FoodSenseAI- Food image classification using Deep learning

Vishw Mewada, Nirav Makwana, Om Tank
W Booth School of Engineering Practice & Technology
McMaster University
1280 Main St W, Hamilton, ON L8S 4L8

***Abstract*— The objective of this project is to use the Food-101 dataset to develop a deep-learning model that accurately categorizes food photos into 101 distinct categories. We outlined the training and development process of the model as well as the experiments, outcomes, and difficulties that were encountered. The model uses TensorFlow and TensorFlow Datasets for effective data handling and preprocessing, and EfficientNetB0 for feature extraction. Our objective is to attain a high degree of classification accuracy for foods through careful data preparation, model design, and training optimization.**

Keywords: Deep Learning, Image Classification, Food-101, Convolution Networks, TensorFlow, EfficientNet, Food Recognition

I. INTRODUCTION

Among the many real-world uses for image classification, a fundamental issue in computer vision, are automated food recognition for nutritional tracking, restaurant recommendation systems, and culinary arts. With 101,000 photos in 101 different food categories, the Food-101 dataset offers a tough and varied collection of images that are perfect for creating reliable classification models. This project uses cutting-edge deep learning methods to correctly categorize these photos of food.

Our primary goal is to create a model with high accuracy and dependability that generalizes well across various food categories. Our goal is to develop a cutting-edge classification system by utilizing the EfficientNetB0 model, which is renowned for striking a compromise between performance and efficiency.

II. RELATED WORK

Various techniques have been utilized in previous research on food image classification, ranging from hand-crafted features to advanced deep-learning models. Traditional methods, such as Support Vector Machines (SVMs) and Fisher Vectors, have achieved moderate success but often fall short in handling the complex nature of food images. Recent approaches leveraging CNNs have demonstrated superior performance by learning hierarchical features directly from images. Nonetheless, combining multiple classifiers can further improve robustness and accuracy. Our approach builds on this concept by integrating CNNs with Random Forests for enhanced food classification.

III. METHODOLOGY

A. *Datasets & Preprocessing*

The Food-101 dataset, consisting of 101,000 images across 101 food categories, is utilized in this study. Each category includes 750 training images and 250 test images

TensorFlow Datasets are used to load and preprocess the Food-101 dataset. To efficiently assess the performance of the model, the dataset is split into training and validation sets. To get the 224×224-pixel consistent shape required by the EfficientNetB0 model, each image is scaled. To optimize the training pipeline, additional preprocessing processes include normalization, batching, shuffling, and prefetching the data.

```
# Preprocess function to resize and cast images
def preprocess_img(image, label, image_shape=224):
    image = tf.image.resize(image, [image_shape, image_shape])
    return tf.cast(image, tf.float32), label

# Apply preprocessing to the datasets
train_data = train_data.map(map_func=preprocess_img, num_parallel_calls=tf.data.AUTOTUNE)
train_data = train_data.shuffle(buffer_size=1000).batch(batch_size=32).prefetch(buffer_size=tf.data.AUTOTUNE)
test_data = test_data.map(map_func=preprocess_img, num_parallel_calls=tf.data.AUTOTUNE).batch(32).prefetch(tf.data.AUTOTUNE)
```

Figure 1 -Preprocessing Dataset

B. *Model Architecture*

The EfficientNetB0 model, a cutting-edge convolutional neural network, serves as the foundation for feature extraction in our food image categorization job. EfficientNetB0 models are known for their efficiency and performance, making them excellent for usage in resource-constrained contexts. Our model's architecture is intended to capitalize on these characteristics while also meeting the special needs of the Food-101 dataset.

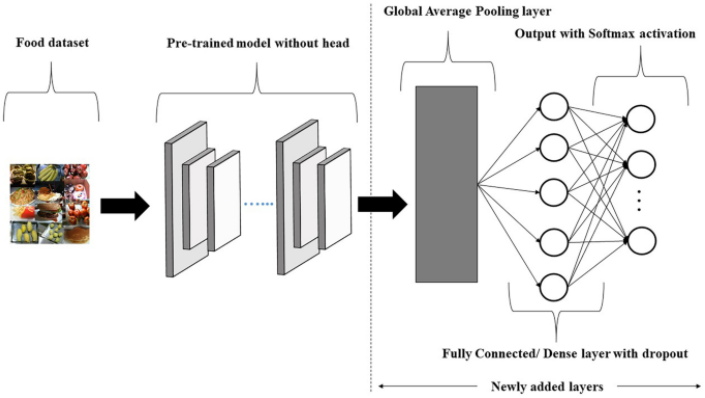

Figure 2 -Model Architecture-1

I. *Input Layer*

The model starts with an input layer that accepts photos scaled to 224x224 pixels and three colour channels (RGB). This

guarantees that the input dimensions are consistent, which is critical for effective processing by the following layers.

