Selçuk Efe Koçkan | Ahmet Bera Özbolat | Mert Atakan Ümit

22102120 | 21902777 | 21902804

Group 2

CS 484 Introduction to Computer Vision

Term Project Progress Report

Food Image Classification

1. Introduction

This project aims to develop a deep learning model that accurately classifies images of various food items with items that are taken from the Food-101 dataset [1], which comprises 101 distinct food item categories with about 1,000 images per category. The Food-101 dataset is a widely-used benchmark for food image classification tasks, created to provide a challenging dataset with a variety of food categories such as pancakes, pizza, cheescake, guacomole and so on.

Figure 1:FOOD 101 [2]

The classification task poses challenges due to variations in food presentation, lighting conditions, and viewing angles, different times for the food items and so on, an example being the class “Pasta” includes variety of pastas, also with different sauces and settings,with every class similar to this variety. A Convolutional Neural Network (CNN) [3] architecture will be implemented using transfer learning, specifically leveraging a ResNet-50 [4] model for feature extraction to enhance classification accuracy. The dataset is accessible via a Kaggle link provided in the appendices**.**

2. Motivation:

Food image classification has applications in restaurant services, dietary monitoring, and recipe suggestions. By enabling the accurate classification of food images, this project seeks to advance visual recognition tasks, exploring deep learning techniques within a complex and varied dataset. A successful model could contribute to developments in food recognition systems, aiding in industries from food delivery to health tracking applications.

3.Methodology

3.1 Data Pre-processing

In this project, data pre-processing is an essential step to ensure that the images are in a consistent and usable format for training the model. The Food-101 dataset, containing 101,000 images across 101 food

categories, is already split into training and test sets, first the test set is split in half as test and validation sets to allow for a well-rounded evaluation of model performance. To handle varying image dimensions, all images are resized to a standard size of 224x224 pixels, which aligns with the input requirements of the ResNet-50 model used in this project. Padding is applied to maintain aspect ratios across images, ensuring that key visual information is retained regardless of image orientation. Additionally, pixel values are normalized to improve the stability and efficiency of training, enabling the model to converge faster and reducing potential issues with vanishing or exploding gradients. This normalization step also aids in creating a balanced pixel intensity distribution, which enhances model accuracy and consistency across different lighting conditions and food presentations. Through this careful data pre-processing, the dataset becomes optimized for training a deep learning model that is robust to visual variability and can be optimized perfectly to train our algorithm.

3.2 Data Processing and Methodology

The model development for this project centers around using ResNet-50, a pre-trained deep learning model on ImageNet [5], as the backbone for feature extraction. Transfer learning is applied, where ResNet-50's extensive prior training on diverse visual data enables it to capture essential visual features from the Food-101 dataset. A custom Convolutional Neural Network (CNN) classifier is built on top of ResNet-50, with only the final layers trained specifically on Food-101, while earlier layers retain the weights from ImageNet. This approach allows the model to leverage ResNet-50's robust feature extraction capabilities while fine-tuning it for the unique challenges presented by the food images. To optimize model performance, several hyperparameters are tuned, including the learning rate, batch size, and dropout rate. Adjusting the learning rate allows control over the speed and stability of model convergence, balancing rapid learning with the risk of overshooting minima. Batch size adjustments help manage memory usage and stability during training, and dropout layers are incorporated as a regularization technique to prevent overfitting. This structured model development, combined with hyperparameter optimization, creates a resilient architecture that is better equipped to generalize across the varied and complex images in the Food-101 dataset.

ResNet-50 was chosen as the primary model for this project due to its proven effectiveness and efficiency in handling complex image classification tasks. As a deep Convolutional Neural Network (CNN) with 50 layers, ResNet-50 introduces a unique architecture that incorporates residual connections, allowing the model to overcome the "vanishing gradient" problem commonly faced by deeper networks. These residual connections enable the model to maintain strong performance by effectively learning both shallow and deep features, which is particularly advantageous when dealing with the significant intra-class variability and inter-class similarity present in the Food-101 dataset.

ResNet-50's pre-training on ImageNet, a large and diverse dataset, has equipped it with a strong foundation in recognizing general features across various objects and scenes. By employing transfer learning, we leverage this foundation, adapting ResNet-50 to the specific challenges of food classification with only minimal training required on the Food-101 dataset. Additionally, ResNet-50's 50-layer architecture strikes a balance between depth and computational efficiency, making it manageable to fine-tune on standard hardware while still delivering high accuracy and robustness to variations in lighting, orientation, and food presentation styles.

