University Leipzig
Faculty of Economics and Management Science
Institute for Business Information Systems

Seminar paper on the topic

Classifying cafeteria meals using Machine Learning at University of Leipzig

Supervising university lecturer:	Prof. Dr. Patrick Zschech
Authors:	Nicas Frank (3778296), Elias Teoman Eroglu (3787582), Joshua Nerling (3772226), Hannes Simon (3758539), Benedikt Schuster (3783517)

Submitted on: 31.01.2025

Structure

List of Illustrations

List of Tables

List of Listings

List of Abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Network
FLOPs	Floating Point Operations
Grad-CAM	Gradient-weighted Class Activation Mapping
KPMG	Klynfeld-Peat-Marwick-Goerdeler
LIME	Local Interpretable Model-Agnostic Explanations
ML	Machine Learning
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
UI	User Interface
XAI	Explainable Artificial Intelligence

1 Introduction

The integration of Artificial Intelligence (AI) into daily operations has witnessed a surge in popularity, with self-checkout systems becoming a staple in the retail and hospitality sectors, according to the latest Consumer Barometer from Klynfeld-Peat-Marwick-Goerdeler (KPMG). These systems promise enhanced efficiency, cost savings, and improved customer experience (KPMG Consumer Barometer, 2024). In Leipzig, the Studentenwerk managing university cafeterias face unique challenges, including staff shortages and budget constraints (Studentenwerk Jahresbericht, 2023). These pressing issues underscore the urgent need for innovative solutions that streamline operations while addressing financial and operational limitations.

In this context, the potential of AI, particularly Machine Learning (ML), to transform service delivery is profound. By leveraging Convolutional Neural Networks (CNNs), which are renowned for their ability to classify and process visual data, it becomes feasible to automate the identification of cafeteria meals. This capability has the potential to revolutionize the checkout process, making it faster and less dependent on human resources. At the same time, such automation supports broader societal goals by enhancing transparency. Customers can access detailed information about allergens, nutritional values, and ingredients in real time, enriching their overall dining experience (Shakir, 2024).

Building on these economic and societal benefits, the adoption of this technology offers substantial advantages for cafeteria operations. Automating meal recognition not only reduces reliance on manual input but also accelerates transactions, effectively shortening queues during peak hours. This efficiency minimizes operational costs and enables staff to focus on other critical tasks. For students and staff, these improvements contribute to a more pleasant and efficient dining environment. From a technological perspective, implementing such an AI-driven system exemplifies how ML can solve practical, real-world problems, setting a precedent for similar applications across other sectors.

Considering these considerations, this paper will investigate the potential of Artificial Intelligence to optimize the payment process in university cafeterias for both Studentenwerk and customers. This investigation will employ ML and CNNs. By addressing this question, the study aims to provide a comprehensive analysis of the benefits, limitations, and practical implications of deploying such a solution in a real-world setting.

2 Business Case

The implementation of self-checkout systems in various sectors has demonstrated significant advantages in terms of operational efficiency and customer satisfaction. Originally introduced in retail environments, these systems have now found applications in other service-oriented domains. Given the increasing demand for automation and the persistent issue of labor shortages, self-checkout technology presents an opportunity to optimize cafeteria operations while enhancing the overall dining experience for students.

2.1 Current State of the Studentenwerk Leipzig Cafeterias

The Studentenwerk Leipzig operates a total of ten cafeterias across various university locations, providing essential dining services to the student population. These cafeterias serve over 40,000 students from multiple institutions, including Leipzig University and other affiliated universities. In 2023, a total of 1,727,740 meals were sold over all cafeterias. According to the 2023 annual report, the cafeteria division generated revenues of approximately 7.7 million euros, a 6% increase from pre-pandemic levels. However, financial sustainability remains a concern, as the cafeterias required state subsidies. Staffing shortages have been a persistent issue. The report highlights that the number of employees in the Studentenwerk remained relatively stable, averaging around 316 employees across all divisions, including cafeterias. Nevertheless, recruitment challenges have led to operational limitations. Presently, the execution of the payment process is conventionally conducted manually by employees during the checkout procedure (Studentenwerk Jahresbericht, 2023).

2.2 The Case for Self-Checkout Systems in University Cafeterias

This section discusses the relevance of self-checkout systems for university cafeterias by evaluating their benefits from both an operational and a customer perspective.

2.2.1 Addressing Operational Challenges in University Cafeterias

University cafeterias, particularly those managed by organizations such as the Studentenwerk Leipzig, have faced considerable operational difficulties in recent years. The 2023 annual report of the Studentenwerk Leipzig highlights the persistent challenge of staffing shortages. The report states that, despite a 6% increase in revenue compared to pre-pandemic levels, personnel recruitment remained a pressing issue throughout the year.

„Hohe Auslastung in den Mensen und Cafeterien erforderte eine verstärkte Wiederbesetzung der Stellen, wobei die Bewerber:innenlage in diesem Bereich überschaubar war, so dass der Personalbedarf nicht in voller Höhe gedeckt werden konnte.“ (Studentenwerk Jahresbericht 2023, S.26)

The difficulty in filling vacancies resulted in the closure of certain facilities, such as the evening dining service at Mensa am Park and the Bistro in the same location, as well as reduced hours for other cafeterias. Moreover, the report underscores the financial constraints faced by the Studentenwerk, which required a state subsidy to maintain cafeteria operations (Studentenwerk Jahresbericht, 2023). These financial dependencies highlight the necessity for cost-saving innovations such as self-checkout systems, which can alleviate some of the economic pressure by reducing labor costs and reallocating human resources to other essential functions.

2.2.2 The Benefits of Self-Checkout for Students

From a customer perspective, self-checkout systems provide numerous advantages, including reduced waiting times, enhanced convenience, and greater autonomy in the dining experience. Recent research from 2024 suggests that self-checkout technology significantly enhances transaction efficiency and overall customer satisfaction in retail environments (Shakir, 2024). These findings are applicable to university cafeterias, where peak dining hours often lead to long queues and extended waiting times. By implementing self-checkout kiosks, students can complete their transactions more quickly, leading to a smoother and more efficient dining process.

Additionally, self-checkout technology aligns with broader consumer trends emphasizing digital convenience. The KPMG Consumer Barometer reports that 82% of consumers who regularly use self-checkout systems rate their shopping experience positively. Moreover, the study highlights that younger consumers, such as university students, are particularly receptive to self-service technologies, with 71% of individuals under the age of 24 frequently using self-checkout systems in retail environments (KPMG Consumer Barometer, 2024). These statistics suggest that students would likely embrace similar solutions in cafeteria settings, further justifying the implementation of automated payment systems in university dining facilities.

Considering the operational challenges faced by university cafeterias and the clear advantages for students, the integration of self-checkout systems appears to be a well-founded investment. From an economic perspective, these systems help address staffing shortages and reduce financial dependencies by optimizing resource allocation. Simultaneously, they enhance customer satisfaction by expediting the checkout process and catering to the digital preferences of a tech-savvy student population. In light of these benefits, self-checkout systems present a viable solution to the growing demands of university dining services.

2.3 Challenges in Classifying Cafeteria Meals

The implementation of self-checkout systems in university cafeterias relies heavily on automated meal recognition technology. However, several challenges arise when attempting to accurately classify cafeteria meals using machine learning models.

One of the primary difficulties is the variability in dishes, as cafeteria menus change frequently, and the same meal can be presented differently depending on portioning and plating. This inconsistency complicates classification, as a single meal type may have multiple visual representations. Additionally, environmental factors such as variations in lighting conditions, camera angles, and occlusions, such as side dishes partially covering main meals, can negatively impact the accuracy of image recognition models.

Another critical challenge is data scarcity. Unlike widely studied datasets for common objects, high-quality labeled datasets for cafeteria meals are limited. This lack of comprehensive data necessitates the use of data augmentation techniques, such as image rotation, flipping, and contrast adjustments, to artificially expand training sets and improve model robustness. Furthermore, class imbalance, where some meals appear significantly more frequently than others, can lead to biased predictions, requiring careful dataset balancing strategies. Beyond dataset limitations, real-time processing is a crucial requirement for self-checkout applications. The classification system must operate with high speed and accuracy to ensure that meal recognition does not introduce delays at checkout. This demands optimized neural network architectures capable of efficient inference while maintaining high precision.

Finally, for widespread adoption, the system must be robust and reliable. The model should not only achieve high accuracy under controlled conditions but also adapt to new or modified dishes over time. Continuous learning mechanisms and periodic model retraining are necessary to sustain performance and minimize misclassifications. By addressing these challenges, a dependable meal classification system can be developed to enhance the efficiency of self-checkout processes in university cafeterias.

2.4 Objectives

The implementation of an AI-powered self-checkout system in university cafeterias requires clearly defined objectives to ensure its efficiency, usability, and reliability. The following key goals have been established to guide the development and deployment of the system.

2.4.1 Automated Identification of Meals via Images

The system must accurately recognize and classify cafeteria meals using computer vision techniques. This is achieved through:

- The application of machine learning models, specifically CNNs, to classify meal images.
- Addressing Classification Challenges by implementation data augmentation techniques, such as rotation, flipping, and Gaussian blur, to improve model generalization.
- Ensuring a robust and reliable system that maintains accuracy under diverse real-world conditions, including varying lighting environments and camera quality

2.4.2 Improved User Experience and Additional Information

Beyond meal recognition, the system aims to enhance the customer experience by providing relevant nutritional and allergen information. Key features include:

- Real-time meal classification, displaying results within milliseconds, ensuring a seamless checkout process.
- A confidence threshold for each classification, allowing the system to indicate its certainty and, if necessary, prompt user verification.
- Displaying nutritional values, allergen warnings, and ingredient details, improving transparency and catering to dietary restrictions.
- Integration of a mobile application, allowing students to scan and pay for their meals directly via their smartphones, reducing reliance on physical self-checkout kiosks.
- Increased efficiency by streamlining the payment process and reducing congestion during peak dining hours.

By addressing these objectives, the AI-based self-checkout system can significantly enhance operational efficiency, reduce waiting times, and provide students with an improved dining experience.

2.5 Solution Approach

To address the inefficiencies in university cafeteria payment processes, this solution introduces an AI-driven self-checkout system that integrates automated meal classification with multiple payment options. By leveraging advanced machine learning techniques, the system enhances operational efficiency, reduces waiting times, and provides students with a seamless dining experience. The solution is based on three key payment methods: traditional checkout, fixed self-checkout kiosks, and mobile self-checkout via an app. Each method is designed to cater to different user preferences and optimize the transaction process.

The traditional checkout remains in place to accommodate customers who prefer conventional payment methods or require assistance. However, to alleviate workload and

minimize congestion, fixed self-checkout kiosks are introduced. These kiosks enable students to independently scan and pay for their meals, utilizing a custom-trained meal classification model for accurate recognition. In addition to kiosks, a mobile self-checkout application offers a more flexible alternative, allowing students to scan their meals via their smartphones and complete transactions without the need for physical kiosks. This approach is particularly beneficial during peak dining hours, reducing bottlenecks and improving service efficiency.

A crucial component of this solution is the AI-powered meal classification system, which enables accurate and rapid identification of cafeteria meals. The classification process begins with the collection and preprocessing of image data, where high-quality images of various cafeteria meals are annotated and structured into a dataset. In order to ensure optimal model performance, three different models were compared: A custom-designed CNN and two EfficientNet model with transfer learning on different datasets. Subsequently the best performing model was implemented.

