Duale Hochschule
Baden-Württemberg
Mannheim

Duale Hochschule Baden-Württemberg Mannheim

Projectreport

Coin Counter Application

Business Information Systems

Study Program Data Science

Authors:	Nik Yakovlev, Aref Hasan
Matriculation Numbers:	4144212, 1335610
Course:	WWI-21-DSA
Program Director:	Prof. Dr.-Ing. habil. Dennis Pfisterer
Modul:	Machine Learning Project
Lecturer:	Prof. Dr. Maximilian Scherer
Submission Date:	24.07.2024

Contents

List of Figures

1 Introduction

In an era characterized by rapid digital advancements, the necessity for efficient and accurate financial transaction management is very important. One enduring challenge that continues to hinder productivity across various sectors is the manual counting of coins. This process, whether conducted by small business owners managing cash transactions, non-profit organizations tallying donations, or individuals sorting through personal collections, is inherently time-consuming and prone to human error.

Our project addresses this problem through the development of an innovative coin counter application, which leverages the capabilities of computer vision and machine learning. By employing state-of-the-art image recognition algorithms, this application is designed to automatically identify and count coins from a single uploaded image. The core of our solution lies in its ability to accurately discern different coin denominations based on their visual characteristics, even under varying conditions such as lighting, angle, and wear.

The application utilizes ResNet-18, a deep residual network, for coin classification [1]. ResNet-18's architecture, known for its efficiency and accuracy in image classification tasks, enables our application to learn and recognize the subtle features that distinguish each coin denomination.

For object detection, our application employs YOLOv8 (You Only Look Once, version 8), a cutting-edge model known for its speed and precision [8, 11]. YOLOv8 excels in detecting multiple objects within an image, allowing our application to identify and count several coins simultaneously. This capability is crucial for handling images with multiple coins, ensuring that each coin is accurately detected and classified.

The integration of ResNet-18 and YOLOv8 provides a robust and efficient framework for our coin counter application. The user-friendly interface simplifies the process of uploading images and obtaining results, making the technology accessible to users with varying levels of technical proficiency.

2 Related Work

2.1 Coin Detection and Recognition Using Neural Networks

In the realm of coin recognition systems, significant advancements have been made utilizing neural networks for object detection and classification. The paper titled "Coin Detection and Recognition Using Neural Networks" by S. Mohamed Mansoor Roomi and R.B. Jayanthi Rajee provides a comprehensive approach to coin recognition using a combination of image processing techniques and neural networks [9].

The authors of this paper address the challenges of coin recognition, such as varying rotations and patterns on coins, by implementing a Fourier Transform to account for surface variations and utilizing a multilayer backpropagation (BP) neural network for classification. The method proposed by Roomi and Rajee involves capturing coin images under controlled illumination conditions, applying image segmentation, edge detection, and Hough Transform to detect circular shapes, followed by Fourier Transform to extract rotation-invariant features. These features are then fed into a neural network trained to classify coins based on denomination [9].

This method stands out for its robustness to rotation and translation, leveraging the invariant properties of the Fourier coefficients. The neural network's training involves a significant database of coin images to ensure high accuracy in recognition. The authors report a recognition rate of around 82%, indicating a strong performance for the proposed system. This work serves as a foundational reference for implementing neural networks in the domain of coin recognition and demonstrates the effectiveness of combining classical image processing with advanced neural network techniques for object detection and classification [9].

By comparing this approach with other methodologies in the literature, such as mechanical and electromagnetic systems, the image-based neural network approach offers superior accuracy and robustness, making it a valuable reference for further developments in automated coin counting and recognition systems [9].

2.2 Deep Residual Learning for Image Recognition

He et al. introduced the concept of deep residual learning, which addressed the degradation problem in deep neural networks by reformulating the learning objective as learning residual functions instead of direct mappings. This approach allowed for the training of extremely deep networks, achieving significant improvements in various benchmarks including ImageNet classification, detection, and localization tasks. The ResNet architecture, particularly ResNet-50 and ResNet-101, demonstrated state-of-the-art performance, significantly outperforming previous models such as VGG-16 and GoogLeNet in terms of accuracy and efficiency. This work laid the foundation for using deep residual networks in various computer vision applications, highlighting their robustness and generalization capabilities across different datasets and tasks [1].