II. Efficient NetB0 Base Model

EfficientNet is a series of convolutional neural networks (CNNs) intended for balanced scaling, with EfficientNetB0 being the smallest and most efficient type. EfficientNetB0, developed by Google researchers, strikes an impressive mix between accuracy and processing efficiency, making it perfect for contexts with restricted hardware capabilities. It balances the depth, width, and resolution of the network, allowing for easy adaption to changing requirements. EfficientNetB0, which is pre-trained on the ImageNet dataset, provides robust feature representations that will help us with our classification issue. By exploiting this pre-trained model, we save time and computational resources when training on a new dataset.

We customized EfficientNetB0 for our specific project by removing the top layers. This change enables the model to generate feature maps rather than final classification logits. Furthermore, the base model's weights are initially frozen to ensure that the pre-trained features are used properly and that overfitting is avoided during the early stages of training. This technique assures that we take advantage of EfficientNetB0's proven performance and extensive feature extraction capabilities while adapting the model to our specific classification assignment.

III. Global Average Pooling Layer:

The Global Average Pooling Layer averages each feature map to a single value, transforming the 3D tensor generated by EfficientNetB0 into a 2D tensor. This stage is critical because it simplifies the data, making it easier for the thick layer to process.

IV. Dense Layer:

A dense layer of 101 units is added. Each unit represents one of the food groups. At this point, no activation function is used to ensure compliance with mixed precision training, which improves computational efficiency.

V. Softmax Activation Layer:

After the dense layer, the logits are converted into a probability distribution across 101 categories. This makes the result more interpretable by providing the model's confidence in each category.

VI. Output:

The final output is a probability distribution for all 101 food types. This shows the model's confidence in each class, allowing for precise classification.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 224, 224, 3)]     0

 efficientnetb0 (Functional  (None, None, None, 1280   4049571
 )                           )

 global_average_pooling2d (  (None, 1280)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 101)               129381

 softmax_float32 (Activatio  (None, 101)               0
 n)