In conclusion, the proposed solution not only enhances the efficiency of meal classification but also improves the overall user experience. The combination of self-checkout kiosks and a mobile application provides students with multiple convenient payment options, ensuring flexibility and accessibility. At the same time, the AI-driven classification minimizes the need for manual input, reducing labor costs and streamlining cafeteria operations. This represents a significant advancement in the digitalization of university cafeterias. By combining AI-based meal recognition with multiple transaction methods, it addresses key operational challenges while enhancing service efficiency and user convenience. The next sections will further explore the implementation details, performance evaluation, and potential improvements to ensure the long-term success of this approach.

3 Methods

The following section outlines the methodological approach of the project. It is divided into the data collection process and data preprocessing steps. The latter includes aspects such as data splitting, data transformation, and data augmentation methods.

3.1 Data Collection

To provide a customized solution for the meals offered in the university cafeteria, a systematic data collection process was utilized. This approach involved the capture of images of various dishes to create a baseline dataset for training machine learning models. The image capture process was designed to simulate the functionality of a self-checkout machine. Specifically, images were captured vertically from above, simulating the perspective of a camera positioned above a tray of food (see Figure 1). This configuration ensures a realistic depiction of the system's functionality in a real-world setting.

Figure 1: Image Capture Setup with Tripod Simulating Self-Checkout Perspective.

It should be noted that this approach is optimized for the self-checkout machine. The alternative implementation using mobile phone scanning is not optimized for image capture, as further discussed in the limitations section.

The dataset consists of 10 categories, with each category containing 50 images, leading to a total of 500 images. All images were captured using an iPhone 15+ with a resolution of 4284x5712 pixels. The images presented in Figure 2 illustrate the classes 2 “Schwarzbierpfanne vom Rind”, 9 “Putengyros”, and 10 “Alaska-Seelachsfilet in Eihülle”.

Figure 2: Samples of classes 2, 9 and 10.

To increase the robustness of the models during the training process, variability was intentionally introduced into the captured images. The plates containing the food were rotated or repositioned on the tray to capture the food from different angles. In addition, the arrangement of the food on the plates was changed to reflect realistic scenarios in which food presentation might vary. This methodological approach has been shown to enhance the system's capacity for generalization, leading to improved performance in scenarios where the dish arrangement differs from that observed during training or where minor deviations from the training data are present.

3.2 Data Preprocessing

3.2.1 Train-, Validation- and Test-Split

The division of data into distinct sets for training, validating and testing is a fundamental step in the machine learning model development process (Hastie et al., 2009). This approach ensures the effective evaluation of model quality and the avoidance of biases during model training: The training data is used to adjust the model’s parameters, enabling it to capture patterns and relationships within the data (Hastie et al., 2009). Validation data, on the other hand, is employed to evaluate the quality of the model during or after the training process. It serves as a basis for hyperparameter tuning and can regulate the training process through techniques such as early stopping. Test data, also referred to as "hold-out" data, remains entirely separate from the training process and is used exclusively to assess the model’s generalization ability at the end of the training process (Hastie et al., 2009). This step ensures that the evaluation accurately reflects the model’s performance on unseen data.

To implement this process, we divided our dataset into 10 classes, each representing a specific dish. Each class contains 50-60 images, which were further split into training, validation, and test datasets in a 64:16:20 ratio. This approach ensures that there is sufficient data for the model to learn while preserving a portion for assessing the quality of the model's predictions.

3.2.2 Data Transformation and -Augmentation

In the preceding step, three distinct data categories were established. As a component of the data preprocessing pipeline, a range of data transformation and augmentation techniques are employed on these datasets. The objective of data augmentation is to enhance data diversity, thereby enhancing the model's robustness during training (Krizhevsky, A. et al., 2012). These augmentation techniques are exclusively applied to the training data and are not utilized for validation or test data to ensure unbiased model evaluation.

During each training epoch, a “RandomResizedCrop” is implemented on the training data. This method randomly extracts a cropped region from each sample, which is then resized to a fixed size. In this work, a random crop that retains 50% to 100% of the original image size was selected, and the extracted region is resized to 224×224 pixels. Additionally, all training samples undergo a “RandomHorizontalFlip” transformation. This operation involves the random flipping of an image horizontally with a 50% probability, thereby introducing further variability in the dataset. Furthermore, a “Gaussian Blur” is applied to all training samples, a method that introduces random blurring with varying intensity, simulating natural variations in sharpness and helping the model generalize better. Since these transformations are performed during every training epoch, the total number of image variations is determined by the product of the number of epochs and the number of training images. For instance, with 10 training epochs and a training dataset comprising 50–60 images (64% of the dataset), the total number of augmented images ranges from 320 to 384.

Unlike data augmentation, data transformation is applied to training, validation, and test data. As part of this process, every image in all three datasets is converted into tensors. This conversion ensures compatibility with deep learning models and transforms images from a standard pixel representation (ranging from 0 to 255) to a normalized numerical format. Since the images are in the Red, Green, Blue (RGB) format, they are represented as three-dimensional tensors with dimensions (3, height, width), where the three channels correspond to the red, green, and blue color channels. An additional preprocessing step is applied to the validation and test data, whereby the images are resized to 256 pixels on their shortest side while maintaining the aspect ratio, and then a centre crop of 224×224 pixels is extracted to ensure a consistent input size. Finally, the pixel values are normalized using ImageNet statistics to standardize the input distribution, improving the model's stability during training.

3.3 Relevant Metrics

For the evaluation of individual models, metrics for classification problems are used. In the first step, predictions are made for the images in the test dataset using the optimized models. In the next step, the results can be viewed in a confusion matrix. Figure 3 shows an example of a confusion matrix in the simplified case of two classes. In the matrix, each row represents

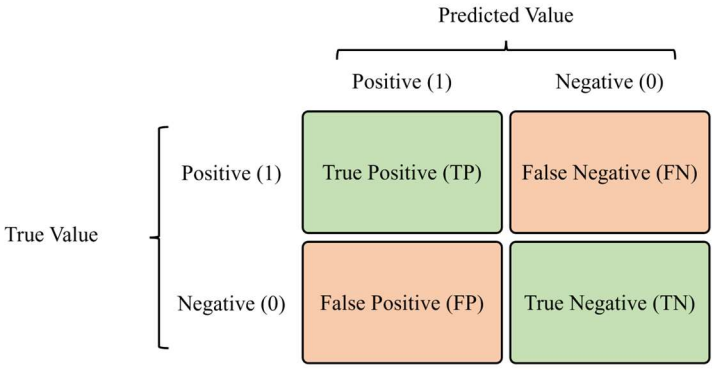


Figure 3: Confusion Matrix for Two Classes (Own illustration based on https://h2o.ai/wiki/confusion-matrix/).

the classes of the actual values of the target variable, and the columns represent the predicted values of the target variable (Géron, 2019), see Figure 3. This allows the predictions to be divided into four categories: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). By analyzing this matrix, several metrics can be determined to assess the model's quality. Additionally, it's possible to identify specific systematic errors. For example, it might become evident that the model has more difficulty predicting certain classes, and these issues can be further analyzed afterwards.

The sum of the True Positive and True Negative categories represents the absolute frequency of correctly classified observations. The number of incorrectly classified observations is determined by the sum of False Positive and False Negative. Therefore, a perfect classification would only include results from the True Positive and True Negative categories (Géron, 2019). Consequently, the goal of optimization is to maximize the number of correctly classified classes. To measure this, the "Accuracy" metric is commonly used in practice. It is calculated according to Equation 3.1 as the proportion of correctly classified observations to the sum of all observations in the test dataset (Runkler, 2015).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

By using Accuracy, we can assess how well the individual models perform in predicting the classes of the various dishes. Another metric that can be derived from the confusion matrix

is Precision, along with the F1-Score. As shown in Formula 3.2, Precision represents the proportion of correctly classified positive samples relative to the total number of predicted positive samples (Tharwat, A. 2021)

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.2}$$

The F1-Score calculates the harmonic mean of Precision and Recall, with Recall, as shown in Equation 3.3, measuring how many of the actual positive cases were correctly predicted (Tharwat, A. 2021).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.3}$$

The F1-Score is calculated according to Equation 3.4, and it is particularly useful when there is an imbalanced class distribution in the dataset, just like Precision (Tharwat, A. 2021).

$$\text{F1} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.4}$$

Since the datasets in this study are balanced, metrics such as Precision and F1-Score offer limited additional insight and are therefore included primarily for completeness. More relevant metrics in this context, in addition to Accuracy, are inference time, the number of trainable parameters, and Floating Point Operations (FLOPs). These metrics, unlike Accuracy, which is used to assess the predictive quality of the model, focus on the efficiency of the model.

The number of trainable parameters provides information about how many weights and biases need to be adjusted during the training process. Since a higher number of trainable parameters leads to a more complex model that requires more training time, it may be advantageous to select a model that utilizes a smaller number of parameters while maintaining equivalent model performance (Goodfellow et al., 2018).

FLOPs measure the number of floating-point operations a model requires during inference (Indiana University, 2024). A high number of FLOPs can indicate that the model is computationally intensive, which also corresponds to higher resource consumption (Indiana University, 2024). Given that our mobile implementation will be deployed on resource-

limited devices, incorporating FLOPs into the evaluation is highly relevant. Inference time measures how long it takes for a model to make predictions on new data (Ranganathan, 2025). This is particularly important in problems like the one at hand, where predictions on new data need to be made quickly in real-time.

3.4 Development

Two different approaches were explored to develop a model suitable for the given use case: (1) the design of a custom CNN and (2) the application of Transfer Learning. For the latter, two variations of EfficientNet were examined: one trained on the ImageNet dataset and the other on the Food101 dataset.

3.4.1 Customized CNN

CNNs are a specialized type of artificial neural networks that have been designed for the specific purpose of processing structured grid data, such as images (LeCun et al., 1998). These networks employ convolutional layers to automatically detect spatial patterns and hierarchical features, making them highly effective for image classification tasks (LeCun et al., 1998).

The core idea behind designing a custom CNN in this study was to develop a lightweight architecture that meets the given requirements while ensuring efficient execution on the target devices. The model's structure was primarily inspired by established CNN architectures from relevant literature and practical implementations, with additional optimizations to improve performance. Figure 4 provides an overview of the model structure."

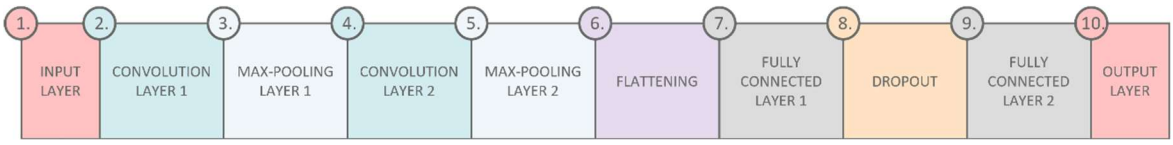


Figure 4: Model Structure of the customized CNN.