2.3 YOLO: You Only Look Once

The original YOLO (You Only Look Once) paper by Redmon et al. revolutionized object detection by reframing the task as a single regression problem, directly predicting bounding boxes and class probabilities from full images in one evaluation. This approach significantly simplified the object detection pipeline, allowing for real-time processing with high accuracy. YOLO's unified architecture processes images at 45 frames per second, and a faster variant, Fast YOLO, achieves up to 155 frames per second. This efficiency is achieved without compromising on detection accuracy, making YOLO highly suitable for applications requiring real-time object detection [8].

Building on the foundations laid by YOLO, the YOLOv7 paper presents several enhancements aimed at further improving the speed and accuracy of real-time object detection. YOLOv7 introduces innovative training techniques and architectural optimizations, such as model re-parameterization and dynamic label assignment, which enhance the training efficiency and detection accuracy without increasing the inference cost. The YOLOv7 model demonstrates state-of-the-art performance across various benchmarks, significantly reducing the number of parameters and computation required compared to previous versions while maintaining faster inference speeds [11].

While YOLOv8 has been implemented and utilized in various applications, there is currently no official YOLOv8 paper available. The advancements and optimizations from YOLOv7

continue to serve as the most recent documented progress in the YOLO series, informing the development and deployment of subsequent versions like YOLOv8. This absence highlights the ongoing evolution and practical application-driven improvements in the YOLO family of models.

3 Methodology

This chapter explains the methods and approaches we used to build the Coin Counter Application.

3.1 Object Detector

To achieve precise and efficient coin detection, our project employs an object detection model based on the YOLOv8 architecture, leveraging the capabilities of the Roboflow platform for model deployment. The following outlines the key steps and methodologies involved in integrating the object detector component of our coin counter application.

Instead of training a custom model, we utilized the pre-trained YOLOv8-based object detection model available through the Roboflow API [10]. This model was specifically designed for object detection tasks and was pre-trained on a vast dataset with US-coins, making it well-suited for our application of recognizing coins [10].

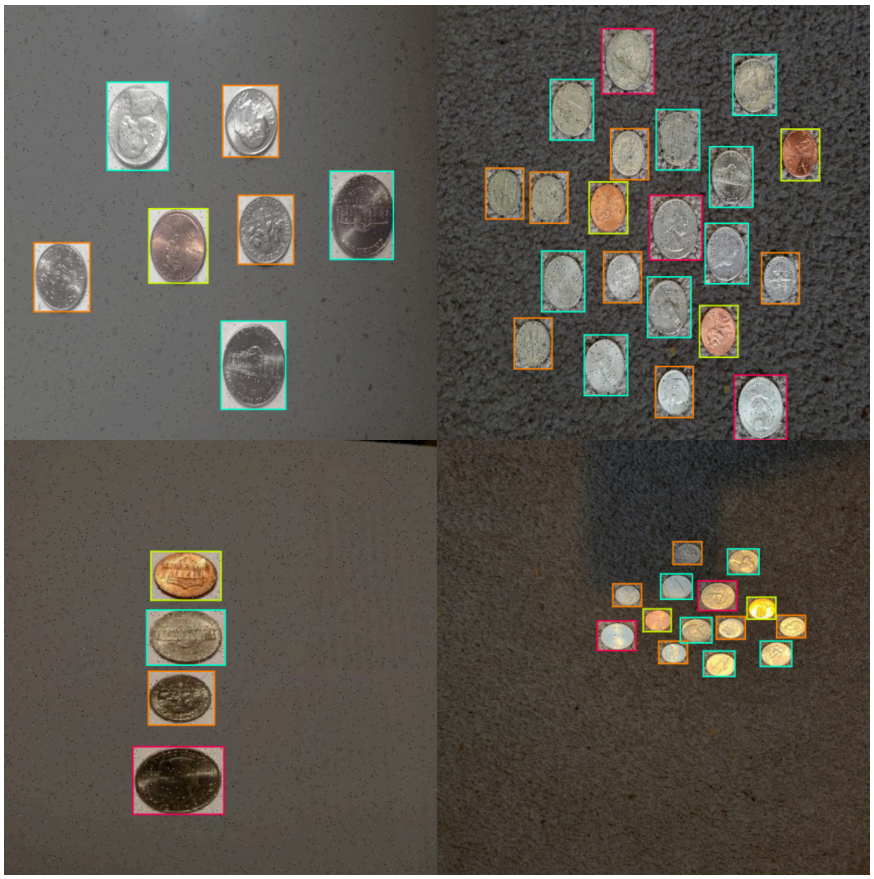

Figure 3.1: Object-Detector-Training-Data
Quelle: [10]

The integration of the YOLOv8 model into our coin counter application was achieved through several steps. We integrated the Roboflow API into our application code, writing functions to send coin images to the API endpoint and receive the detection results. Upon uploading an image to the application, the image is sent to the Roboflow API, which returns the detected bounding boxes for each coin in the image. This information is then processed to create cropped images which are then sent to the classifier part of our Coin Counter.

