BIM447 Introduction to Deep Learning Term Project Proposal

I. Project Title

Fast Food Classification

II. Project Members

İSMAİL ÇINAR
ŞULENUR ŞAHAN

III. Problem Definition

Recognizing different types of fast food from images is a useful task in many areas such as smart restaurants, mobile apps, and health tracking. However, many fast food items look very similar — for example, fries and baked potatoes, or tacos and taquitos — which makes the classification difficult.

This project aims to train a deep learning model that can classify images of 10 different fast food categories. The Fast Food Classification Dataset from Kaggle is used for this purpose. The main goal is to develop a system that can automatically recognize the type of fast food shown in an image.

IV. Methodology

In this study, two different convolutional neural network architectures were implemented and compared: a custom-built Simple Convolutional Neural Network and a pretrained DenseNet121 model adapted through transfer learning. The objective was to observe the performance differences between a model trained entirely from scratch and a deep, pretrained model fine-tuned for the specific classification task.

Both models were trained and evaluated using the same dataset under controlled experimental conditions. Data augmentation techniques, optimization parameters, and training strategies were applied differently for each architecture based on their design and capacity.

Simple Convolutional Neural Network: The first model is a manually built convolutional neural network that was implemented entirely from scratch. The model consists of seven convolutional blocks, each containing a 3×3 convolutional layer, batch normalization, a ReLU activation function, and a max pooling layer. The number of filters in these blocks increases progressively from 16 up to 1024. This progressive depth enables the network to learn both low-level patterns such as edges and textures, as well as more abstract representations like shapes and object parts.

After the final convolutional block, an adaptive average pooling operation reduces the feature maps to a 1×1 dimension. This output is passed into a fully connected classifier composed of three linear layers, with dimensions reducing from 1024 to 512, then to 256, and finally to 10, representing the number of food categories. Dropout is applied after the first two dense layers, with rates of 0.3 and 0.2, to reduce overfitting.

The model was trained from scratch, which requires strong generalization ability. Therefore, data augmentation was an essential component of the pipeline. Training images were first resized to 224×224 pixels to match the input dimensions expected by the model. Then, multiple transformations were applied to introduce diversity into the training set. These included random horizontal flipping, rotations up to 10 degrees, and brightness, contrast, and color adjustments to simulate different viewpoints and lighting conditions.This combination of augmentations improves the model's robustness to variation in image quality, viewpoint, and lighting. After augmentation, all pixel values were normalized using mean and standard deviation values commonly used for ImageNet-pretrained models to standardize the data distribution.

Training was conducted using the Adam optimizer with a base learning rate of 0.0001 and weight decay of 0.0001 to control overfitting. The loss function used was cross-entropy loss combined with label smoothing at a value of 0.1. Label smoothing adjusts the target labels slightly to prevent the model from becoming too confident in its predictions, which can improve generalization. The One Cycle Learning Rate scheduler adjusted the learning rate dynamically throughout the 40 training epochs. A batch size of 64 was used, balancing computational efficiency with model performance. Finally, early stopping was implemented to terminate training if the model failed to improve on the validation set for five consecutive epochs.

DenseNet121: The second model used in this study is based on DenseNet121, a deep convolutional neural network architecture known for its densely connected layers. Unlike traditional CNNs where each layer passes information only to the next, DenseNet connects each layer to every subsequent layer in a feed-forward fashion. This design encourages feature reuse and improves gradient flow, making the network more parameter-efficient and easier to train.

In this implementation, a pretrained DenseNet121 model from the torchvision.models library was used. The model was originally trained on the large-scale ImageNet dataset, allowing it to extract a wide variety of visual features. To adapt the network to the fast food classification task, the final classification head of the network was removed and replaced with a new fully connected block suitable for 10 output classes.

The new classifier consists of three linear layers: the first reduces the 1024-dimensional feature vector to 512 units, followed by a ReLU activation and a dropout rate of 30

To allow the model to adapt while preserving the useful features learned during ImageNet training, only the last 45 layers of the DenseNet feature extractor were set to be trainable. The earlier layers were frozen, which reduces the number of parameters being updated and protects the general feature representations already encoded in the network.

Because this model already contained powerful learned features, a more conservative data augmentation strategy was adopted. The input images were resized to 224×224 pixels and subjected only to horizontal flipping, which introduces minor variation without significantly altering the image structure. No rotation, brightness, or color shifts were used, in order to maintain compatibility with the pretrained layers. Like the SimpleCNN, input images were normalized using ImageNet mean and standard deviation values.