The input layer is designed to accept images of 224x224 pixels with three RGB channels. This is followed by two convolutional layers, each succeeded by a max-pooling layer, which reduces complexity and emphasizes key features (Zafar et al., 2022). The resulting feature map is then flattened and passed through a fully connected layer that employs dropout regularization to prevent overfitting (Srivastava et al., 2014). The final layer consists of another fully connected layer that outputs ten prediction values corresponding to the different dishes. The activation function used throughout the network is the rectified linear unit (ReLU) activation function, which is widely regarded as the standard for image recognition tasks (Brownlee, 2020).

3.4.1.1 Transfer Learning

Transfer learning is a technique in machine learning that involves transferring knowledge from a pre-trained model, which has been trained on a large and general dataset, to a related but different task. This approach helps reduce training time and often leads to better performance, particularly when there is limited training data available (Weiss et al., 2016). In the context of this project, transfer learning was applied to the task of food image classification.

For this study, EfficientNet was chosen due to its efficiency in scaling and its high accuracy in image classification tasks (Tan & Le, 2019). EfficientNet utilizes a uniform scaling method, optimizing the depth, width, and resolution of the network in a balanced way, resulting in a model that provides exceptional accuracy while minimizing computational costs (Tan & Le, 2019). This made it an ideal candidate for transfer learning in our case. The setup is shown in Figure 5.

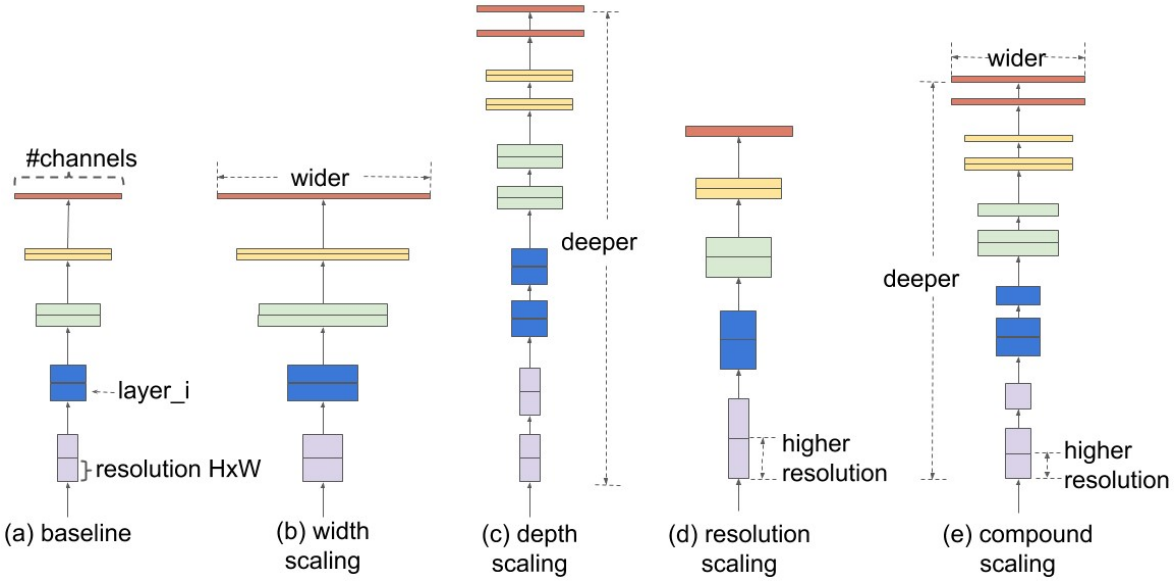


Figure 5: Transfer Learning (Tan & Le, 2019).

Two variations of EfficientNet were explored in this project, each trained on different datasets:

1. EfficientNet pre-trained on ImageNet: The first model used was EfficientNet-b1, pre-trained on the ImageNet dataset. ImageNet is a widely-used dataset containing over 14 million labeled images across thousands of categories (ImageNet, 2025). By using this pre-trained model, we leveraged the extensive knowledge it had gained from a general image classification task. The model's early layers, which capture low- and

mid-level features like edges and textures, were used as a feature extractor for our food classification task.

2. EfficientNet pre-trained on Food101: The second model tested was also EfficientNet-b1, but this time pre-trained on the Food101 dataset (Bossard et al., 2014). The Food101 dataset contains 101 food categories, providing a much more domain-specific feature set for food classification. By using a model already trained on food-related images, it was expected that the features learned from these specific food categories would translate better to our task of cafeteria dish classification.

The advantage of using transfer learning with these two EfficientNet variants was to explore how models pre-trained on different domains (general images vs. food-specific images) would perform for the task at hand. The ImageNet-pretrained model is well-suited for general object classification tasks, while the Food101-pretrained model is expected to bring more relevant features for food classification, potentially leading to better results for identifying dishes in a cafeteria setting.

By comparing the performance of these two models, the study aimed to understand whether the pre-trained models could achieve better results while requiring less training effort in food image classification.

3.4.1.2 Training

Training a deep learning model involves adjusting its internal parameters to minimize the error in its predictions. This is achieved through an iterative process, where the model makes predictions, calculates the error, and updates its parameters using optimization algorithms. In this study, both EfficientNet and the custom CNN model were trained using the backpropagation algorithm. Backpropagation calculates the gradient of the loss function with respect to each parameter and applies the gradients to adjust the parameters to minimize the overall loss (Rumelhart et al., 1986).

The training function used for both architectures is the same and can be found in the appendix. Both EfficientNets were adapted to the given use case by modifying the output layer of the classifier to output ten predictions, corresponding to the ten dish classes. The layers preceding the output were frozen to retain the feature extraction capabilities learned from ImageNet and Food101, allowing the model to be trained on the smaller, domain-specific canteen food dataset (A. Bosco, 2024).

The optimizer used is Adaptive Moment Estimation (Adam) Optimizer, a widely used optimization algorithm that adapts the learning rate for each parameter individually,

improving training efficiency. The loss function selected was Cross-Entropy Loss, commonly used for classification tasks.

A suitable batch size of 16 was manually determined, taking into account the small size of the training dataset. The learning rate was chosen using a separate learning rate finder, which plots the model's losses for various learning rates. The optimal learning rate is identified at the point where the loss shows the greatest rate of decrease (A. Zafar, 2021). Figure 6 shows the recorded curves for both architectures. The code for the learning rate finder is included in the appendix. For our custom CNN, a learning rate of 0.0003 was found to be optimal, while for EfficientNet, 0.002 was selected.

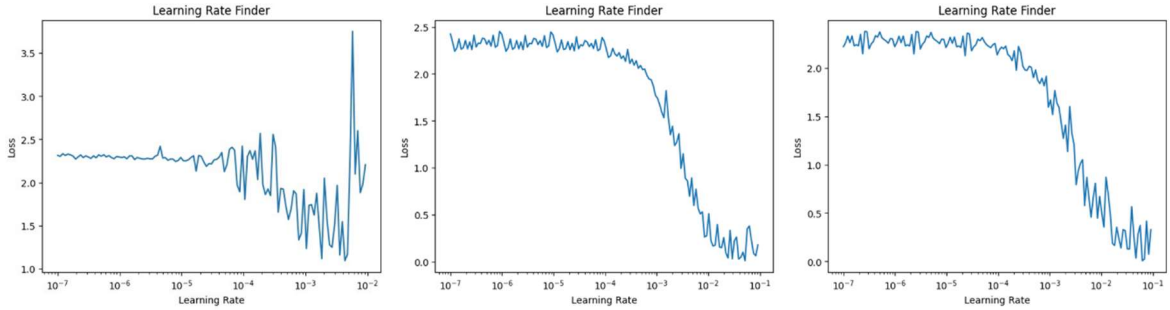


Figure 6: Learning Rate Finder.

To prevent overfitting and unnecessary training, a stop loss mechanism was implemented. This halts training after five epochs without improvement in the validation loss. After two epochs without improvement, an on-plateu scheduler quintuples the entered learning rate to overcome the plateu.

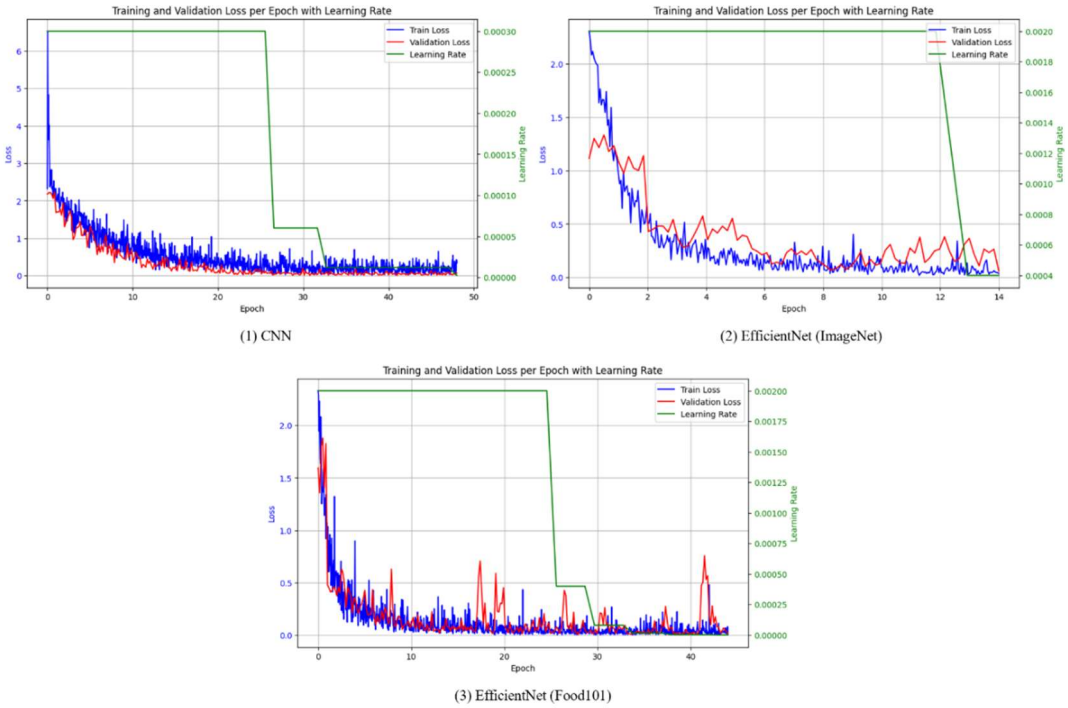


Figure 7: Training and Validation Loss per Epoch for each Model with Learning Rate.

During training, the model's training and validation losses were recorded, along with the learning rate, and visualized in graphs to evaluate the training process. These graphs can be seen in Figure 7. Whenever the validation loss reached a minimum, the model was saved for further evaluation.

3.5 Explainable Artificial Intelligence (XAI)

3.5.1 The importance of transparency in AI systems

The main obstacle to the use of AI-based systems is usually a lack of transparency (Adadi & Berrada, 2018). Machine learning algorithms in particular are often difficult to understand. This makes it challenging to understand their internal working mechanisms, which makes them appear less trustworthy. An increasing amount of attention is being paid to this problem as AI has a growing impact on critical decision-making processes, such as medical diagnoses. In these areas, understanding how AI systems arrive at a particular decision is fundamental to gaining the trust of users. The black box nature of systems means that the internal decision-making processes remain hidden, even though the results are precise. This leads to a debate about Explainable AI (XAI), as there is no direct way to explain the system's decisions. XAI is a field of research that promises to lead to significant improvements in trust in AI-based systems and transparency of internal working mechanisms. According to Barocas et al. (2018), the goal of explainability in machine learning is to ensure that algorithmic decisions and all underlying data can be explained to end users and other stakeholders in understandable, non-technical language. When implementing this goal, it is important to maintain the high level of performance of the systems and not to influence them.