By utilizing the YOLOv8 architecture through the Roboflow API, our object detector component effectively identifies and classifies coins in images, forming a crucial part of our Coin Counter application.

3.2 Classifier

To complement our object detection model and achieve precise coin recognition, our project employs a classifier based on the ResNet-18 architecture [1]. This classifier is specifically trained to identify different denominations of Euro coins. The following outlines the key steps and methodologies involved in developing the classifier component of our coin counter application.

We utilized a pre-trained ResNet-18 classifier from PyTorch [5], leveraging transfer learning to adapt the model to our specific task. The pre-trained ResNet-18 model, which has been trained on the ImageNet dataset, provides a robust foundation for recognizing a wide range of image features. This allows us to build on this foundation and specialize the model for Euro coin classification with relatively less training data and computational resources.

Our training dataset comprised approximately 1000 close-up images of each Euro coin denomination [3, 2]. These images were collected and annotated to ensure accurate labeling, providing a diverse set of examples for each coin type. The dataset included coins of different conditions, lighting variations, and orientations to enhance the model's robustness.

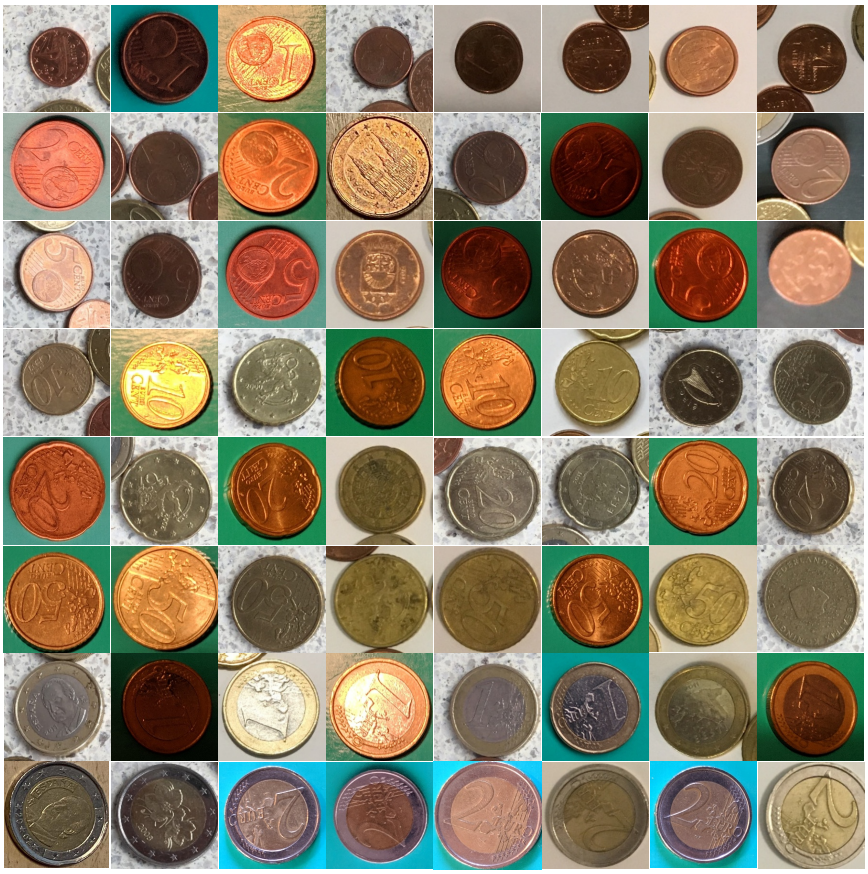

Figure 3.2: Classifier-Training-Data
Quelle: [3, 2]

The training process involved several key steps. First, we fine-tuned the pre-trained ResNet-18 model using our annotated dataset. This involved replacing the final fully connected layer of the ResNet-18 model with a new layer tailored to our specific classification task, corresponding to the number of Euro coin denominations. We then trained this modified model using our dataset, adjusting hyperparameters such as learning rate, batch size, and number of epochs to optimize performance. The training was conducted using PyTorch, a flexible and efficient deep learning framework [4, 7, 6].