The training configuration mirrored that of SimpleCNN: the Adam optimizer was used with the same learning rate and weight decay, and the cross-entropy loss function included label smoothing at a value of 0.1. The One Cycle Learning Rate scheduler was also applied to control the learning rate throughout the training process. Due to the increased memory requirements of DenseNet121, the batch size was reduced to 16. Early stopping was once again employed to monitor validation performance and terminate training when necessary.

V. Dataset Description

The dataset used in this study is based on Version 2 of the Fast Food Classification Dataset, which contains real-world images of 10 popular fast food items: *Baked Potato, Burger, Crispy Chicken, Donut, Fries, Hot Dog, Pizza, Sandwich, Taco, and Taquito*. Each class includes 2,000 images, split into training, validation, and testing subsets.

The dataset was manually organized to ensure balanced distribution across all classes. Images are divided as follows: 15,000 images for training, 3,500 for validation, and 1,500 for testing. All images were resized to 224×224 pixels and preprocessed before being passed to the models. This structure supports robust evaluation while maintaining realistic visual diversity. The original dataset is publicly available on Kaggle:

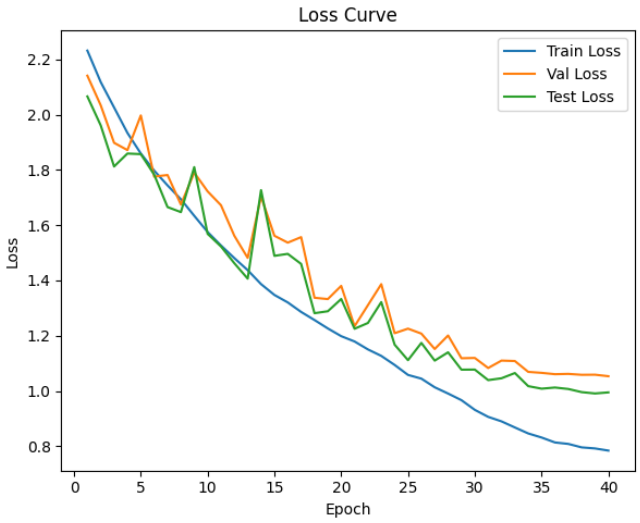
https://www.kaggle.com/datasets/utkarshsaxenadn/fast-food-classification-version-2

Key Features

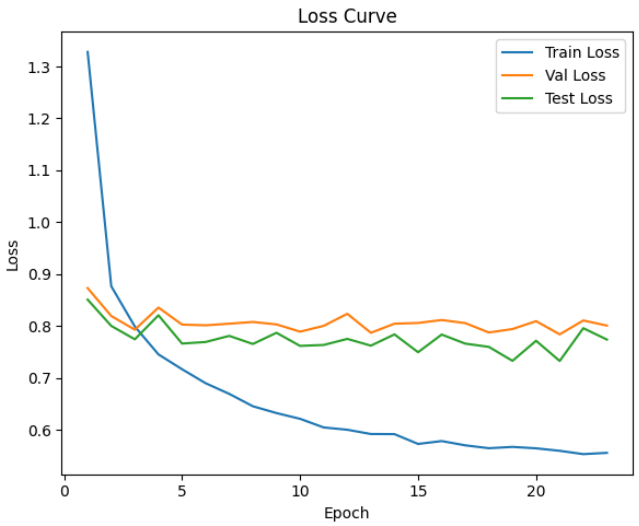
- Balanced class distribution across training, validation, and testing sets
- Ten diverse fast food categories with high inter-class similarity
- Images collected from real-world environments with varying lighting, angle, and presentation
- Image format: color JPEGs resized to 224×224 pixels
- Suitable for deep learning-based image classification and transfer learning experiments

VI. Results

The performance of both models was evaluated using training, validation, and test datasets. The SimpleCNN model was trained for a total of 40 epochs. In contrast, the DenseNet121 model utilized an early stopping strategy, which automatically stopped training at epoch 23 when no further improvement was observed in validation accuracy. This approach helped DenseNet121 avoid overfitting and enhanced its generalization ability.