3.5.2 Integration of XAI

The integration of Explainable AI into the project serves to make the model's decision-making processes transparent and comprehensible. This is particularly important in the context of image classification and object recognition, as it enables the results of the model to be better understood and evaluated. Integration can take place in the following project phases: Development phase, Optimization phase and from an end-user perspective.

In the development phase, XAI is used to make the decision-making processes of AI models more transparent, which is crucial for developers. In particular, it enables the analysis of how classification models work by visualizing the image areas that the model takes into account when classifying certain categories. This helps to understand whether the model's decisions make sense and are based on relevant image regions. An example of the practical use of XAI in this phase is the work "Human-AI Interaction in Industrial Robotics: Design and Empirical Evaluation of a User Interface for Explainable AI-Based Robot Program

Optimization" (Alt et al., 2024), which investigates the development and evaluation of a user interface for explainable AI in robot program optimization.

In the optimization phase, XAI enables the fine-tuning of models by analyzing misclassifications and providing understandable explanations that make it easier for developers to improve the performance of AI systems. For example, it is possible to investigate whether the model is using irrelevant image features for its decisions. Based on these observations, the model is adapted, for example by expanding or cleaning up the data set or by modifying the model architecture. This not only leads to an improvement in the accuracy of the model but also increases its robustness. The study "Transcending XAI Algorithm Boundaries through End-User-Inspired Design" (Jin et al., 2022) emphasizes the importance of XAI in overcoming the limitations of existing algorithms and developing user-centered explanations.

From the end-user perspective, XAI strengthens users' trust and understanding of AI systems by making decisions transparent and comprehensible. The paper "What Do End-Users Really Want? Investigation of Human-Centered XAI for Mobile Health Apps" (Weitz et al., 2022) shows how important it is to adapt explanation styles and content to the needs of users. For example, less technology-oriented users prefer intuitive and simple visualizations, while detailed explanations are beneficial for experts. Such user-centered approaches not only promote acceptance of the technology but also enable more efficient and trusting interaction with AI-supported applications.

In the following, XAI was used to look at the created AI system from an end-user perspective to understand why the model makes certain decisions for or against a class. In addition, the knowledge gained in this report can be used as a basis for future work.

3.5.3 Local Interpretable Model-Agnostic Explanations (LIME)

One promising XAI method is Local Interpretable Model-Agnostic Explanations (LIME). The LIME approach focuses on explaining individual predictions of a particular black-box model. The basic idea is to simulate the behavior of the black-box model in a small neighborhood of the instance of interest (Molnar, 2022). For this purpose, the contribution of individual features to a particular prediction is evaluated by LIME generating perturbations of the data. These perturbations are used to train a simple local model (surrogate model) that attempts to approximate the predictions of the complex model as closely as possible in the vicinity of the instance of interest (Yu-Hsin Hung & Chia-Yen Lee, 2024). LIME provides a unified method for explaining predictions regardless of the type of model used. To analyze image data, LIME uses the superpixel technique to specifically take the pixel structure into account. This involves dividing the image into larger, contiguous

units consisting of nearby pixels with similar colors. As individual pixel changes often have no significant influence on the model prediction, this division allows the image features to be captured more precisely. Copies of the original image are created, with individual segments randomly activated or deactivated to create different variants of the image. These variants are used to train a simple model that identifies which areas of the image are critical to the prediction of the model. After the model is applied to each variant, LIME fits a linear model that approximates the behavior of the complex model near the original image. The activation states of the segments act as input data. The resulting weights of the linear model illustrate how strongly each segment contributes to the prediction. Finally, particularly relevant segments are visually highlighted to clearly show which image regions influence the model decision. An illustrative example of the application of LIME to image data is the classification of images by a neural network. If the model classifies an image as a "dog", LIME can identify the superpixels that contributed to increasing the likelihood of it being classified as a dog, such as areas of fur or snout (Ribeiro et al., 2016).

To begin with, it is necessary to transform the image to be examined in the same way as the training, test and validation images, which can be seen in Listing 1 of the Appendix. It can then be passed to the Lime-Explainer, which passes the image into the black box model five thousand times according to the num_samples variable and changes it slightly each time to identify the decisive image features. Superpixel visualization and a heatmap with the weights of the superpixels are particularly suitable for outputting these features in image data. When visualizing the superpixels, the five most important image features for deciding on the class with the highest probability are displayed. Only the superpixels that had a positive influence on the decision for the class are shown, but due to the high model accuracy, there were hardly any images with negative superpixels. Accordingly, most heat maps only show that all areas in the image were either neutral or positive for the model's decision.

3.5.4 Gradient-weighted Class Activation Mapping (Grad-CAM)

Another XAI method for determining decision-relevant features is Gradient-weighted Class Activation Mapping (Grad-CAM). It is a method for visually interpreting CNNs without influencing their structure or reloading the model. Similar to LIME, Grad-CAM uses heat maps to highlight certain areas of an image. However, the structure of Grad-CAM differs considerably, as the basic idea is to use the last layer of the CNN to determine the importance of each neuron for the individual classes.

The gradients are calculated before the softmax function for the respective classes in relation to the property maps of a layer. This gradient is then used to determine the importance of the neurons. In the following, they are combined with the feature maps using a partial

linearization of the network. The ReLU function is then used to ensure that only the positive features of a class are extracted. This ensures that the final heatmap does not highlight more than the desired class and therefore performs better. The result is a representation in which the most important elements are marked with warmer colors.

4 Results

This section compares the performance of three models (Custom CNN and EfficientNet variants. Following this, the results of XAI methods, specifically LIME and Grad-CAM, are analyzed, with a focus on their functionality and limitations.

4.1 Comparing Model Results

To determine which model would best meet the requirements of the use case, a comprehensive evaluation was performed using several performance metrics, which were previously discussed in Section 3.3. The evaluation was carried out using the "evaluate_model" function, which computes key metrics such as accuracy, precision, recall, F1-score, and inference time, as well as generating confusion matrices to assess classification performance. The evaluation was conducted using the scikit-learn and torchprofile libraries (scikit-learn, 2025; torchprofile, 2025). The full code for this process can be found in the project GitHub.

	Custom CNN	EfficientNet (ImageNet)	EfficientNet (Food101)
Accuracy	0.92	0.96	0.99
Precision	0.92	0.95	0.99
Recall	0.93	0.96	0.99
F1-Score	0.92	0.95	0.99
Inference time	0.087s	0.231s	0.238s
Parameters	25.71 million	6.53 million	6.53 million
FLOPs	9.61 GFLOPs	0.41 GFLOPS	0.41 GFLOPS

Table 1: Results of evaluate_model-function.

Table 1 summarizes the evaluation results for the three models. All models achieve excellent accuracy, with the EfficientNet models outperforming the custom CNN in several key areas. The custom CNN, with an accuracy of 0.92, performs well but falls short in comparison to the EfficientNet models, which achieve accuracy rates of 0.96 (ImageNet-trained) and 0.99 (Food101-trained).

In addition to accuracy, the EfficientNet models also demonstrate better efficiency. Despite their slightly higher inference times (0.231s for ImageNet and 0.238s for Food101), they feature significantly fewer parameters (6.53 million) and FLOPs (0.41 GFLOPs) compared to the custom CNN (which has 25.71 million parameters and 9.61 GFLOPs). This translates to reduced memory usage and energy consumption, making the EfficientNet models more

suitable for resource-constrained environments, such as mobile devices or embedded systems, as is the case with our alternative mobile app implementation. The slightly higher inference time should hardly be noticeable in real-world scenarios and is negligible.

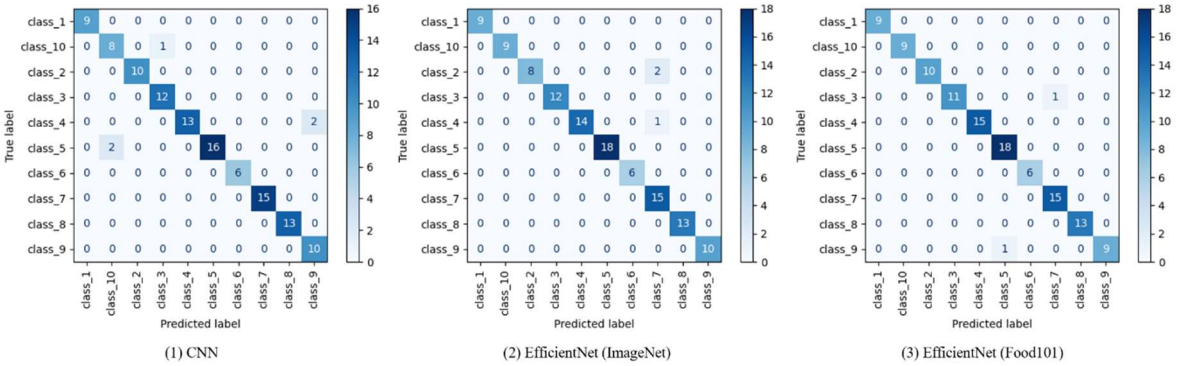


Figure 8: Confusion matrix of each model.

The confusion matrices shown in Figure 8 further corroborate the models' high performance, where no prominent patterns of systematic errors are observed. The predictions are well-distributed across the classes, suggesting that the models are equally capable of generalizing across different dish categories. These results indicate that the models are robust, with no specific class exhibiting a significant number of misclassifications or consistent error trends. Based on these results, the EfficientNet (Food101) model was selected for further use. It offers the best combination of high accuracy, efficient resource usage, and suitability for deployment on mobile devices, with a negligible increase in inference time.

4.2 XAI Results

In the following chapter, both the basic functionality and the limitations of the two XAI methods are tested on the CNN created by the authors, and its strengths and weaknesses are evaluated.

4.2.1 Basic Functionalities

In the first phase of the study, the results from LIME were characterized by the weaknesses already explained, such as inaccuracy and inconsistency. Explanations generated by the XAI method showed that sometimes irrelevant features such as the tray or the edge of the plate were important. This "tablet problem" severely limited the functionality of the LIME method. Furthermore, fundamentally different explanations were issued for the same input of a class. In this context, it was found that the best results were obtained with five successfully registered image features. These were achieved by class 1, pasta with red sauce as seen in figure 9.

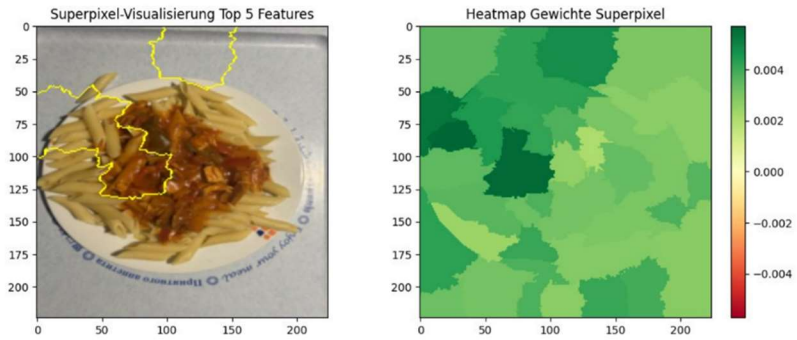


Figure 9: LIME Heatmap for class 1

On the other hand, the results from the Grad-CAM method were very promising at first glance. The dishes were appropriately marked as heat maps from the outset and delivered consistent results within the same classes. For example, the pasta plate, shown in Figure 10, was correctly recognized and marked with a high probability of 97%. However, other similar classes such as class 4, spaetzle with chicken, or fundamentally different classes such as class 6, pollock fillet in eggshell, could also be identified with probabilities of over 90%, as shown in Figure 10.

