# Food Classification using Convolutional Neural Networks

Ismael Deka
Georgia State University
33 Gilmer Street SE Atlanta, GA 30303
ideka1@student.gsu.edu

Hoan Ngo
Georgia State University
33 Gilmer Street SE Atlanta, GA 30303
hngo15@student.gsu.edu

## Abstract

*In this project, we aim to classify food images using Convolutional Neural Networks (CNNs). We created a dataset, which consists of 3000 images, and we use transfer learning with pre-trained CNN models to achieve high classification accuracy. Our result demonstrates that using pre-trained CNNs can achieve state-of-the-art performance on the dataset.*

## 1. Introduction

Food classification using CNNs has been a topic of research in recent years, with many studies exploring the effectiveness of various CNN architectures and training techniques for this task. These studies typically involve collecting a large dataset of food images, preprocessing the images to remove noise and enhance their features, and training a CNN on the preprocessed images.

Our dataset is composed of 10 standalone food categories, each with exactly 300 images to ensure balance in the dataset. The food categories included in the dataset are cheeseburger, cake, cookie, fries, hotdog, pizza, salad, shrimp, steak, and sushi. The images used in the dataset were scraped from various sources such as Pinterest, Tumblr, and Reddit. This was done to ensure that there was diversity in the images and that the dataset was not biased towards any specific source. The dataset was then split into training, validation, and testing sets, which are randomly split to make sure that each set has a representative number of images for each food item.

## 2. Related Work

One example of a study on image recognition using CNNs is "Image Recognition Method Based on an Improved Convolutional Neural Network to Detect Impurities in Wheat," by Yin Shen et al. The article proposes a method for recognizing and classifying impurities in wheat using an improved convolutional neural network. A labeled dataset of normal wheat and five impurities is constructed, and image preprocessing is done using the Wiener filtering algorithm and the multi-scale Retinex enhancement algorithm. The WheNet convolutional neural network is designed and compared with ResNet_101 and Inception_v3 networks through comparative experiments. The WheNet network achieves the most efficient results with a shorter training time and higher recognition accuracies for Top_1 and Top_5 of the test set. The proposed method can be used to detect impurities in other fields as well.

Another study on food image classification using deep learning methods is "Development of Korean Food Image Classification Model Using Public Food Image Dataset and Deep Learning Methods," by Minki Chun et al. The study aimed to train a classification model for Korean food images to be used in diet management apps. Korean food images were collected from the AI-Hub platform, pre-processed, and augmented for model training. The proposed model was evaluated in an experiment and effectively classified Korean food images based on transfer learning. The study revealed that the model's performance in classifying food images depended on the type of food, and future work is required to improve the performance of the classification model for some food image types.

Shady Elbassuoni et al. published "DeepNOVA: A Deep Learning NOVA Classifier for Food Images," in 2022. The manuscript proposes an end-to-end deep learning approach to detect and localize various food items in each food image using a customized object detection model. The approach assesses the healthiness of each detected food item by classifying it into one or more of the four NOVA groups. The food item detection model achieved a mAP of 0.90, and the NOVA food classifier achieved an average F1-score of 0.86 on test data. The datasets used for training include two public datasets and a custom one created with images of food taken using wearable cameras, and a custom dataset manually labeled by expert nutritionists.

## 3. Model/Method

We use transfer learning with pre-trained CNN model – ResNet50, to classify food images from the dataset. We fine-tune the last few layers of the pre-trained models on our dataset and evaluate their performance using accuracy and loss metrics. We also perform data augmentation to

increase the size of the training dataset and prevent overfitting.

## 4. Experiments and Results

To prepare the dataset for training, preprocessing of the images is required. This involves transforming the images into tensors, resizing their dimensions to 224x224, and normalizing the input data. The images are represented by tensors of shape (C x H x W,) where C is the number of channels (3 for RGB images and 1 for grayscale,) H is the height of the image, and W is the width. The images are resized to 224x224 to fit the input shape that ResNet50 was originally trained on. Normalization of input images is also performed using the same mean and standard deviation of ImageNet. This is done to improve the model's performance on input images.

After preprocessing the dataset, it is split into 3 sets: train (70%), validation (10%), and test (20%). The splitting is done randomly to ensure that each set has a representative number of images for each food category. Then, data loaders are created for each set with a batch size of 32. Data loaders are used to load the dataset in batches during training, making the training process more efficient.