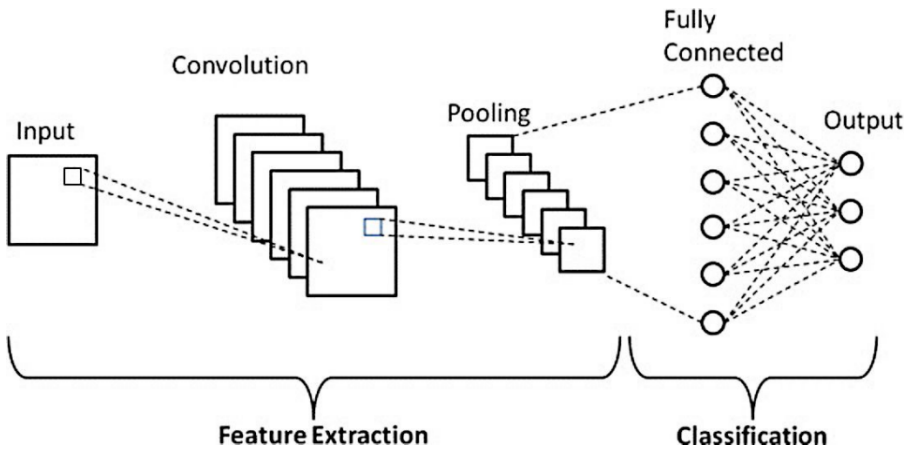

Figure 2: Feature Extraction [6]

4. Implementation

As it is stated in the previous part, Convolutional Neural Network is used for the implementation of the project, ResNet-50 to be exact. CNNs use a series of layers to automatically learn and extract features from input images. The key components of CNNs are convolutional layers, which apply filters to detect patterns like edges and textures, and pooling layers, which reduce the spatial dimensions, making the network more efficient and robust to variations in the input. At the end of the network, fully connected layers combine these extracted features to make the final classification or prediction. This layered approach allows CNNs to effectively handle complex image data, achieving high accuracy in tasks such as image recognition and classification. As it is explained previously and seen by the image, CNN consists of 2 different stages, feature extraction and Classification.

In the Feature Extraction stage, the input image is processed through several layers. The Convolution layers apply filters to capture essential features like edges, textures, and shapes by sliding small receptive fields over the input image. This is followed by Pooling layers, which reduce the spatial dimensions of the feature maps, making the computation more efficient and reducing sensitivity to minor variations in the input.

In the Classification stage, the extracted features are passed through one or more Fully Connected layers, where neurons are densely connected. This part of the network processes the abstracted features and outputs the final prediction, categorizing the input image into specific classes based on the learned features. This two-part structure allows CNNs to handle complex image data and perform accurate classifications.

First, we have started with the feature extraction stage. As we have mentioned previously, ResNet-50 algorithm is used. Before implementing the model, first we have reshaped all of the images to 224x224, but in order to do that, zero padding has been done, later resizing has been done to several images in order to make all of them 224x224. The code for it does all of it in one function and makes new folder for these image classes, all of them reshaped. Then with the reshaped data, we split the data into train, test and validation. To start training the data, first part of CNN algorithm is started, the feature extraction.

4.1. Convolutional Layers and Feature Maps

The feature extraction process starts with the convolutional layers of ResNet-50. These layers apply a series of filters or kernels to the input image. Each filter slides over the image and performs a convolution operation, capturing local patterns such as edges, corners, and textures. This operation generates feature maps—grids that represent the presence of certain features in specific spatial locations in the image. As the image progresses through deeper layers of the network, the convolutional layers detect increasingly complex features, moving from simple edges and lines in the first layers to more abstract shapes, objects, and textures in the deeper layers.

4.2. Pooling Layers for Dimensionality Reduction

After each set of convolutional layers, the network uses pooling layers to reduce the spatial dimensions of the feature maps. Pooling layers perform operations like max pooling, which selects the maximum value from a small window within each feature map, reducing the amount of data while preserving the most prominent features. This process not only makes the model more computationally efficient but also adds a level of invariance to slight changes in the image, such as shifts or rotations. The pooling layers help in creating compact, informative feature representations that can be effectively used for classification.

4.3. Feature Extraction Outputs

Once the input image has passed through the convolutional and pooling layers of ResNet-50, the resulting feature maps provide a condensed representation of the image, focusing on the most critical visual details.