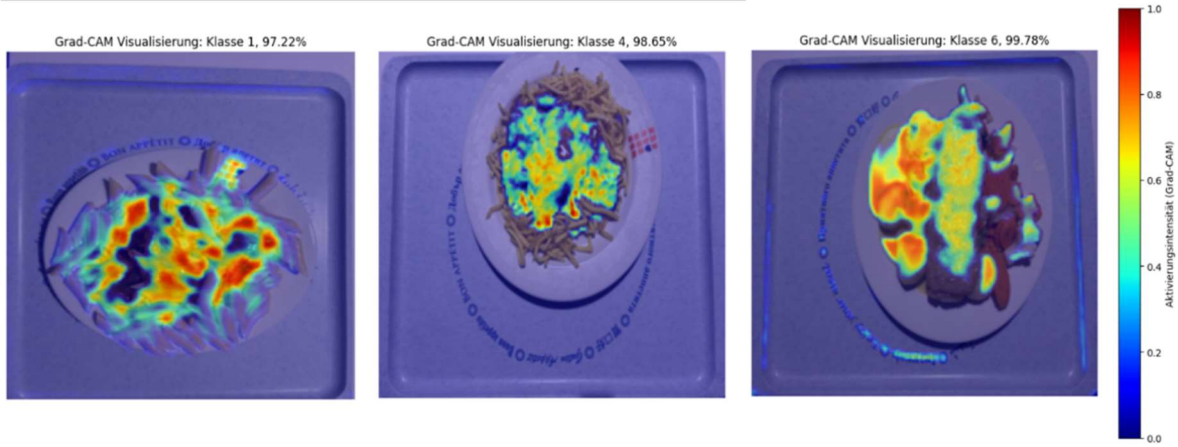


Figure 10: Grad-CAM Heatmap for class 4 and 6.

4.2.2 Classification of multiple Inputs

The second phase focused on the limitations of the XAI methods. For this, two dishes were used as input to analyze the decisions of LIME and Grad-CAM and to determine possible limitations of the network. The optimal output of such an input would be the labeling of one sample alone, neglecting the other. Within this experiment, three different characteristics were used.

Combination I of pasta plate (class 1) and beef black beer pan (class 2) resulted in a 49% probability of the flatbread with vegan seitan kebab (class 7), closely followed by the pasta plate (class 1) with 33%, but a probability of class 2 of only 3%. When looking at the

explanation, it was noticeable that the yellow noodles, red sauce and green beans were the five most important image features. When looking at class 7, a color correlation is noticeable: shiny yellow bread, red tomatoes, green lettuce. The conclusion of the combination of classes 1 and 2 is that class 7 represents the best mix of features.

Combination II aimed to integrate two dishes with chips in order to determine possible bias. For this purpose, the pollock fillet with potato breading (class 3) and the turkey gyros (class 9) were combined. The result here was 67% class 3 followed by the black beer pan of beef (class 2) with 25 %. A closer look at class 2 reveals that the beans and the meat look similar to class 9, but the question remains as to why the model classifies only 2% probability for class 9. With this combination, it is noticeable that the old “tablet problem” from the first investigation phase occurs again. This phenomenon was therefore investigated in detail later on.

The idea origin of combination III is similar to the previous experiment set-up, as two dishes with pollack are used. The Alaska pollack fillet in egg coating with hollandaise sauce (class 6) and Alaska pollack fillet in egg coating (class 10). Here, LIME opted for pollack fillet in potato breading (class 3) with 48 % probability, followed by class 6 with 23 % probability. Here, too, it is noticeable that class 3 combines many different characteristics, such as a similar fish, fries that resemble the shiny fried potatoes and another relevant ingredient, mayonnaise (Figure 11). Furthermore, a corner of the tray had a negative influence on the decision for class 3, which means that the model probably makes an assignment to class 3 dependent on the tray, among other things, and calls our data quality into question. The task in future would therefore be to re-examine and interpret these findings and derive actions.

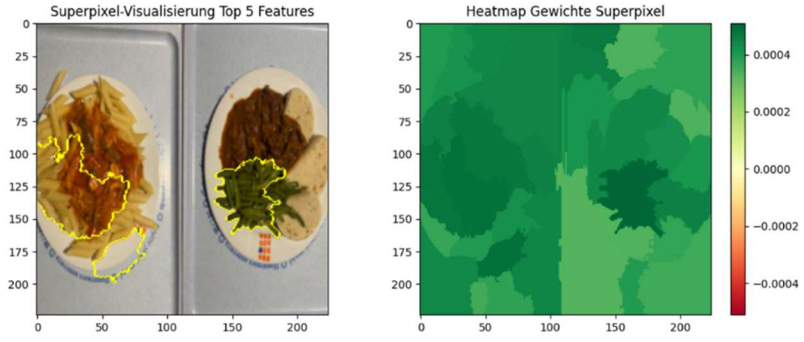


Figure 11: Resulting Heatmaps for combined dishes with LIME.

However, the assignment by Grad-CAM did not lead to a reliable result. Regardless of the combination of dishes used, the model identified class 3 in every trial, as can be seen in Figure 12, with a high prediction value of 98% in some cases. This means that there are

irrelevant but nevertheless learned features within this class that the model uses for decision making.

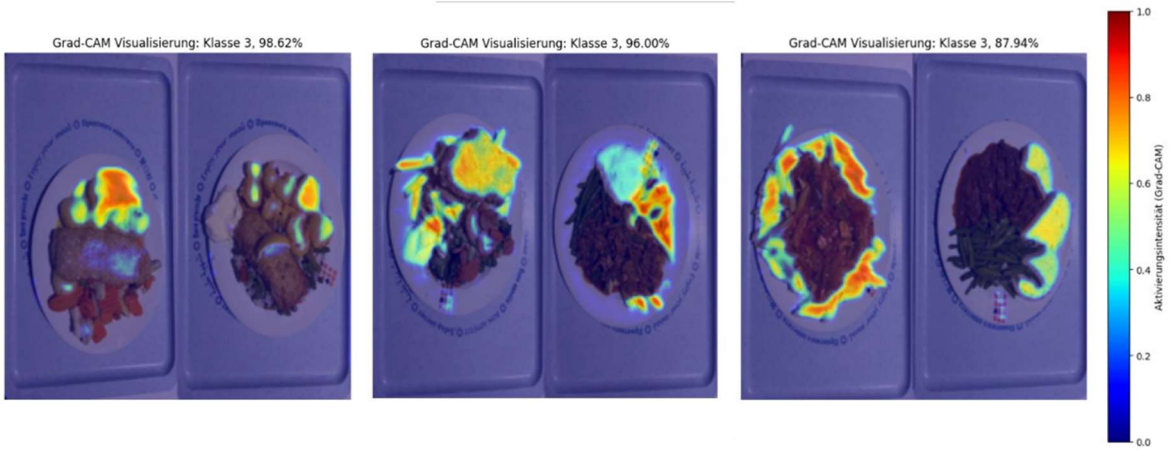


Figure 12: Resulting Heatmaps for all combined classes with Grad-CAM.

4.2.3 Analyzing the "Tablet Problem"

As mentioned in combination II, a more detailed analysis of the tablet problem is necessary. Therefore, a picture was cut together showing an empty tray. The results for LIME, which can be seen in Figure 13, were spring roll with vegetable filling (class 8) with 39%, followed by pollock fillet in potato breading (class 3) with 29% and black beer pan of beef (class 2) with 24%.

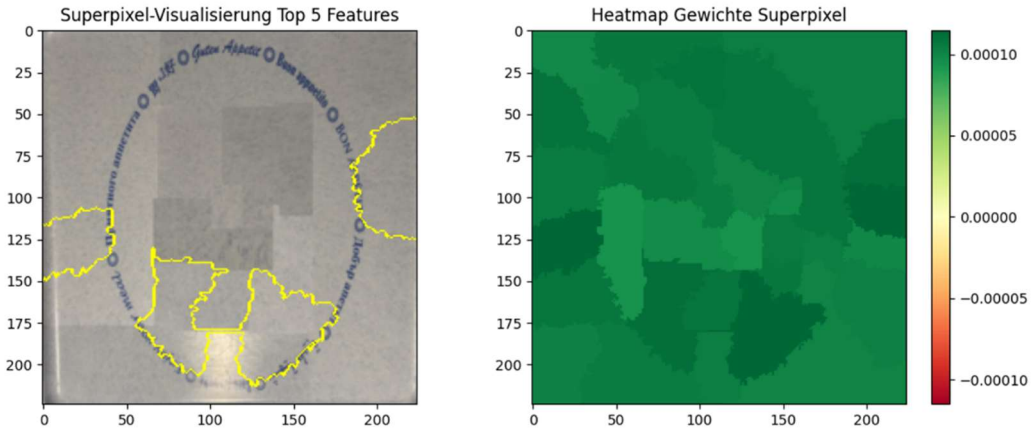


Figure 13: Resulting Heatmap only tablet visible with LIME.

Grad-CAM identified the empty plate as class 3 with 77%. (Figure 14) The theory behind these results is that the plates with the dishes were moved around on the plate to varying degrees or often during photography. As a result, many different parts of the tray were also visible when the original 50 to 60 photos were taken. Both methods could therefore have drawn unintended conclusions from this and learned characteristics of the tray.

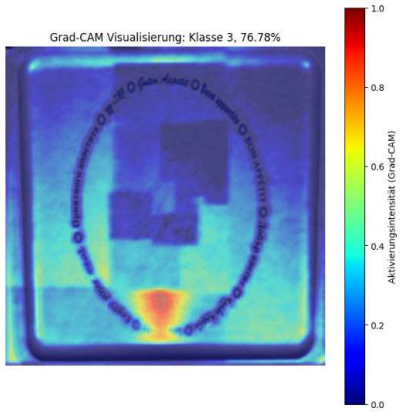


Figure 14: Resulting Heatmap only tablet visible with Grad-CAM.

4.2.4 LIME vs. Grad-CAM

In a direct comparison of the two models for visualizing individual dish predictions, Grad-CAM proves more effective for basic identification. Its heatmaps highlight the key areas of the input more distinctly, providing a clearer view of the decisive pixels. However, for more complex scenarios, such as multiple dishes in a single image, LIME delivers a more accurate approach to which classes are relevant. The labels and predictions indicate that LIME outperforms Grad-CAM in this context. This discrepancy may be influenced by dataset bias, among other factors, and could potentially be mitigated by expanding the dataset.

Despite this, refining the Grad-CAM model could enhance its suitability for deployment on end-user devices. Its more intuitive heatmap representation makes it a valuable tool for users, offering a clearer and more interpretable visualization of model decisions. Ultimately, the usage of both models improved the understanding of possible bias within the dataset as well as the different influences which can partly falsify certain outputs.