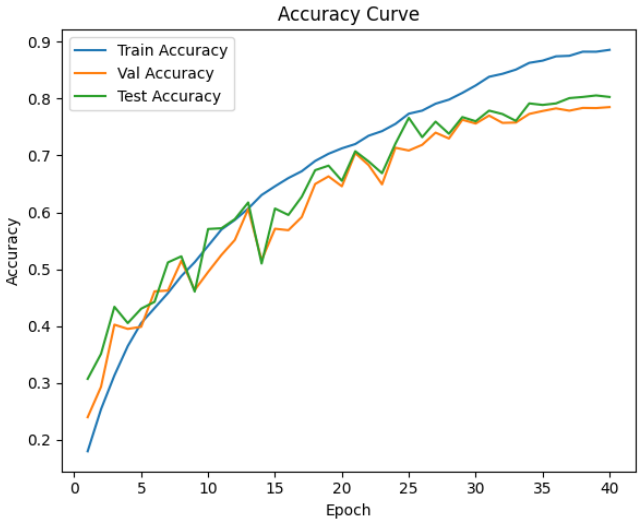
(a) SimpleCNN – Loss Curve



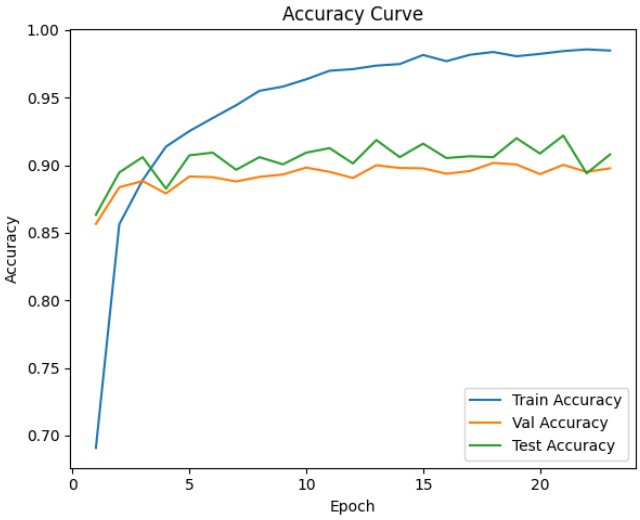
(b) DenseNet121 – Loss Curve

Fig. 1: Training, validation, and test loss curves for both models.

Figure 1 compares the loss curves of both models. The SimpleCNN model shows a gradual and steady decrease in loss across epochs, while DenseNet121 converges much faster and maintains lower loss values across all datasets, despite stopping at an earlier epoch.



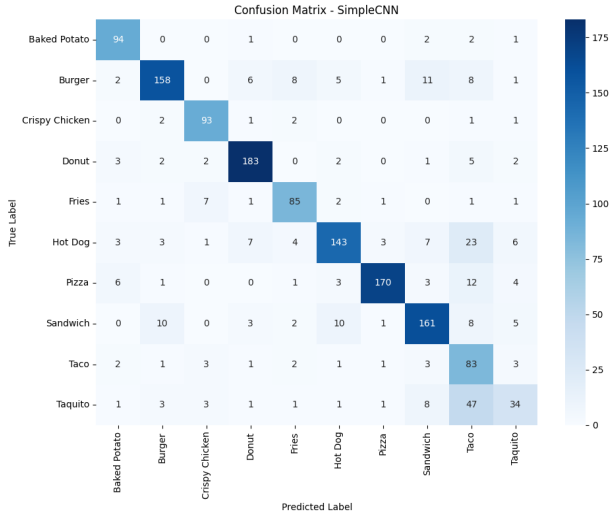
(a) SimpleCNN – Accuracy Curve



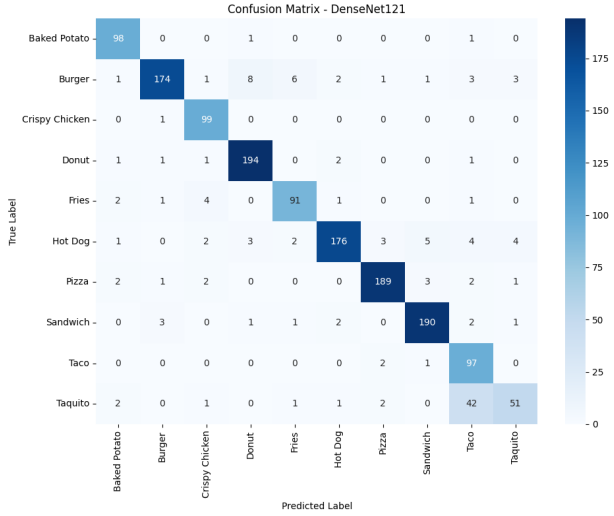
(b) DenseNet121 – Accuracy Curve

Fig. 2: Training, validation, and test accuracy curves for both models.