3.3 Combining Object Detector and Classifier

Once the classifier was trained, we integrated it with our object detection model to create a seamless coin counting system. The object detector, based on the YOLOv8 architecture, identifies and localizes coins within an uploaded image, returning the bounding boxes for each detected coin. These bounding boxes are then used to crop the detected coin regions from the image.

The cropped images are subsequently passed to the trained ResNet-18 classifier, which predicts the denomination of each coin. The classifier outputs are combined with the bounding box information from the object detector to accurately count and categorize the coins in the original image. This integrated approach ensures that our coin counter application can effectively handle multiple Euro coins in a single image, providing accurate counts and denominations for the user.

3.4 Application

For this project, a web application was developed using Flask, a lightweight and highly flexible web framework for Python. Flask is designed to facilitate the rapid and straightforward creation of web applications while offering the scalability necessary for more complex implementations. Flask's core capabilities include routing, template rendering, and HTTP request handling, making it an ideal choice for this application.

The Coin Counter application is designed to provide users with an efficient way to count and categorize coins using image recognition technology. By leveraging machine learning and image processing techniques, the application aims to simplify the process of coin counting and provide accurate results quickly.

Flask handles URL routing through a series of route definitions. Each route in the application corresponds to a specific URL endpoint and a function that processes the request and returns a response. The primary routes in this application include the homepage (/), the image upload page (/upload), the photo capture page (/take-photo), the QR code generation page (/scan_qr_code), the mobile-specific upload page (/phone_page), and the results page (/result). This modular approach ensures a clear and maintainable codebase. Users are provided with three methods to upload images: directly from the computer, capturing a photo using the device's camera, or utilizing a smartphone to upload images.

HTML templates define the structure of the web pages. These templates are stored in the templates directory and rendered using Flask's render_template function. This function dynamically inserts content into predefined HTML structures, enabling the creation of interactive and data-driven web pages. Static files such as CSS, JavaScript, and images are stored in the static directory, ensuring a clean separation between the application's logic and its presentation. HTML provides the structural framework, CSS enhances the visual styling, and JavaScript handles client-side interactivity, ensuring a responsive and engaging user experience.

The application facilitates image uploads through a dedicated form on the Upload Page. The uploaded files are saved in the static/uploads directory. The file upload process is managed through an HTML form, which posts the image data to a Flask view function. This function processes the image, performs the necessary computations (such as detecting coins and calculating their total value), and returns the results to be displayed on the Result Page. This integration showcases Flask's ability to handle file uploads, perform server-side processing, and render dynamic content.

To enhance accessibility from mobile devices, the application generates a QR code that points to the local IP address of the server. This feature allows users to upload images from their smartphones directly to the web application running on a local machine. The QR code functionality is implemented using the qrcode library, which generates a scannable QR code linking to the upload page. This approach demonstrates the seamless integration of third-party libraries into Flask applications to extend functionality.

Recognizing the importance of mobile accessibility, the application includes dedicated routes and templates optimized for mobile use. The phone_page route is specifically designed for mobile users to upload images, ensuring a seamless experience on smartphones. This page is tailored to the constraints and capabilities of mobile devices, providing large, touch-friendly buttons and a streamlined interface.

In the following section, we will provide detailed steps and screenshots to illustrate the workflow of the Coin Counter application. This will showcase the user interface and functionality through various stages, offering a clear visual representation of the process.

4 Coin Counter App

4.1 Evaluating the Performance of the Coin Counter

4.1.1 How did we evaluate?

To test the performance of our Coin Counter Application, we created a comprehensive dataset of 100 images featuring multiple Euro coins of different denominations. Each image was accompanied by a corresponding ground truth sum, representing the total monetary value of the coins present in the image. This dataset served as the benchmark for evaluating the accuracy and reliability of our model.

The evaluation process involved several key steps. First, we fed the test images into our Coin Counter Application, which used the YOLOv8-based object detector to identify and localize the coins within each image. The detected coins were then classified by the ResNet-18 classifier to determine their denominations. Finally, the application calculated the sum of all detected coins for each image.

We compared the calculated sums from our model to the actual ground truth sums to evaluate the model's performance. The following metrics were used to assess the accuracy and reliability of the model:

- Accuracy: Accuracy measures the overall correctness of the model in calculating the total sum of coins. It is defined as the number of correctly calculated sums divided by the total number of images.