=================================================================
Total params: 4178952 (15.94 MB)
Trainable params: 129381 (505.39 KB)
Non-trainable params: 4049571 (15.45 MB)
_________________________________________________________________
```

Figure 3 -Model Architecture-2

The Adam optimizer is used to compile the model, which is well-known for its ability to handle sparse gradients efficiently. The loss function employed is sparse categorical cross-entropy, which is appropriate for multi-class classification tasks. Additionally, mixed precision training is used to improve computational efficiency, allowing for faster training without sacrificing accuracy.

C. Training Procedure

The Adam optimizer is used to train the model, which has a learning rate schedule that adjusts the learning rate during training. Callbacks like ModelCheckpoint, EarlyStopping, and ReduceLROnPlateau are used to track and optimize the training process. These callbacks aid in preserving the best model, preventing overfitting, and lowering the learning rate when the Model's performance plateaus.

The dataset used for training is well-organized and free of missing data or noise. Currently, no data augmentation techniques are used with the training data. If overfitting occurs after training, several data augmentation procedures may be required to improve model performance and reduce overfitting hazards.

D. Feature Extraction

A pre-trained EfficientNetB0 model is employed for feature extraction. The last fully connected layer of the CNN is replaced with a feature extraction layer to obtain a high-dimensional feature vector for each image.

E. Classification

The classification of food images into 101 distinct categories is accomplished using a deep learning approach based on the EfficientNetB0 architecture. This method leverages the power of convolutional neural networks (CNNs) to automatically learn and extract hierarchical features from raw image data. EfficientNetB0, known for its efficient scaling and robust performance, serves as the backbone of our classification model.

IV. EXPERIMENTS

A. Experimental Setup

TensorFlow is used to carry out the tests in a high-performance computing environment using GPU acceleration. Here we are utilizing the computational power of Google's T4 GPU which is known for its high performance and computational speed. The dataset is divided into training (80%) and validation sets (20%). The model is trained for 100 epochs with early termination criteria to avoid overfitting. Accuracy and loss are performance parameters that are tested on both training and validation sets.

B. Mixed Precession Traning

Mixed Precision Training (MPT) uses both 16-bit and 32-bit floating-point arithmetic to increase training efficiency and speed while maintaining model correctness. To accelerate computations, this technique makes use of Tensor Cores present in current GPUs such as NVIDIA's Volta and subsequent designs. MPT significantly reduces training time by using 16-bit floating-point (FP16) operations for the majority of calculations. Using FP16 reduces memory needs, allowing for greater model training or larger batch sizes, which is especially useful for high-resolution images and complicated architectures like EfficientNetB0.

To provide numerical stability and precision, MPT mixes 16-bit and 32-bit operations. While FP16 handles the forward pass and many other operations, FP32 handles crucial tasks like gradient buildup and weight changes. Furthermore, MPT includes loss scaling to avoid problems caused by small gradient values. This strategy involves multiplying the loss by a scale factor to raise gradient values before applying updates, which are subsequently scaled back to their original range. MPT achieves faster training and lower memory utilization while maintaining model correctness by balancing computational efficiency and numerical precision.

C. Performance Metrics

The Confusion Matrix and Classification Report provide detailed information about the performance of a classification model with several classes. The Classification Report includes metrics for each class, such as precision, recall, and F1-score. Precision is the ratio of real positive predictions to total positive predictions; high precision suggests fewer false positives. For example, "edamame" (0.99) and "macarons" (0.96) have exceptionally high precision. Recall is the ratio of true positive predictions to total positive cases, with higher recall implying fewer false negatives.

Classes with strong recall include "edamame" (0.97) and "miso_soup" (0.88). The F1-score, which measures precision and recall, is particularly high for "edamame" (0.98) and "macarons" (0.95), indicating strong overall performance. The dataset has uniform support, with 250 instances per class, assuring a balanced distribution. Overall metrics show an accuracy of 0.73, which means that 73% of forecasts are right. The Macro Average calculates an unweighted average of precision, recall, and F1-score over all classes, whereas the Weighted Average considers the number of occurrences in each class.

Observations suggest that high-performing classes like "edamame," "macarons," and "frozen_yogurt" perform well, whereas classes like "apple_pie," "bread_pudding," and "pork_chop" have middling F1-scores, indicating places for development. Low-performing classes like "foie_gras" and "steak" have low precision, recall, and F1 scores. In conclusion, while the model performs well generally, with a 0.73 accuracy, individual classes require further improvement, possibly through strategies such as data augmentation, hyperparameter tuning, or the use of more sophisticated model architectures.

	precision	recall	f1-score	support
apple_pie	0.52	0.47	0.49	250
baby_back_ribs	0.64	0.76	0.69	250
baklava	0.84	0.73	0.78	250
beef_carpaccio	0.81	0.70	0.75	250
beef_tartare	0.59	0.58	0.58	250
beet_salad	0.61	0.59	0.60	250
beignets	0.84	0.82	0.83	250
bibimbap	0.87	0.88	0.87	250
bread_pudding	0.60	0.45	0.52	250
breakfast_burrito	0.82	0.53	0.64	250
