch 11: Test Loss: 0.0307, Test Acc: 0.5450
Batch 12: Test Loss: 0.0374, Test Acc: 0.5933
Batch 13: Test Loss: 0.0319, Test Acc: 0.6433
Batch 14: Test Loss: 0.0345, Test Acc: 0.6933
Batch 15: Test Loss: 0.0340, Test Acc: 0.7433
Batch 16: Test Loss: 0.0238, Test Acc: 0.7967
Batch 17: Test Loss: 0.0409, Test Acc: 0.8433
Batch 18: Test Loss: 0.0284, Test Acc: 0.8917
Batch 19: Test Loss: 0.0213, Test Acc: 0.9283
```



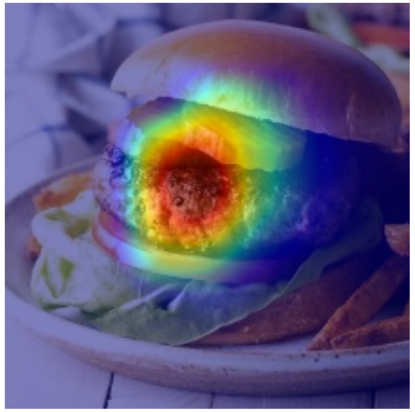
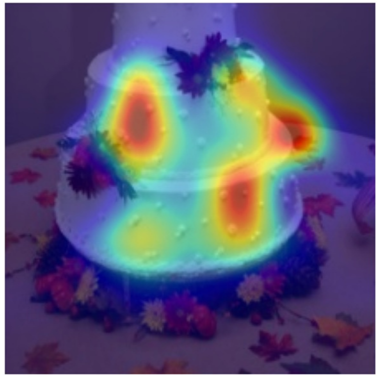
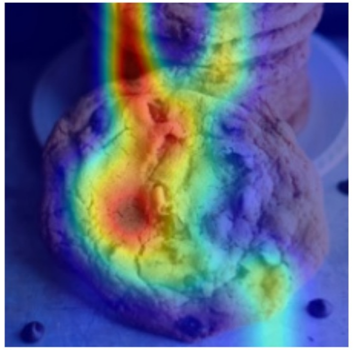
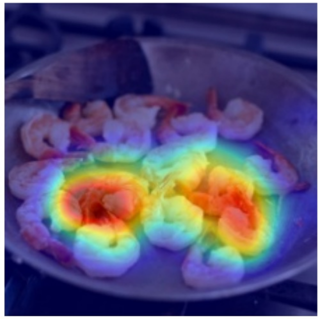
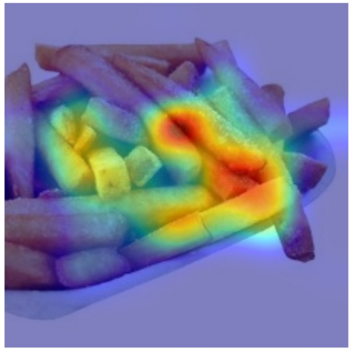
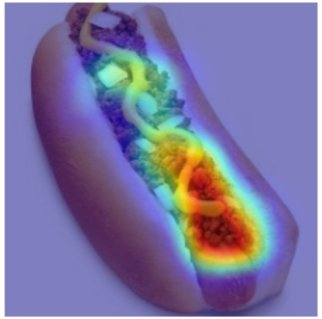
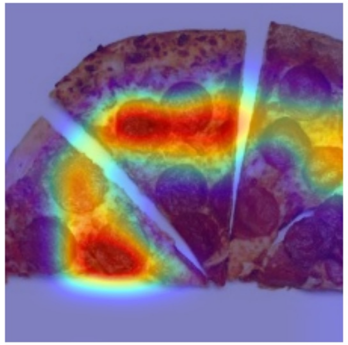
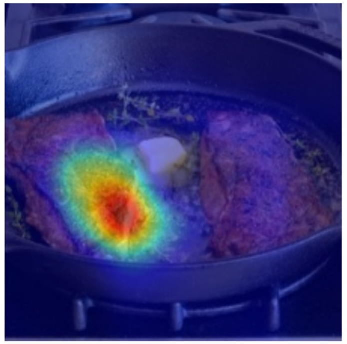
```
Testing Complete!

Test Summary
----------------------------------------------
Batch Size: 32

Total Test Loss: 0.0213
Total Test Accuracy: 92.83%
----------------------------------------------


Class-wise Accuracy
----------------------------------------------
"burger" Accuracy: 98.18%
"cake" Accuracy: 81.48%
"cookie" Accuracy: 95.31%
"fries" Accuracy: 91.53%
"hotdog" Accuracy: 95.31%
"pizza" Accuracy: 89.36%
"salad" Accuracy: 96.83%
"shrimp" Accuracy: 90.91%
"steak" Accuracy: 92.96%
"sushi" Accuracy: 94.74%
----------------------------------------------
```

Results - Testing


Confusion matrix

True label \ Predicted label	burger	cake	cookie	fries	hotdog	pizza	salad	shrimp	steak	sushi
burger	54	0	0	0	0	0	0	0	1	0
cake	0	44	1	0	0	0	0	1	3	5
cookie	0	2	61	0	0	0	0	0	1	0
fries	0	0	0	54	0	0	0	4	0	1
hotdog	0	0	0	2	61	0	1	0	0	0
pizza	0	0	0	0	0	42	0	3	1	1
salad	0	0	0	0	0	0	61	0	0	2
shrimp	0	0	0	1	0	1	1	60	1	2
steak	0	1	1	0	0	0	0	3	66	0
sushi	0	2	0	0	1	0	0	0	0	54

Heatmaps

Conclusion

- Our project showed that transfer learning with a pre-trained ResNet50 architecture can achieve high accuracy for food classification on a relatively small dataset.
- The model achieved an overall accuracy of 92.83%, with class-wise accuracies ranging from 81.48% to 98.18%.
- This shows that transfer learning for image classification tasks can be a powerful tool particularly for individuals developing computer vision software