Recipe application based on food classification

1st Lennard Hurst
Fakultät Informatik
Hochschule Furtwangen
Furtwangen, Deutschland
lennard@blank-hurst.de
275593

2nd Lara Maria Meister
Fakultät Informatik
Hochschule Furtwangen
Furtwangen, Deutschland
lara.maria.meister@gmail.com
274416

3rd Jonathan Rissler
Fakultät Informatik
Hochschule Furtwangen
Furtwangen, Deutschland
jonathan.rissler@gmx.de
275736

***Abstract*—This paper presents a recipe application, utilizing AI trained on the extensive Food-101 dataset. This application identifies dishes from user-uploaded photos and provides detailed recipes and cooking instructions sourced from a vast collection. The application leverages deep learning models, including convolutional neural networks (CNNs) and transformer-based models, to achieve high accuracy in food classification tasks. This study details the methodology, data preprocessing, model training, and API integration, showcasing the practical implementation and performance of the system. Furthermore, Gustar.io, the API, used for recipe retrieval, is compared with other popular APIs such as Spoonacular and EDAMAM, highlighting the strengths and weaknesses of each. The paper concludes by discussing the results, potential use cases, and future prospects for the application, emphasizing its potential to make cooking more accessible and inspiring for users.**

The executable application:

https://huggingface.co/spaces/larimei/foodrecipe-ai

GitHub Repository of the Project:

https://github.com/larimei/foodrecipe-ai

***Index Terms*—Computer Vision, Deep Learning, Food Classification, Food101, Recipe Application**

I. Introduction

In today's world, convenience in cooking and good meals are highly valued. This trend has resulted in a surge of food-related images online [1]. The application implements image recognition technology to identify these dishes from pictures and shows appropriate recipes, addressing common challenges like meal planning. By leveraging artificial intelligence and machine learning, the app quickly provides recipe suggestions based on the identified dish, saving time and effort for users. This solution makes cooking accessible and enjoyable, inspiring people with new meal ideas.

This paper is structured as follows. The related work section reviews existing recipe applications and food recognition research. The methodology details the chosen approaches and base architectures. It covers the data and preprocessing steps, followed by the training and evaluation of models. The API integration is explained, followed by an overview of the application implementation as a whole. The results and discussions are presented, highlighting use cases and future prospects. Finally, conclusions and implications of the study are provided.

II. Related Work

A. Overview of Existing Recipe Applications

The integration of textual, structural, and nutritional information for recipe representation has been explored using advanced neural network architectures. Studies such as those by Tian et al. (2021) investigate the utilization of textual CNN and transformers to create comprehensive recipe representations, showcasing the potential for neural networks to enhance the understanding and generation of recipe content [2].

Further advancements in the field have been made through the development of systems that predict recipes from images using CNN and Transformer models. Li et al. (2023) proposed a novel approach that combines these models to enhance the portability and efficiency of recipe prediction, incorporating knowledge distillation techniques to improve model performance [3].

B. Current Research on Food Recognition and Classification

The application of CNNs for food image classification has seen significant progress, particularly through the use of transfer learning techniques. Patil et al. (2021) demonstrated the effectiveness of using CNN architectures such as ResNet-50 and InceptionV3 for this purpose. Their research highlights how transfer learning can be leveraged to achieve high accuracy in food image classification tasks [4].

Comparative studies of various deep pre-trained CNN architectures have further illuminated the capabilities of different models in classifying food images. Singh et al. (2023) conducted a thorough comparison of models like EfficientNet and Xception, showcasing their relative effectiveness and establishing benchmarks for future research [5].

A comprehensive survey by Plested et al. (2022) reviews the state of deep transfer learning in image classification, with a particular focus on its applications in food image classification. This survey underscores the advancements and ongoing research in utilizing deep learning for food recognition, providing a broad overview of current methodologies and their efficacy [6].

C. Relevance and Importance of the Food101 Dataset

The Food101 dataset has emerged as a critical resource in the field of food recognition and classification [7]. Its extensive collection of 101,000 images across 101 categories provides a

robust benchmark for evaluating image classification models. The diversity and large number of categories in the dataset make it particularly valuable for training and testing deep learning models, as evidenced by its widespread use in recent research. The studies mentioned above frequently utilize the Food101 dataset, underscoring its significance in advancing the state of food image classification.

Overall, the Food101 dataset plays a pivotal role in the ongoing development of more accurate and efficient models for food recognition, as demonstrated by its integration into various research methodologies and the substantial improvements in model performance that it facilitates.