Figure 2 shows that the SimpleCNN model gradually improves in accuracy but suffers from fluctuations in validation and test sets. DenseNet121 reaches high accuracy more quickly and remains more stable, even though it completed training in fewer epochs due to early stopping.



(a) SimpleCNN – Confusion Matrix



(b) DenseNet121 – Confusion Matrix

Fig. 3: Confusion matrices of SimpleCNN and DenseNet121 on the test dataset.

Figure 3 illustrates that DenseNet121 produces more accurate class-level predictions with fewer misclassifications, particularly for visually similar categories such as Taco and Taquito. In contrast, SimpleCNN struggles more with overlapping class boundaries.

Final Epoch Accuracy Comparison Table

Metric	CNN	DenseNet121
Train Accuracy	0.8855	0.9849
Validation Accuracy	0.7849	0.8977
Test Accuracy	0.8027	0.908

Fig. 4: Final epoch accuracies of CNN and DenseNet121 across training, validation, and test datasets.

VII. Discussion and Analysis

The experimental results reveal clear differences in the behavior and performance of the two models. DenseNet121 consistently outperforms SimpleCNN in terms of accuracy, stability, and generalization ability. While both models showed improvements during training, the gap between them became more visible in validation and test results.

DenseNet121 achieved higher accuracy at earlier stages of training and maintained that performance with minimal fluctuations. This was largely supported by the use of early stopping, which automatically terminated the training process at epoch 23. By doing so, the model avoided overfitting and maintained robust performance across unseen data. In contrast, SimpleCNN completed the full 40 epochs but still showed instability, especially in validation accuracy. This suggests that SimpleCNN is more sensitive to overfitting and less effective at generalization.

When we look at the loss curves, SimpleCNN demonstrated a slow but steady reduction in loss, while DenseNet121 showed a sharper and faster convergence. The difference in performance is also reflected in the confusion matrices. DenseNet121 made fewer classification errors and handled visually similar classes such as Taco and Taquito more effectively. SimpleCNN, on the other hand, struggled more with such overlapping categories, leading to higher misclassification rates.

DenseNet121 Analysis: DenseNet121 proved to be highly effective for this image classification task. Its deep architecture, pretrained weights, and use of early stopping helped it achieve high accuracy in fewer epochs. It remained stable across validation and test sets, showing strong generalization, although it required more computational resources.

SimpleCNN Analysis: Although SimpleCNN did not reach the same level of performance, it is still a lightweight and interpretable model. It may be a practical option for scenarios where hardware is limited. However, its fluctuations in performance and slower convergence indicate that it is less suitable for complex classification tasks without further optimization.

In conclusion, DenseNet121 is the better-performing model for this project. Its ability to reach high accuracy quickly, avoid overfitting with early stopping, and perform reliably across datasets makes it a stronger choice for real-world fast food classification systems.

VIII. Conclusion

In this project, two deep learning models—SimpleCNN and DenseNet121—were implemented and compared for the task of fast food image classification using the Fast Food Classification Dataset (Version 2). The goal was to build a system that could accurately identify different types of fast food from images.

The experimental results demonstrated that DenseNet121 significantly outperformed SimpleCNN in all key aspects, including training speed, accuracy, stability, and generalization. Thanks to its pretrained weights and deeper architecture, DenseNet121 was able to learn faster, avoid overfitting, and produce better predictions across all datasets. In contrast, SimpleCNN was slower to converge and showed noticeable fluctuations during validation and testing.

The confusion matrix analysis further confirmed that DenseNet121 made fewer errors and was more successful in distinguishing between visually similar food items. SimpleCNN, although lighter and more efficient in terms of computation, struggled with class-level accuracy and stability.

Overall, DenseNet121 is a better choice for image classification tasks that require high accuracy and reliability. However, SimpleCNN may still be useful in low-resource environments where simplicity and speed are more important than top-level performance.

This study highlights the importance of model selection in deep learning projects and shows how deeper, pretrained architectures can make a significant difference in classification results. Future work can explore other models, optimize training strategies, or extend this system into real-world applications such as restaurant menu scanners, calorie tracking apps, or smart kitchen devices.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105, 2012.

[2] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, *Densely Connected Convolutional Networks*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2261–2269, 2017.

[3] PyTorch Documentation, *https://pytorch.org/docs/stable/index.html*

[4] Kaggle Dataset, *Fast Food Classification V2*, Available at: `https://www.kaggle.com/datasets`