5 Model-Deployment

An Android app prototype was developed for demonstration purposes, integrating the final model. The app, named “MensAI,” allows users to capture images of dishes using their camera, which are then classified into one of the ten test categories. If the image does not contain any of the trained dishes, an error message is displayed, informing the user that no match was found.

For optimization on mobile devices, the PyTorch model was exported for mobile use using the TorchScript library (TorchScript, 2025). This step converts the model into a format suitable for efficient execution on mobile devices. To load the model and integrate it into the app, the Pytorch-Android library was utilized (Pytorch-Android, 2025). The model is loaded directly from the app’s assets and integrated into the data layer of MensAI.

Although this on-device approach presents the typical edge computing challenges, such as limited computing power, it also offers several advantages. Key benefits include enhanced security and privacy as no data needs to be sent to a server (Y. Mao et al., 2017). It also provides reduced latency, as all computations are performed directly on the device (Y. Mao et al., 2017). Notably, for this project, the fact that no server infrastructure is required is especially significant, greatly simplifying the solution and reducing costs.

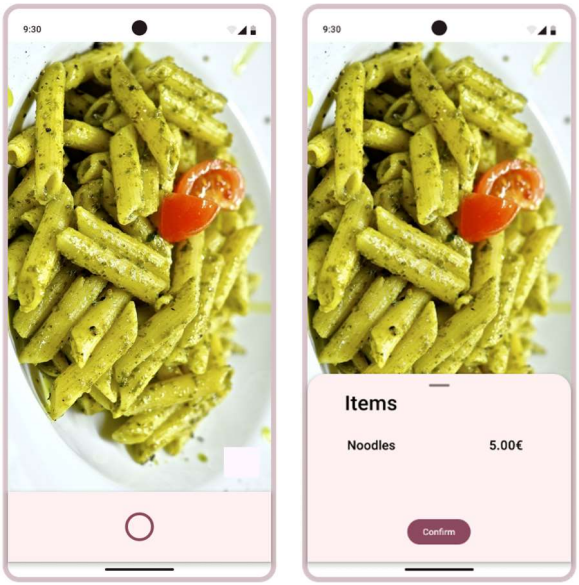


Figure 15: First Mockup.

A mockup of the user interface (UI) was initially created to visualize the design. This can be seen in Figure 15. Based on this, the user interface was developed using Jetpack Compose (Jetpack-Compose, 2025), a modern UI toolkit for Android that facilitates declarative UI development and ensures high performance.

To classify a dish, the user must align the camera accordingly and press the record button. A Bottom Sheet then appears, displaying the recognized dish with its price and offering the option to proceed with payment or, if the wrong food is recognized, to take a new picture. If the model's inference confidence does not exceed a 0.8 threshold, the recognition fails, and the user is asked to take a new picture.

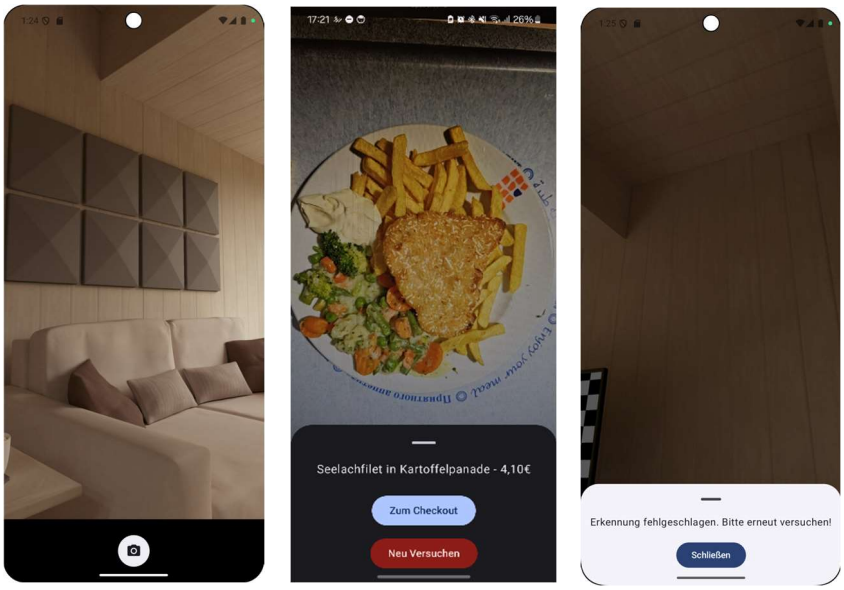


Figure 16: UI of MensAI.

The final user interface of the MensAI app, including all interaction options and display of recognition results, is shown in Figure 16. The complete code and additional details about the implementation can be viewed in the associated GitHub repository.

6 Discussion

This section analyzes the model results, with particular emphasis on the challenges related to data collection. Additionally, the model design is examined, and key considerations regarding the business case are discussed.

6.1 Critique of the Results

The models evaluated in this study achieved impressive performance metrics, particularly with respect to accuracy, precision, and recall. The EfficientNet (Food101) model demonstrated the highest accuracy, outperforming both the custom CNN and the EfficientNet (ImageNet) models. However, while these results are promising, it is essential to consider the real-world implications of such high accuracy values.

One potential issue with very high accuracy is the possibility of overfitting. Given the controlled environment in which the models were trained and tested, it is crucial to ensure that this high accuracy translates to real-world robustness. In particular, the lack of significant errors or misclassifications in the confusion matrices (shown in Figure 16) might suggest that the models have been highly tuned to the training set. Since only 50-60 images were taken for each class, with 20% of them reserved for testing, this results in only 10-12 test images per class for model evaluation. To conduct a more realistic evaluation, a significantly larger number of images would be required to fully assess model performance.

6.2 Challenges in Data Collection

In addition to the limited number of images, one of the major challenges in data collection is ensuring that the dataset reflects the diversity of possible real-world scenarios. Developing an effective meal classification system relies heavily on the quality and variety of the dataset. One of the primary challenges is the high variability in meal presentation. Unlike standardized products in retail, cafeteria meals can appear in numerous forms due to differences in plating, portion sizes, and even preparation methods. For instance, a meal classified as pasta with tomato sauce can look quite different depending on how the sauce is distributed or how the pasta is arranged on the plate. These variations complicate the training process and require the dataset to include diverse representations of each meal.

To address this, variance was intentionally introduced during the data collection and transformation processes. However, with only 50 images per class-further split into training, testing, and validation sets-this dataset may not be sufficient to capture the full range of variations encountered in a real-world setting. This limitation became evident in the XAI, which revealed that the model sometimes learned unintended features rather than focusing

on the actual dish composition. The model could mistakenly associate certain features, such as the shape of the plate or background elements, as important features for classification.

Environmental factors also pose significant challenges. Cafeteria lighting conditions can vary throughout the day, with overhead lights casting shadows or creating reflections that distort image quality. The angle at which a photo is taken can also impact classification performance. If a user captures an image from an extreme angle, the model may struggle to recognize the meal correctly. While standardized camera positioning and controlled lighting in fixed self-checkout kiosks could enhance consistency, such an approach may not be feasible for a mobile-based solution. As such, robust preprocessing techniques and adaptive augmentation strategies would be necessary to account for these variations.

To improve generalization and achieve more realistic, real-world results, it is essential that future testing incorporates images captured under diverse, real-life conditions. The model should be evaluated in settings that mimic the variability and unpredictability of actual cafeteria environments, such as differences in lighting, plate arrangements, and varying levels of crowding. Additionally, it would be valuable to include images taken by users with different devices, as varying camera quality could also impact classification performance.

Future efforts should focus on expanding the dataset to include a larger and more diverse collection of images, ensuring it accurately reflects the dynamic and evolving nature of cafeteria settings. This will help account for the wide range of meal presentations, environmental factors, and potential edge cases that may not have been fully captured in the initial dataset. By doing so, we can improve the model's ability to generalize to real-world scenarios and ultimately enhance its reliability and robustness in practical applications.

6.3 Model Design and Model Training

Given the challenges highlighted in the previous section, it's clear that the current model design must evolve to better accommodate the complexity of real-world cafeteria environments. The current model, framed as a classification problem, focuses on categorizing an image of a tray into one of ten predefined meal categories. However, as noted, in a real cafeteria setting, trays may contain additional non-food items such as cutlery, napkins, beverages, and receipts or even multiple dishes.

The existing architecture does not account for these extraneous elements leading to misclassifications or irrelevant focus on non-food items. To address this, integrating an object detection component into the model is essential. This addition would allow the system to first detect all objects on the tray, identifying the relevant food items before making a classification decision.

Additionally, exploring different training approaches could help improve classification accuracy and adaptability. One approach, as previously employed, involves training a single large model to classify each meal based on learned features. Alternatively, training individual models for each dish could allow for more precise classifications, with each model specialized to detect a particular meal. While this approach offers advantages such as easier integration of new dishes and potentially higher accuracy through binary classification (meal present or not), it also presents challenges. Distinguishing between similar meals may be more difficult, and managing multiple models could increase the complexity of the system. Both approaches present distinct benefits and trade-offs, and a comprehensive evaluation of these options will be critical in developing a model that is both accurate and efficient in a real-world setting.

6.4 Business Case Considerations

Beyond the limitations of the classification model and small dataset, there are additional key factors that significantly impact the feasibility and success of implementing an AI-powered self-checkout system in university cafeterias. A primary consideration is the deployment method. There are two potential approaches: The use of a mobile application or a fixed self-checkout kiosk.

A mobile app offers flexibility, allowing users to scan their meals using smartphones. However, this approach is highly dependent on the quality of the smartphone camera and the environmental conditions, which can negatively affect classification accuracy. For example, poor lighting or extreme angles could distort the image and impact recognition. In contrast, a fixed kiosk with a high-quality camera would ensure more consistent image capture and better control over environmental variables but requires a higher initial investment in hardware and maintenance. These two methods could complement each other, with the mobile app providing maximum flexibility and kiosks offering a more reliable alternative in controlled environments.

Another critical consideration is trust and fraud prevention. The system assumes users will correctly scan their meals, but there is the potential for both accidental and intentional misuse. For example, users might attempt to manipulate the system by hiding food items, such as placing a piece of Schnitzel under a pile of pasta to be charged for a lower-priced dish. This highlights the importance of addressing uncertain classifications. One potential solution could be QR-code verification, where users are given a generated code after successful classification and payment, which is then scanned at the exit to ensure accuracy. This method mirrors the approach used in some automated retail self-checkout systems, such

as those implemented by REWE supermarkets, where customers are required to scan their receipts upon leaving the store.

Pricing differentiation is another important aspect to consider. University cafeterias typically have different pricing structures for students, faculty, and external visitors. The AI-powered checkout system would need to identify the user category to apply the correct pricing, which could involve integration with university ID systems or digital payment platforms. Without this mechanism, the system could apply the wrong pricing, leading to financial discrepancies. One possible solution is the implementation of user accounts, requiring some form of authentication as proof of their affiliation.

Finally, the success of the implementation depends on collaboration with the Studentenwerk Leipzig. Engaging with stakeholders, cafeteria managers, and IT departments is crucial for assessing the feasibility and ensuring smooth integration of the AI-powered meal recognition system into existing cafeteria infrastructure.