Next, we loaded the ResNet50 model, which was pre-trained on the ImageNet dataset. All the pre-trained layers of the original model are then frozen, meaning that they will not be updated during the training of the new model. The final layer of the ResNet50 model is replaced with a new layer that has the same number of inputs as the original model but with 10 outputs for the food categories in the dataset. This is necessary to adapt the pre-trained model for the task of recognizing the 10 different food items in the dataset.

During the final training of the model, a stochastic gradient descent optimizer is used for the new final layer with a learning rate of 0.001 and a momentum coefficient of 0.9. This choice of optimizer helps to avoid getting trapping in local minima. Additionally, the cross-entropy loss function is used as the loss function during training. The model will default to using the CPU for training when the GPU is not available.

Our training and validation of the model are set to run for 10 epochs. During training, the optimizer's gradients are set to zero at the start of each batch. Then the batch input is fed into the model, and the model returns the output probabilities for each of the 10 food categories. The criterion is then used to calculate the loss between the model's output and the true label, and the gradients of the loss are computed using backpropagation so that they can be recorded later. The optimizer is then used to update the model using the gradients. At the end of each batch during training, the loss, accuracy, predictions, and ground-truths are recorded. During validation, a forward pass is performed on each batch in the validation set, and the loss, accuracy, predictions, and ground-truths of each batch are calculated and recorded.

```
Epoch 1: Training Loss: 1.6290, Training Acc: 0.6029 Validation Loss: 0.9469, Valiation Acc: 0.8767
Epoch 2: Training Loss: 0.7280, Training Acc: 0.9095 Validation Loss: 0.5852, Valiation Acc: 0.9167
Epoch 3: Training Loss: 0.5031, Training Acc: 0.9238 Validation Loss: 0.4481, Valiation Acc: 0.9267
Epoch 4: Training Loss: 0.4100, Training Acc: 0.9357 Validation Loss: 0.3844, Valiation Acc: 0.9333
Epoch 5: Training Loss: 0.3496, Training Acc: 0.9448 Validation Loss: 0.3350, Valiation Acc: 0.9500
Epoch 6: Training Loss: 0.3101, Training Acc: 0.9481 Validation Loss: 0.2988, Valiation Acc: 0.9533
Epoch 7: Training Loss: 0.2807, Training Acc: 0.9519 Validation Loss: 0.2732, Valiation Acc: 0.9467
Epoch 8: Training Loss: 0.2528, Training Acc: 0.9557 Validation Loss: 0.2592, Valiation Acc: 0.9533
Epoch 9: Training Loss: 0.2379, Training Acc: 0.9543 Validation Loss: 0.2379, Valiation Acc: 0.9600
Epoch 10: Training Loss: 0.2224, Training Acc: 0.9538 Validation Loss: 0.2310, Valiation Acc: 0.9533
```

Figure 1: Training and Validation

After completing the training and validation, we calculated the total loss, accuracy, and class-wise accuracy for both the training and validation sets.

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

Total Training Loss: 0.2224
Total Training Accuracy: 95.38%
-----------------------------------------------

Training Class-wise Accuracy
-----------------------------------------------
"burger" Accuracy: 97.20%
"cake" Accuracy: 94.81%
"cookie" Accuracy: 96.31%
"fries" Accuracy: 92.42%
"hotdog" Accuracy: 93.33%
"pizza" Accuracy: 96.30%
"salad" Accuracy: 97.51%
"shrimp" Accuracy: 95.45%
"steak" Accuracy: 96.37%
"sushi" Accuracy: 94.12%
-----------------------------------------------

Total Valing Loss: 0.2310
Total Valing Accuracy: 95.33%
-----------------------------------------------