III. Methology

A. Overview of the Chosen Approaches

This project developed a recipe application based on food classification using deep learning models. The power of convolutional neural networks (CNNs) and transformer-based models is leveraged for image classification tasks, implementing models using PyTorch and Hugging Face libraries. The approach consisted of training and evaluating models locally and on Google Colab to ensure flexibility and scalability. Utilizing the Food101 dataset, a well-known benchmark for food classification, the models are trained and tested. PyTorch has been shown to achieve high accuracy rates in food type recognition [8].

B. Description of Base Architectures

Several architectures known for their effectiveness in image classification tasks were employed, including ResNet-50, Google ViT (Vision Transformer), and EfficientNet.

- ResNet-50: A deep residual network with 50 layers, utilizing skip connections to mitigate the vanishing gradient problem, allowing for training of deeper networks [9].
- Google ViT: A transformer-based model that applies the transformer architecture to image classification, excelling in capturing long-range dependencies in images [10].
- EfficientNet: A model family that scales the width, depth, and resolution of networks efficiently, providing high performance with lower computational costs [11].

These architectures were chosen due to their balance between depth, computational efficiency, and performance, making them suitable for the food classification task. The benefits of employing these architectures for improved classification accuracy are well-documented [12].

C. Local Models with PyTorch

For the local training of models, PyTorch was used. Training locally with PyTorch offered shorter runtimes due to the use of a powerful GPU. However, the model trained locally could not be deployed on Hugging Face due to issues with converting the model to a format compatible with Hugging Face. Since the application runs on Hugging Face, this model was not usable in the final application.

D. Models on Google Colab with Hugging Face

To leverage cloud-based resources, initial training of models was conducted on Google Colab using Hugging Face's libraries. The Colab environment provided GPU acceleration, though the free version used a less powerful GPU, resulting in longer runtimes. Additionally, Colab does allow the saving of checkpoints during training. However, long training sessions often led to timeouts after several hours when training for many epochs. This approach allowed for an efficient use of Hugging Face's ecosystem for managing and deploying models. If the training would ran successfully on Colab, the same model was then trained locally on a more powerful GPU to ensure optimal performance. The integration of Hugging Face with PyTorch for efficient model training and deployment is well-supported [13].

E. Local Models with Hugging Face

Using Hugging Face, the Transformers and Datasets libraries were employed for a streamlined training process locally. Training locally with Hugging Face provided shorter runtimes, automated the pushing of models to Hugging Face, and enabled the automatic deployment of the model on Hugging Face Spaces, utilizing local resources effectively. This approach involved loading the Food101 dataset, applying necessary transformations, and training models such as ResNet-50, Google ViT, and EfficientNet using Hugging Face's Trainer API.

IV. Data and Preprocessing

A. Description of the Food101 Dataset

The Food101 dataset is a large-scale dataset designed for food classification tasks. It contains 101,000 images across 101 categories of food, with each category containing 1,000 images. The dataset is split into 75,750 training images and 25,250 test images. The diversity and large number of categories make it an excellent benchmark for evaluating image classification models in the context of food recognition.

B. Data Preparation and Augmentation

To prepare the dataset for training, a series of preprocessing steps were applied using PyTorch's and Hugging Face's data transformation utilities. The preprocessing steps included resizing the images, converting them to tensors, and normalizing them to ensure consistent input to the models. These transformations ensured that the input images were of consistent size and pixel intensity distribution, which is crucial for training deep learning models effectively.

Data augmentation techniques such as random resizing and cropping were also employed to increase the diversity of the training data. This helps improve the model's robustness by exposing it to a variety of image conditions during training [14].

C. Data Splitting and Validation

The dataset was split into training and validation sets to evaluate the model's performance and avoid overfitting. The Food101 dataset inherently provided predefined training and test splits, which were used directly for model evaluation. Additionally, a portion of the training data was set aside for validation purposes during training.

By setting aside a portion of the data for validation, the model's performance on unseen data could be monitored during the training process and adjustments to the model or training parameters could be made as necessary to improve generalization. This approach helped ensure that the final model would perform well not only on the training data but also on new, unseen data in real-world applications.

V. Model Training and Evaluation

A. Training Models with PyTorch

Training models with PyTorch involved setting up a local environment and leveraging its flexibility for deep learning. With the Food101 dataset the data preprocessing steps are implemented including resizing, normalization, and data augmentation. ResNet-50, Google ViT, and EfficientNet models are trained, fine-tuning them for food classification. The training loop included forward passes, loss computation using cross-entropy, backpropagation, and optimization using stochastic gradient descent (SGD) and Adam optimizers.