7 Conclusion

This study set out to explore the potential of AI, specifically ML and CNNs, to enhance efficiency and customer experience in university cafeterias managed by Studentenwerk Leipzig. For this purpose advanced image classification techniques were employed and their potential for AI based meal recognition evaluated.

Throughout the study, we designed and implemented a classification model trained on meal images, ensuring robustness through data augmentation and optimization strategies. A comparative analysis of different CNN architectures, including a custom-designed CNN and EfficientNet models, demonstrated that transfer learning on domain-specific datasets yielded superior classification accuracy. The best-performing model, EfficientNet fine-tuned on the Food101 dataset, achieved an accuracy of 99%, combining high precision with computational efficiency making it suitable for mobile deployment. Furthermore, the integration of XAI techniques, such as LIME and Grad-CAM, provided valuable insights into model interpretability, highlighting areas for potential refinement and bias mitigation.

These results are promising and indicate that AI-powered meal recognition is a feasible approach for cafeteria environments. However, as discussed in Section 6, these results must be critically evaluated in light of data limitations and inherent challenges in the model design. The high accuracy observed may not fully reflect the model's robustness in real-world scenarios due to factors such as limited data diversity, environmental variations, and the risk of overfitting.

To address these challenges, future work should focus on expanding the dataset to include a broader range of meal variations, integrating object detection for better multi-item classification, and refining deployment strategies to account for environmental variability and differences in user devices. These improvements will help enhance the model’s performance and ensure more reliable results in diverse real-world conditions.

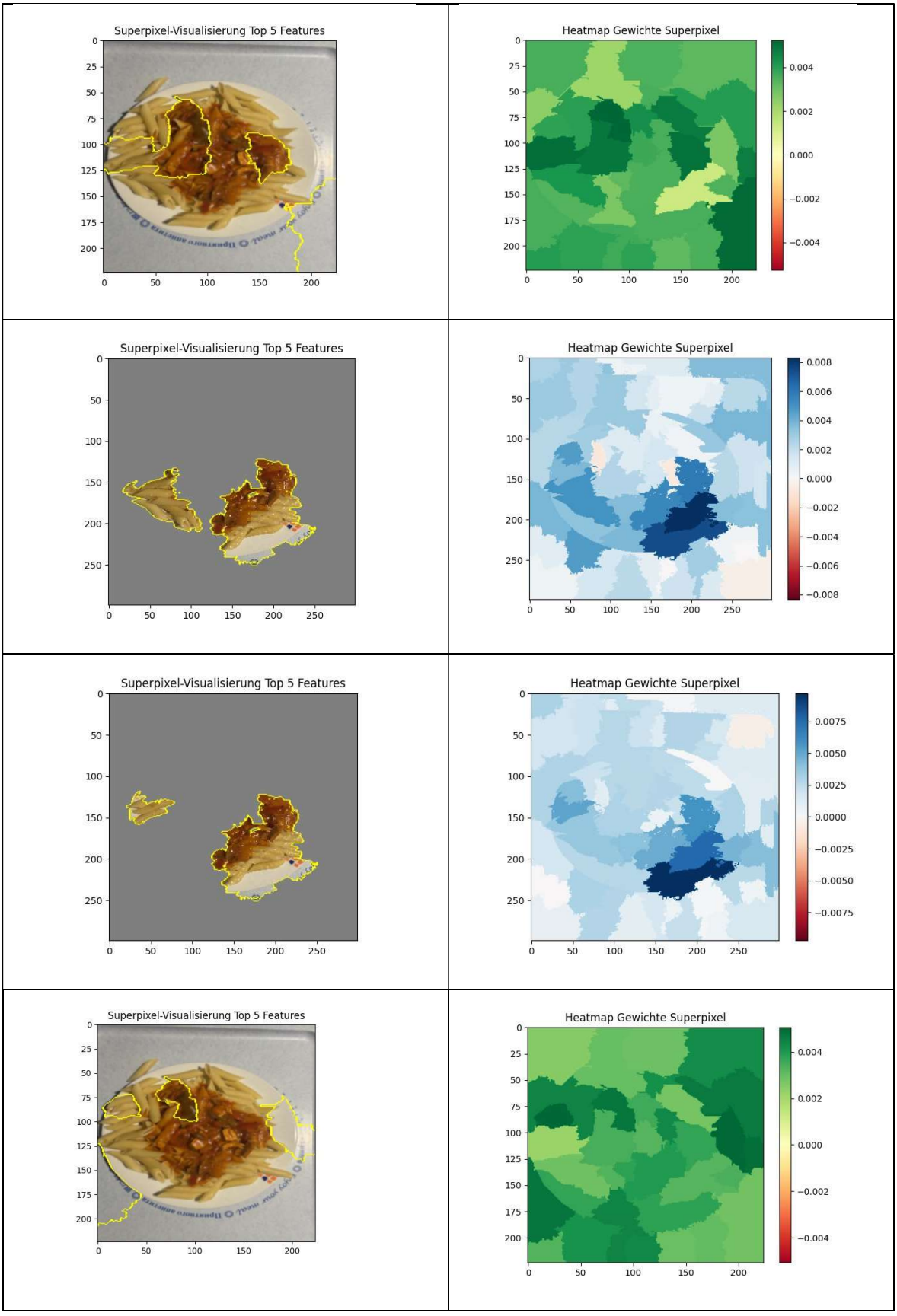
8 Appendix

```
explainer = lime_image.LimeImageExplainer()
explanation =
explainer.explain_instance(np.array(pill_transf(img)),
                                        batch_predict,
                                        top_labels=5,
                                        hide_color=0,
                                        num_samples=5000)
#Anzahl an Bildern die an die Klassifikationsfunktion übergeben
werden

# Superpixel-Visualisierung (positive_only=True)
temp, mask =
explanation.get_image_and_mask(explanation.top_labels[0],
positive_only=True, num_features=5, hide_rest=False)
img_boundry1 = mark_boundaries(temp/255.0, mask)
plt.title('Superpixel-Visualisierung Top 5 Features')
plt.imshow(img_boundry1)
plt.show()

# Heatmap der Gewichte der Superpixel
ind = explanation.top_labels[0]
dict_heatmap = dict(explanation.local_exp[ind])
heatmap = np.vectorize(dict_heatmap.get)(explanation.segments)
plt.imshow(heatmap, cmap='RdYlGn', vmin=-heatmap.max(),
vmax=heatmap.max())
plt.colorbar()
plt.title('Heatmap Gewichte Superpixel')
plt.show()
```

Listing 1: Lime Explainer and Plots.

```
# Grad-CAM ausführen
grayscale_cam = cam(input_tensor=input_tensor, targets=None)
# Vorhersage berechnen
with torch.no_grad():
    output = gradcam_model(input_tensor)  # Modellvorhersage
(Logits)
    probabilities = F.softmax(output, dim=1)  #
Wahrscheinlichkeiten berechnen
    predicted_class = torch.argmax(probabilities, dim=1).item()
# Klasse mit höchster Wahrscheinlichkeit
    predicted_probability = probabilities[0,
predicted_class].item() * 100  # Prozentwert
# Klasse und Wahrscheinlichkeit anzeigen
print(f"Vorhergesagte Klasse: {predicted_class}")
print(f"Wahrscheinlichkeit: {predicted_probability:.2f}%")
# Visualisierung vorbereiten
example_image_np = np.array(example_image.resize((224, 224))) /
255.0
visualization = show_cam_on_image(example_image_np,
grayscale_cam[0], use_rgb=True)
# 1. Aktivierungswerte für die Farbskala vorbereiten
activation_map = grayscale_cam[0]  # Grad-CAM Heatmap
(Grayscale)
norm = Normalize(vmin=0, vmax=1)  # Werte normalisieren
(zwischen 0 und 1)
colormap = cm.jet  # Farbskala (Jet ist gängig für Heatmaps)
# 2. Plot mit Farbskala
fig, ax = plt.subplots(figsize=(8, 8))
# Zeige das Bild mit der Heatmap
im = ax.imshow(visualization)
ax.axis("off")
ax.set_title("Grad-CAM Visualisierung mit Farbskala")
# 3. Farbskala hinzufügen
cbar = fig.colorbar(cm.ScalarMappable(norm=norm, cmap=colormap),
ax=ax)
cbar.set_label("Aktivierungsintensität (Grad-CAM)")
plt.show()
```

Listing 2: Grad-CAM Setup and Plots.


Superpixel-Visualisierung Top 5 Features
Heatmap Gewichte Superpixel
Superpixel-Visualisierung Top 5 Features
Heatmap Gewichte Superpixel
Superpixel-Visualisierung Top 5 Features
Heatmap Gewichte Superpixel
Superpixel-Visualisierung Top 5 Features
Heatmap Gewichte Superpixel