Valing Class-wise Accuracy
-----------------------------------------------
"burger" Accuracy: 100.00%
"cake" Accuracy: 100.00%
"cookie" Accuracy: 100.00%
"fries" Accuracy: 93.75%
"hotdog" Accuracy: 93.94%
"pizza" Accuracy: 93.94%
"salad" Accuracy: 91.67%
"shrimp" Accuracy: 95.00%
"steak" Accuracy: 88.24%
"sushi" Accuracy: 100.00%
-----------------------------------------------
```

Figure 2: Training and Validation Results

The saved predictions and ground-truths are used to calculate the confusion matrices for both the training and validation sets. These matrices provide information about how well the model performed in classifying each food item. Additionally, the training and validation curves are plotted, which show how the loss and accuracy changed during the training and validation processes. These curves provide insights into how well the model was able to learn from the training data and generalize to the validation data.

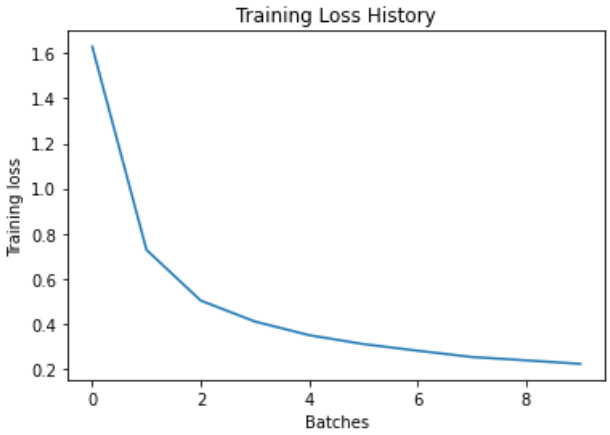


Figure 3: Training Loss History

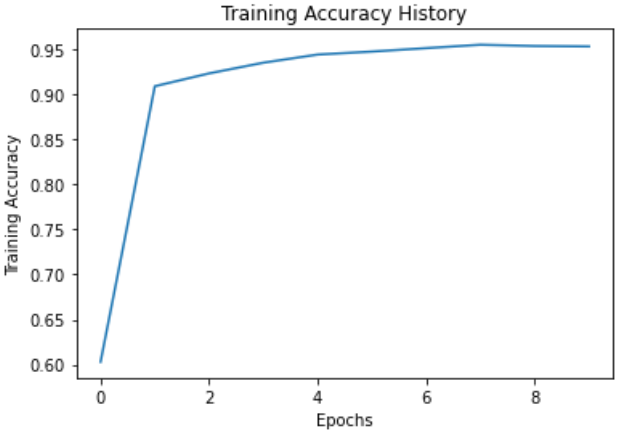


Figure 4: Training Accuracy History

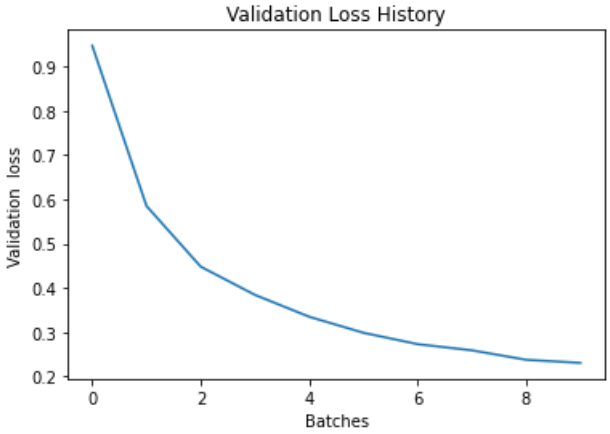


Figure 5: Validation Loss History

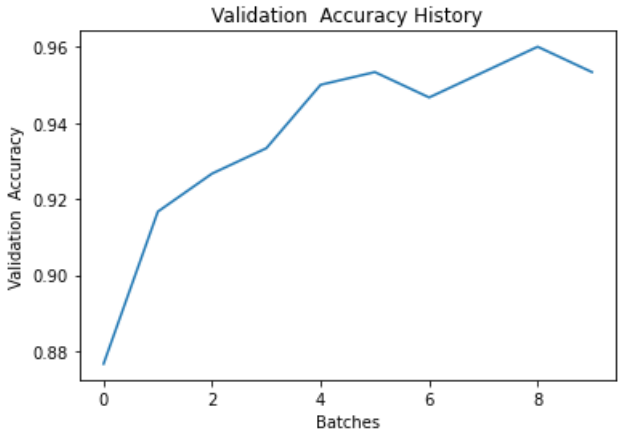


Figure 6: Validation Accuracy History

The model is tested using the test set, which is made up of 20% of the original dataset. The total loss, accuracy, and class-wise accuracy of the test set are then calculated. Additionally, a confusion matrix is plotted between the model's predictions and ground-truths for the test set. This matrix provides information on how well the model was able to classify each food item in the test set.

```
Starting Testing...