B. Training Models on Google Colab with Hugging Face

Training models on Google Colab followed a similar process to the local PyTorch implementation but took advantage of Colab's free GPU resources for training. This environment allowed to run ResNet-50, Google ViT, and EfficientNet models, although the free version of Colab used a less powerful GPU, resulting in longer runtimes compared to local training.

Using Hugging Face locally, the Transformers and Datasets libraries were employed for a streamlined training process. The Food101 dataset was split into training and testing sets, with necessary transformations applied. The trained ResNet-50, Google ViT, and EfficientNet models using Hugging Face's Trainer API simplified the training loop by handling data loading, forward passes, and optimization internally. Training locally with Hugging Face provided shorter runtimes, automated the pushing of models to Hugging Face, and facilitated the automatic deployment of the model on Hugging Face Spaces, using local resources effectively.

C. Comparison of Training Environments and Methods

Each training environment and method offered distinct advantages. PyTorch provided flexibility and control over the training process, making it suitable for customized model tuning. Google Colab's cloud-based environment with GPU support, though slower due to the free version's less powerful GPU, was useful for initial experimentation and validation of the model configuration. Hugging Face offered a high-level interface with pre-trained models and a simplified training loop, enhancing productivity and ease of use. Training locally with Hugging Face allowed for the efficient use of local resources, shorter runtimes, and automated deployment. The final workflow involved initial testing on Colab with Hugging Face, followed by comprehensive training on local GPUs once the model configuration was validated.

D. Evaluation Metrics and Results

Models were evaluated using accuracy as the primary metric, calculated as the ratio of correctly classified images to the total number of images. Additionally, loss values during training were tracked to monitor convergence. Results showed that models trained with PyTorch locally and on Google Colab achieved comparable accuracies. The Hugging Face models also performed well, benefiting from the robust training pipeline provided by the Trainer API.

Among the models, Google ViT exhibited superior performance in the running application due to its advanced transformer-based architecture, which excels in capturing intricate details in images. EfficientNet provided a good balance between accuracy and computational efficiency, while ResNet-50 served as a reliable baseline.

Overall, the choice of environment and method depended on the specific requirements of flexibility, speed, and ease of use. Detailed evaluation results and metrics are discussed in the Results section.

E. Training Procedure

The final model used in the application was trained using a structured approach to ensure optimal performance. Initially, the training was conducted on Google Colab to validate the model configuration. Once the training procedure successfully completed on Colab, the entire training process was repeated on a local GPU to leverage more powerful hardware and achieve better performance.

F. Training Hyperparameters

The following hyperparameters were used during training:

- Learning rate: 5×10^{-5}
- Train batch size: 16
- Eval batch size: 16
- Gradient accumulation steps: 4
- Total train batch size: 64
- Optimizer: Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$
- LR scheduler type: Linear
- LR scheduler warmup ratio: 0.1
- Number of epochs: 5

G. Training Results

The training results for the Google ViT model, fine-tuned on the Food101 dataset, are summarized in the table below:

Training Loss	Epoch	Step	Validation Loss	Accuracy
2.0684	1.0	947	2.0226	0.7242
1.1222	2.0	1894	1.1585	0.7774
0.7885	3.0	2841	0.9156	0.8013
0.6670	4.0	3788	0.8209	0.8145
0.6160	5.0	4735	0.7886	0.8175

TABLE I: Training and Validation Results

The model is a fine-tuned version of "google/vit-base-patch16-224-in21k" trained on the Food101 dataset. It achieves the following results on the evaluation set: a loss of 0.7886 and an accuracy of 81.75%.

VI. API Integration

In this chapter, the Gustar.io API is explained, which is used in the project for classifying and retrieving recipes based on recognized food labels. It is compared to APIs from Spoonacular and EDAMAM, showing the strengths and weaknesses of each, and explained why Gustar.io was selected for this project.

The APIs under consideration are:

- Gustar.io: An API for recipe classification and retrieval based on recognized food labels using artificial intelligence [15].
- Spoonacular: An API offering a wide range of functions, including nutritional information, recipe search, and food classification [16].
- EDAMAM: An API focused on health and nutritional information, providing extensive data on recipes and foods [17].

To make a thorough comparison, the APIs are evaluated based on several criteria: functionality, quality, variety of provided recipes, user-friendliness and documentation.

A. Functionality

Gustar.io offers functions for recipe search, where an artificial intelligence generates a recipe based on food labels. Spoonacular has a broad range of features, including nutritional information, recipe search and shopping list generation. EDAMAM, on the other hand, emphasizes nutritional and health information, supported by an extensive recipe database.