5. Further Implementation

After the feature extraction stage, the outputs are stored in a compressed file for further calculations in the classification part. For our project, we utilize these feature maps as input for further classification. The extracted features are passed to a custom classification head (fully connected layers) that we add on top of ResNet-50, which can be basically called as transferred learning will be implemented for our model starting from the middle, while shifting from feature extraction to classification part. These final layers on our model will be trained specifically on the Food-101 dataset to classify each image into one of the 101 food categories.

As it is stated, we will implement our own custom classification method instead of relying on the fully connected layers of ResNet-50. This approach allows us to have greater control over the network architecture, enabling us to fine-tune specific parameters and adapt the model to the unique challenges for our classification problems presented by the Food-101 dataset. We will try to find optimised number of layers (k layers) for the perfect classification with less complexity and computational cost, also change learning rate for accuracy and also batch size, to construct the neurons and layers for our custom network. By adjusting these variables, we aim to achieve optimal performance in classifying food images with high accuracy and we will also try to achieve low computational costs, as aforementioned.

During training our custom classifier, we will use the Adam optimizer [7] due to its efficiency in handling sparse gradients and adaptive learning rates. To prevent overfitting, we will employ dropout layers, which randomly deactivate neurons during training, ensuring that the model generalizes better to unseen data. Additionally, we will implement early stopping as a regularization technique, which halts training when the model's performance ceases to improve, thus preventing overfitting. By carefully experimenting with different values of learning rate and batch size—key hyperparameters in our

model—our goal is to identify an optimal configuration that balances speed, stability, and accuracy. This tailored approach will allow us to construct a highly adaptable and efficient classification model for food image recognition.

To finally measure the performance of our food image classification model on the Food-101 dataset in the future, we will implement a combination of quantitative metrics and visual analysis tools. Specifically, we will utilize the confusion matrix and an F1 score [8] to measure precision and recall across all 101 different food categories. The confusion matrix will provide a detailed overview of the model's correct and incorrect predictions for each class, helping us identify patterns of misclassification. The F1 score, basically being combination of precision and recall, offers a balanced evaluation metric, especially important in datasets with class imbalances and having large outputs such as our food 101 dataset. This will allow us to gauge how well the model distinguishes between different food items and highlight any categories that may require additional attention or data augmentation.

In addition to these metrics, we will generate accuracy and loss graphs during the training process. By plotting training and validation accuracy and loss over epochs, we can visualize the model's learning progression. These graphs will help us detect overfitting or underfitting tendencies by revealing discrepancies between training and validation performance. For instance, if the training accuracy continues to improve while validation accuracy stagnates or declines, it may be the indicate of overfitting of our model for our specific train data. Conversely, if both training and validation accuracies are low, the model might be underfitting. Monitoring these trends will enable us to make informed adjustments to the model architecture, hyperparameters, or training procedures to enhance overall performance.

References

[1] R. Sarkar, "The Food 101 Data Set," Kaggle. [Online]. Available: https://www.kaggle.com/datasets/ranitsarkar01/the-food-101-data-set. [Accessed: Nov. 14, 2024].

[2] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – Mining Discriminative Components with Random Forests," ETH Zurich Computer Vision Laboratory. [Online]. Available: https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/. Accessed: [Nov. 14, 2024]

[3] S. M. Idelbayev, "A comprehensive guide to convolutional neural networks — the ELI5 way," Towards Data Science, Dec. 15, 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: Nov. 14, 2024].

[4] Viso.ai, "ResNet (Residual Network): Tutorial on ResNet Architecture." [Online]. Available: https://viso.ai/deep-learning/resnet-residual-neural-network/. [Accessed: Nov. 14, 2024].

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.

[6] K. Bansal and A. Singh, "Development of VGG-16 Transfer Learning Framework for Geographical Landmark Recognition," Int. J. Comput. Appl., vol. 175, no. 25, pp. 799–810, Jan. 2023. [Online]. Available: [Accessed: Nov. 14, 2024].

[7] Built In, "Adam optimization algorithm: a method for stochastic optimization," Built In, Jun. 28, 2023. [Online]. Available: https://builtin.com/machine-learning/adam-optimization. [Accessed: Nov. 14, 2024].

[8] S. Terry, "F1 Score: A Simple Explanation," V7, May 18, 2022. [Online]. Available: https://www.v7labs.com/blog/f1-score-guide. [Accessed: Nov. 14, 2024].