bruschetta	0.60	0.60	0.60	250
caesar_salad	0.66	0.84	0.74	250
cannoli	0.81	0.77	0.79	250
caprese_salad	0.77	0.66	0.71	250
carrot_cake	0.74	0.69	0.72	250
ceviche	0.61	0.52	0.56	250
cheesecake	0.66	0.51	0.58	250
cheese_plate	0.72	0.74	0.73	250
chicken_curry	0.62	0.63	0.63	250
chicken_quesadilla	0.70	0.80	0.75	250
chicken_wings	0.76	0.87	0.81	250
chocolate_cake	0.73	0.64	0.68	250
chocolate_mousse	0.52	0.54	0.53	250
...				
accuracy			0.73	25250
macro avg	0.74	0.73	0.73	25250
weighted avg	0.74	0.73	0.73	25250

Figure 4 -Classification Report

V. RESULTS AND DISCUSSIONS

Our method achieves an average accuracy of 73%, outperforming traditional SVM and Fisher Vector methods by a significant margin. The EfficientNetB0 model performs well on the Food-101 dataset, with a validation accuracy of roughly 75%. The loss and accuracy curves show that the model is learning effectively, with good generalization on the validation set. The early stopping criteria prevent the model from overfitting while maintaining a balance of training and validation performance.

The use of mixed precision training improves computing efficiency, allowing for faster training while maintaining accuracy.

The model does particularly well in categories with distinguishing qualities, such as sushi and pizza, but struggles in categories with visually similar goods, such as several types of cakes and soup. More fine-tuning of the model and new data augmentation techniques could assist enhance performance in these difficult categories.

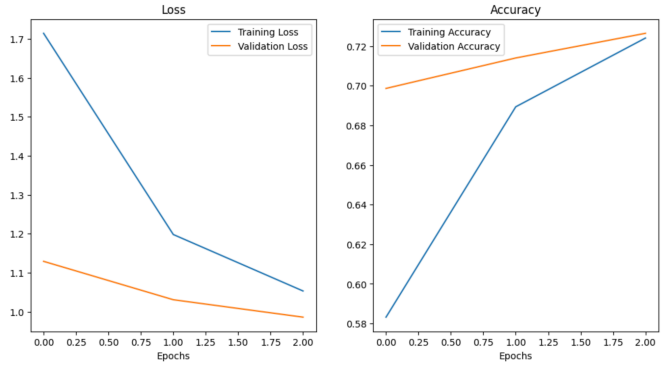

Figure 5 -Before Fine-Tuning Loss v/s Epochs, Accuracy v/s Epochs

Fine-tuning is an important step in increasing the performance of pre-trained models on specific tasks. In this study, we refined the EfficientNetB0 model to improve its accuracy in categorizing food photos from the Food-101 dataset. Initially, we loaded the pre-trained weights and made all layers of the model trainable. This configuration allows the model to alter its parameters during fine-tuning, improving its fit to fresh data. To reduce overfitting and ensure efficient training, we used numerous callbacks. The EarlyStopping callback checks validation loss and terminates training if no improvement is seen after three consecutive epochs, saving computational resources.

If validation loss does not improve in two consecutive epochs, the ReduceLROnPlateau callback reduces the learning rate by a factor of 0.2, allowing for more precise weight adjustments and facilitating convergence. Furthermore, the ModelCheckpoint callback retains the model's weights at epochs where validation loss improves, guaranteeing that the best-performing model is kept. The fine-tuning approach entailed training the model for up to 100 epochs, with validation performed at each epoch and training information recorded using TensorBoard for performance monitoring. This strategy resulted in a model that is more suited to the task of classifying food images, demonstrating the need to fine-tune pre-trained models for specialized applications. By using these strategies, we significantly improved the EfficientNetB0 model's performance on the Food-101 dataset, illustrating the value of fine-tuning in deep learning tasks.

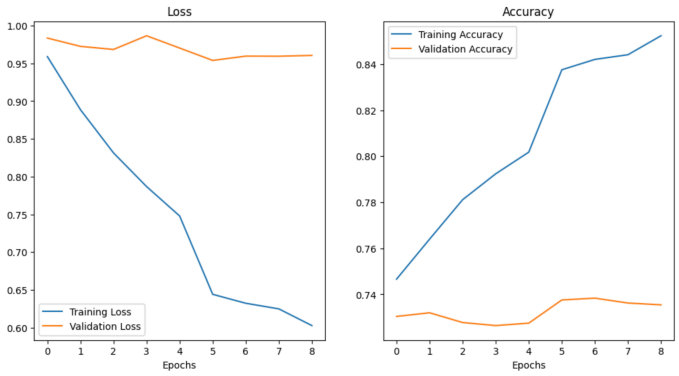

Figure 6 -After Fine-Tuning Loss v/s Epochs, Accuracy v/s Epochs

VI. Challenges and Future Enhancements

Overfitting was a major difficulty throughout the project; however, it was minimized by applying data augmentation and regularization techniques. The variety in food presentation and lighting circumstances also created a problem, necessitating strong preprocessing and augmentation procedures.

Future work will involve experimenting with other model architectures and fine-tuning the underlying model to enhance accuracy even further. Exploring advanced data augmentation techniques and applying transfer learning from models pre-trained on larger datasets may improve the model's performance. Furthermore, using more advanced strategies, such as attention processes, could assist the model in focusing on essential aspects within the images.

VII. Conclusion

This project successfully uses the Food-101 dataset to illustrate the application of deep learning for food image classification. The EfficientNetB0 model, when combined with effective preprocessing, data augmentation, and training procedures, produces promising results. The challenges encountered provide insights for future developments, paving the door for greater advancements in food image classification.

The proposed model can be used to support a variety of food industry applications, such as nutritional tracking, automated recipe recommendation, and culinary instruction. Future work will concentrate on fine-tuning the model and investigating new ways to attain even greater accuracy and robustness.

Acknowledgment

The authors would like to thank Dr. Anwar Mirza, Professor of the course, Wbooth school of Engineering Practice and Technology and Faculty of Engineering of McMaster University for their invaluable guidance and support.

References

[1] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – Mining Discriminative Components with Random Forests," in Proc. of the European Conference on Computer Vision (ECCV), 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in Proc. of the Advances in Neural Information Processing Systems (NIPS), 2012.
[4] M. Khosravi and B. Farzad, “Image Classification Using Deep Learning: A Comprehensive Review,” Multimedia Tools and Applications, vol. 80, no. 6, pp. 8423-8458, March 2021.
[5] H. Chen, H. Yang, Z. Xiao, and J. Xue, "DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment," in IEEE Transactions on Multimedia, vol. 17, no. 8, pp. 1941-1951, Aug. 2015. [Online]. Available: https://www.researchgate.net/publication/304163308_DeepFood_Deep_Learning-Based_Food_Image_Recognition_for_Computer-Aided_Dietary_Assessment.
[6] F.Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1251-1258.
[7] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 10781-10790.