B. Quality and Variety of Provided Recipes:

Gustar.io offers a solid selection of recipes well-aligned with the recognized food. Spoonacular boasts a very extensive recipe database with great variety and numerous categories. EDAMAM focuses on healthy and nutrition-conscious recipes, providing good quality and diversity.

C. User-Friendliness and Documentation:

Gustar.io is well-documented and user-friendly, particularly for specific use cases like food recognition. Spoonacular has excellent documentation, is user-friendly, and includes many examples and tutorials. EDAMAM features outstanding documentation with clear instructions, although the complexity might be overwhelming for new users.

The choice of API depends heavily on the specific requirements of the project. For this project, which focuses on the precise classification of foods and the subsequent recipe search, Gustar.io offers the best conditions. Its functions with artificial intelligenc make Gustar.io the optimal choice to get individual recipes. However, Spoonacular and EDAMAM offer more extensive functions and a greater depth of these recipes, making them appealing for other use cases.

VII. Application Implementation

The implementation of this recipe application uses image classification and a user-friendly interface to enhance the experience for users. Central to the application is the pre-trained image classification model based on the Food-101 dataset, which identifies various dishes from uploaded photographs. This process is hosted on Hugging Face and integrated into a Gradio interface.

The architecture supports a workflow where users upload or capture an image, which is then processed by the model. The model, stored in a Hugging Face repository, classifies the image and returns the dish name with the highest probability. This classification triggers a POST request to a recipe database API, including necessary parameters such as the meal label, content type, API key, and host address. Within seconds, the API responds with the recipe information, including the recipe title, ingredients, and preparation instructions. These details are parsed and displayed to the user in a text field on the website.

Gradio's interface allows real-time interaction, making the application practical for everyday use. Designed for ease of use, the interface includes example images to help users understand and utilize the application. This design ensures a seamless user experience from image upload to recipe display, supporting a practical and efficient workflow 1.

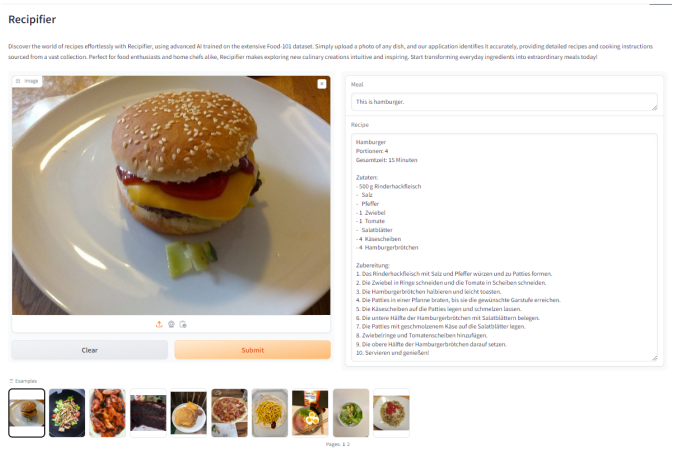

Fig. 1: UI of the Application

VIII. Results and Discussion

A. Presentation of Results

The trained models demonstrated high accuracy and efficiency in classifying food images from the Food101 dataset. The Google ViT model, fine-tuned on the dataset, achieved a final validation accuracy of 81.75%, with a loss of 0.7886. The model's performance was consistent across various food categories, indicating robust generalization capabilities.

B. Analysis of Model Performance

The performance analysis indicated that the Google ViT model excelled in accurately classifying most food images. The model's ability to immediately recognize and classify dishes highlights its potential for real-world applications. The high accuracy rate suggests that the model can reliably assign the correct class to example images, enhancing its usability for everyday mobile use. Users can upload their photos, and the model provides quick and accurate classifications, making it a valuable tool for food recognition tasks.

C. Discussion of Advantages and Disadvantages of Different Approaches

The implementation of different training environments and methods revealed several advantages and disadvantages:

1) Advantages:

- Quick Recognition: The model's fast recognition capability allows for immediate identification of dishes, making it highly efficient for real-time applications.
- High Accuracy: The model consistently assigns images to the correct class, demonstrating a high level of reliability in classification tasks.
- Mobile Usability: The responsive nature of the model allows users to upload their photos and get instant results, making it practical for daily use.