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

Batch 1: Test Loss: 0.0112, Test Acc: 0.0533
Batch 2: Test Loss: 0.0056, Test Acc: 0.1050
Batch 3: Test Loss: 0.0231, Test Acc: 0.1517
Batch 4: Test Loss: 0.0077, Test Acc: 0.2033
Batch 5: Test Loss: 0.0081, Test Acc: 0.2533
Batch 6: Test Loss: 0.0083, Test Acc: 0.3050
Batch 7: Test Loss: 0.0101, Test Acc: 0.3583
Batch 8: Test Loss: 0.0113, Test Acc: 0.4100
Batch 9: Test Loss: 0.0083, Test Acc: 0.4617
Batch 10: Test Loss: 0.0082, Test Acc: 0.5133
Batch 11: Test Loss: 0.0099, Test Acc: 0.5650
Batch 12: Test Loss: 0.0110, Test Acc: 0.6183
Batch 13: Test Loss: 0.0132, Test Acc: 0.6683
Batch 14: Test Loss: 0.0114, Test Acc: 0.7217
Batch 15: Test Loss: 0.0238, Test Acc: 0.7650
Batch 16: Test Loss: 0.0148, Test Acc: 0.8150
Batch 17: Test Loss: 0.0136, Test Acc: 0.8633
Batch 18: Test Loss: 0.0077, Test Acc: 0.9167
Batch 19: Test Loss: 0.0046, Test Acc: 0.9567
```

Figure 7: Testing Result

```
Testing Complete!

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

Total Test Loss: 0.0046
Total Test Accuracy: 95.67%
---------------------------------------------

Class-wise Accuracy
---------------------------------------------
"burger" Accuracy: 98.04%
"cake" Accuracy: 97.01%
"cookie" Accuracy: 96.55%
"fries" Accuracy: 92.86%
"hotdog" Accuracy: 92.86%
"pizza" Accuracy: 94.12%
"salad" Accuracy: 96.83%
"shrimp" Accuracy: 95.00%
"steak" Accuracy: 95.89%
"sushi" Accuracy: 96.92%
---------------------------------------------
```

Figure 8: Testing Results


Figure 9: Confusion Matrix

Based on the confusion matrix, we can see that the model is performing relatively well, with the majority of predictions falling in the diagonal, whish represents correct predictions. However, we can also see that there are some misclassifications in the off-diagonal cells. The reason is because they may be similar in some way or share common features that make it difficult for the model to distinguish between them.

Overall, the model seems to be performing well, but there is room for improvement, especially in distinguishing between some of the more challenging classes.

## 5. Conclusion

In this project, we explored the use of transfer learning with pre-trained CNN models for food classification on our dataset. Our results demonstrate that pre-trained models such as ResNet50 can achieve high accuracy on this task. We found that transfer learning is a powerful tool for image classification tasks, particularly in scenarios where the training dataset is small.

## 6. Peer Evaluation

Our team consisted of two members, each with different roles and responsibilities. We believe that each team member made significant contributions to the project, and out collaborative efforts resulted in a successful outcome.

Hoan Ngo was responsible for data collection, and Testing in terms of coding. He also wrote most of the final report, except for Experiments and Results, which is done by Ismael Deka. Deka was responsible for Processing, Training, and Validation for the coding.

## References


[1] Y. Shen et al., "Image Recognition Method Based on an Improved Convolutional Neural Network to Detect Impurities in Wheat," in IEEE Access, vol. 7, pp. 162206-162218, 2019, doi: 10.1109/ACCESS.2019.2946589.

[2] M. Chun, H. Jeong, H. Lee, T. Yoo and H. Jung, "Development of Korean Food Image Classification Model Using Public Food Image Dataset and Deep Learning Methods," in IEEE Access, vol. 10, pp. 128732-128741, 2022, doi: 10.1109/ACCESS.2022.3227796.

[3] S. Elbassuoni et al., "DeepNOVA: A Deep Learning NOVA Classifier for Food Images," in IEEE Access, vol. 10, pp. 128523-128535, 2022, doi: 10.1109/ACCESS.2022.3227769.

[4] M. Sundarramurthi, N. M and A. Giridharan, "Personalised Food Classifier and Nutrition Interpreter Multimedia Tool Using Deep Learning," 2020 IEEE REGION 10 CONFERENCE (TENCON), Osaka, Japan, 2020, pp. 881-884, doi: 10.1109/TENCON50793.2020.9293908.


## Appendix

[1] Dataset: https://drive.google.com/file/d/1eeGF1GQc97_YIwdqewt6nPmob1bNr7M5/edit

[2] Code: https://github.com/HoanNgo1592/Food-Classification/blob/main/food_classification.ipynb