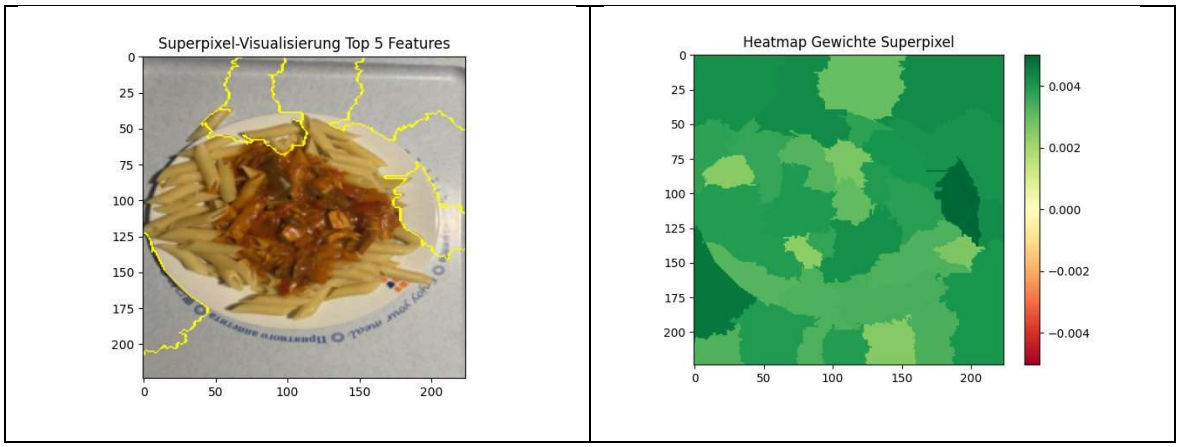
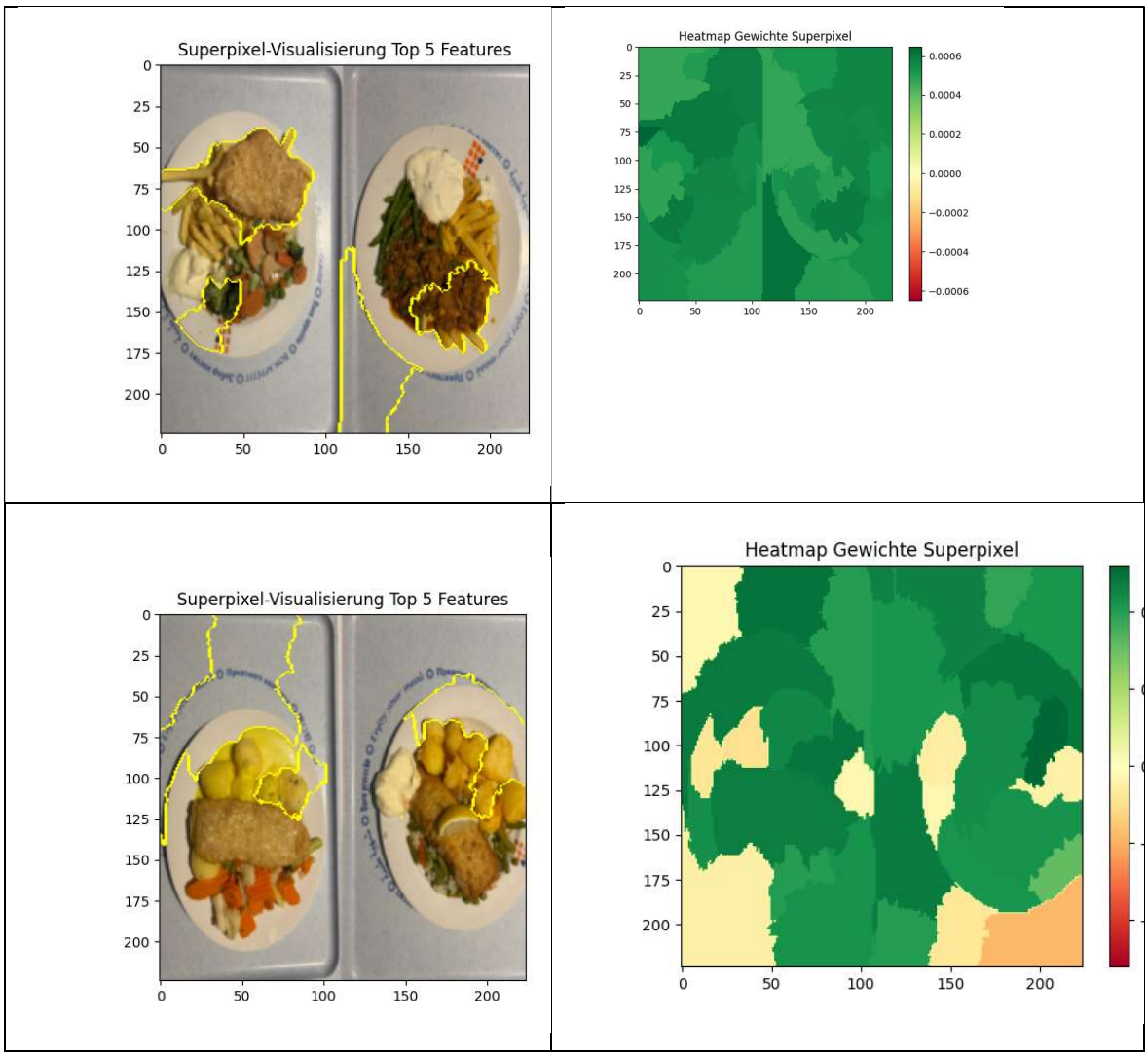


Table 2: Results with LIME



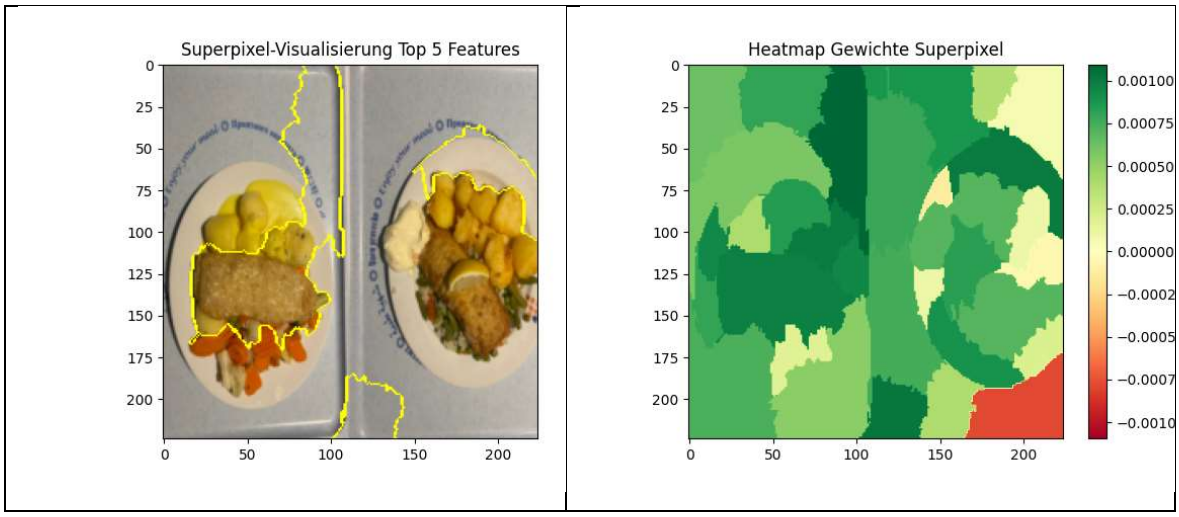


Table 3: Results for Combination of Inputs with LIME

9 Literature

Adadi, A., & Berrada, M., (2018), Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI), *IEEE Access*, *6*, 52138–52160, https://doi.org/10.1109/ACCESS.2018.2870052

Ali, N. S. M. S. (2024). Adoption of Self-Checkout Systems in Retail and Their Impact on Customer Experience. Journal of Informatics Education and Research, 4(3). https://doi.org/10.52783/jier.v4i3.1970

Alt, B., Zahn, J., Kienle, C., Dvorak, J., May, M., Katic, D., Jäkel, R., Kopp, T., Beetz, M. & Lanza, G. (2024, 30. April). Human-AI Interaction in Industrial Robotics: Design and Empirical Evaluation of a User Interface for Explainable AI-Based Robot Program Optimization. arXiv.org. https://arxiv.org/abs/2404.19349

Barocas, S., Friedler, S., Hardt, M., Kroll, J., Venkatassubramanian, S., & Wallach, H., (2018),
The FAT-ML Workshop Series on Fairness, Accountability, and Transparency in Machine Learning.

Bossard, L., Guillaumin, M., Van Gool, L. (2014). Food-101 – Mining Discriminative Components with Random Forests. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8694. Springer, Cham. https://doi.org/10.1007/978-3-319-10599-4_29

Bosco, Alessandro (26 March 2024) Transfer Learning: Feature Extraction and Fine tuning. Data Reply IT | DataTech

Brownlee, Jason (20 August 2020). "A Gentle Introduction to the Rectified Linear Unit (ReLU)". Machine Learning Mastery. Retrieved 2025.

Convolutional Neural Networks. Applied Sciences, 12(17), 8643. https://doi.org/10.3390/app12178643

Geron, A. (2019) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems. 2nd Edition, O'Reilly Media, Inc., Sebastopol. - References - Scientific Research Publishing. (p.91). https://www.scirp.org/reference/referencespapers?referenceid=3265407

GitHub - https://github.com/NicasFrank/MensAI

GitHub - https://github.com/Benedikt-Schuster/Classification-of-cafeteria-meals

Hastie, T., Tibshirani, R. & Friedman, J. (2009). (p.222) The Elements of Statistical Learning. In Springer series in statistics. https://doi.org/10.1007/978-0-387-84858-7
Runkler, T. A. (2015). (p.91) Data mining. In Springer eBooks. https://doi.org/10.1007/978-3-8348-2171-3

Heaton, J. (2018). Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 191-200, isbn: 0262035618. Genetic programming and evolvable machines, 19(1), 305-307.

ImageNet - https://www.image-net.org/ 2025

Indiana University. (2024). Understand measures of supercomputer performance and storage system capacity, Retrieved January 20, 2025, from, https://servicenow.iu.edu/kb?id=kb_article_view&sysparm_article=KB0023145

Jetpack-Compose - https://developer.android.com/compose

Jin, W., Fan, J., Gromala, D., Pasquier, P., Li, X. & Hamarneh, G. (2022, 18. August). Transcending XAI Algorithm Boundaries through End-User-Inspired Design. arXiv.org. https://arxiv.org/abs/2208.08739

KPMG AG Wirtschaftsprüfungsgesellschaft 2024, KPMG-Consumer-Barometer 03/2024: Autonome Kassenprozesse im Einzelhandel. https://hub.kpmg.de/de/consumer-barometer-03-2024?utm_campaign=Consumer%20Barometer%2003%2F24&utm_source=themenseite&

__hstc=85194310.afbcbf41b9208de6ecd49007160932ad.1738065868786.1738065868786.1738065868786.1&__hssc=85194310.1.1738065868786&__hsfp=4247000830

Krizhevsky, A. et al. (2012): ImageNet Classification with Deep Convolutional Neural Networks. In: Advances in Neural Information Processing Systems 25. Hrsg. von F. Pereira et al. Curran Associates, Inc, 2012, p. 1097–1105.

Lent, M., Fisher, W., & Mancuso, M., (2004), An Explainable Artificial Intelligence System for Small-unit Tactical Behavior, 900–907.

Molnar, C., (2022), Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (2. Aufl.), Last accessed on 19. September 2024, https://christophm.github.io/interpretable-ml-book/index.html

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958.

PytorchAndroid - https://pytorch.org/mobile/android/

Ranganathan, B. (2025, January 28). Understanding inference time compute. dzone.com. https://dzone.com/articles/understanding-inference-time-compute

Ribeiro, M. T., Singh, S., & Guestrin, C., (2016), "Why Should I Trust You?": Explaining the Predictions of Any Classifier, http://arxiv.org/pdf/1602.04938v3

Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. Nature 323, 533–536 (1986). https://doi.org/10.1038/323533a0

Scikit-learn – https://scikit-learn.org/stable/

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626).

Studentenwerk Leipzig 2024, Jahresbericht 2023. https://www.studentenwerk-leipzig.de/ext/uploads/2024/06/240611_Studentenwerk_Jahresbericht_2023_web.pdf

Tan, M., & Le, Q.V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ArXiv, abs/1905.11946.

TorchScript - https://pytorch.org/docs/stable/jit.html

Torchprofile - https://github.com/zhijian-liu/torchprofile

Tharwat, A. (2018). Classification assessment methods. Applied Computing and Informatics, 17(1), 170–175. https://doi.org/10.1016/j.aci.2018.08.003

Weiss, K., Khoshgoftaar, T.M. & Wang, D. A survey of transfer learning. J Big Data 3, 9 (2016). https://doi.org/10.1186/s40537-016-0043-6

Weitz, K., Zellner, A. & André, E. (2022b, Oktober 7). What Do End-Users Really Want? Investigation of Human-Centered XAI for Mobile Health Apps. arXiv.org. https://arxiv.org/abs/2210.03506

Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

Yu-Hsin Hung & Chia-Yen Lee, (2024), BMB-LIME: LIME with modeling local nonlinearity and uncertainty in explainability, Knowledge-Based Systems, 294, 111732, https://doi.org/10.1016/j.knosys.2024.111732

Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., Dutta, A. K., & Almotairi, S. (2022). A Comparison of Pooling Methods for Convolutional Neural Networks. Applied Sciences, 12(17), 8643. https://doi.org/10.3390/app12178643

Zafar, Adeel (21 September 2021) The Learning Rate Finder
Improving our deep learning model using learning rate finder. Analytics Vidhya