2) Disadvantages:

- Class Confusion: Similar classes, such as Ramen, Pho, and Bibimbap, are sometimes confused, indicating a limitation in distinguishing between closely related food items.
- Incomplete Coverage: Not all dishes are represented in the dataset, leading to potential gaps in the model's recognition capabilities.
- Limited Requests: The API has a restricted number of requests, which can be a constraint for extensive use.
- Non food recognition: The model always selects the class with the highest probability, even when the image does not even contain food. This behavior highlights the need for incorporating a mechanism to filter out non-food images to improve overall accuracy and user satisfaction.

.

IX. Use Cases

The recipe Application can be used in many ways including the following use cases:

A. Discovering New Recipes:

Users can upload a photo of any dish they encounter, whether at a restaurant or a friend's house, and the application will identify the dish and provide detailed recipes. This allows user to recreate their favorite meals at home and making it easy to try new cooking techniques and ingredients.

B. Meal Planning and Preparation:

By uploading photos of dishes, users can receive recipe suggestions that help them plan meals and make the most of the ingredients they have on hand. This reduces food waste and makes meal planning more efficient.

C. Educational Tool:

Users can use the application as a learning tool to identify dishes and understand their preparation methods, enhancing their educational experience.

X. Future Prospects

The future prospects for the app are promising, with several potential enhancements and expansions:

A. Expanded Dataset and Improved Accuracy:

By incorporating more diverse and comprehensive datasets, the AI could improve its accuracy in identifying dishes from various cuisines and cultural backgrounds.

B. Enhanced User Experience:

Utilizing user preferences and past interactions, the application could offer personalized recipe suggestions tailored to individual tastes and dietary restrictions.

C. Recognizing individual ingredients:

The application could recognize all individual ingredients in the image and show them in the recipe. Therefore alterations to a dish could be shown and taken into account.

XI. Conclusion

The recipe application utilizing deep learning models and the Food101 dataset has demonstrated strong capabilities in food image classification and recipe suggestion. By using CNNs and transformer-based models like Google ViT and EfficientNet, it achieved high accuracy rates in identifying diverse food categories from images.

The application's implementation on platforms like Hugging Face Spaces showcases the scalability and deployment flexibility of deep learning models in real-world applications. Despite challenges such as API limitations and occasional classification errors, the approach showcases AI's potential to simplify meal planning and inspire exploration. Future enhancements could focus on dataset diversity, recognizing individual ingredients in the image and enhancing user engagement with personalized recommendations and interactive features.

REFERENCES

[1] G. Amato, P. Bolettieri, V. M. de Lira, C. I. Muntean, R. Perego, and C. Renso, "Social media image recognition for food trend analysis," *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017.

[2] Y. Tian, C. Zhang, R. A. Metoyer, and N. Chawla, "Recipe representation learning with networks," *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

[3] Z. B. Li, "Predict food recipe from image using cnn/transformer & knowledge distillation for portability," *2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE)*, pp. 852–859, 2023.

[4] P. Patil and V. C. Burkapalli, "Food cuisine classification by convolutional neural network based transfer learning approach," *2021 IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, pp. 1–5, 2021.

[5] P. K. Singh and S. Susan, "Transfer learning using very deep pretrained models for food image classification," *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1–6, 2023.

[6] J. Plested and T. Gedeon, "Deep transfer learning for image classification: a survey," *ArXiv*, vol. abs/2205.09904, 2022.

[7] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *European Conference on Computer Vision*, 2014.

[8] I. Iswahyudi, D. Hindarto, and H. Santoso, "Pytorch deep learning for food image classification with food dataset," *sinkron*, 2023.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[11] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020. [Online]. Available: https://arxiv.org/abs/1905.11946

[12] W. Zuo, W. Zhang, and Z. Ren, "Food recognition and classification based on image recognition: A study utilizing pytorch and prenet," *2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA)*, pp. 1325–1330, 2023.

[13] J. Castaño, S. Martínez-Fernández, X. Franch, and J. Bogner, "Exploring the carbon footprint of hugging face's ml models: A repository mining study," in *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, Oct. 2023. [Online]. Available: http://dx.doi.org/10.1109/ESEM56168.2023.10304801

[14] P. Sengupta, A. Mehta, and P. Rana, "Enhancing performance of deep learning models with a novel data augmentation approach," *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1–7, 2023.

[15] "A Recipe API with 2 Mllion Recipes - Recipe Search API - Edamam." [Online]. Available: https://developer.edamam.com/edamam-recipe-api

[16] D. Urbansky, "spoonacular recipe and food API." [Online]. Available: https://spoonacular.com/food-api/docs

[17] Tümer, "Gustar.io API service," 6 2024. [Online]. Available: https://gustar.io/apis/