4.1.2 Results of the Evaluation

After running all the evaluation images through our Coin Counter App, the coins of 81 of 100 images were summed up correctly. So based on this result the accuracy of our Coin Counter is 0.81.

The following figure 4.1 shows some examples from our evaluation process:

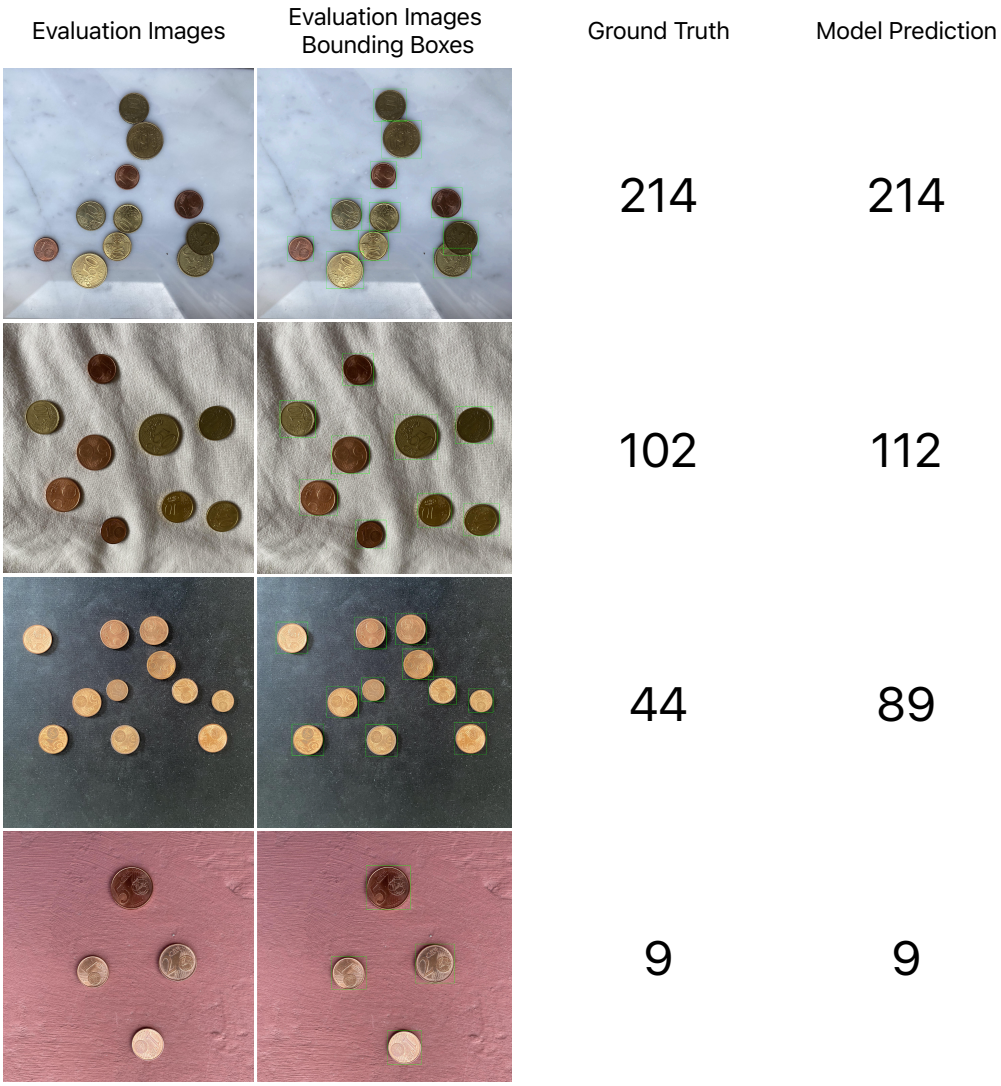

Figure 4.1: Evaluation-Process-Examples
Quelle: own figure

4.2 The finished Application

The following steps outline the workflow of the Coin Counter application, showcasing its user interface and functionality through various stages. Each step is illustrated with screenshots to provide a clear visual representation of the process.

Step 1: Home Page The Home Page serves as the entry point to the Coin Counter

application. It welcomes users with a simple interface and provides options to start using the application. The interface includes a call-to-action button that directs users to the upload options.

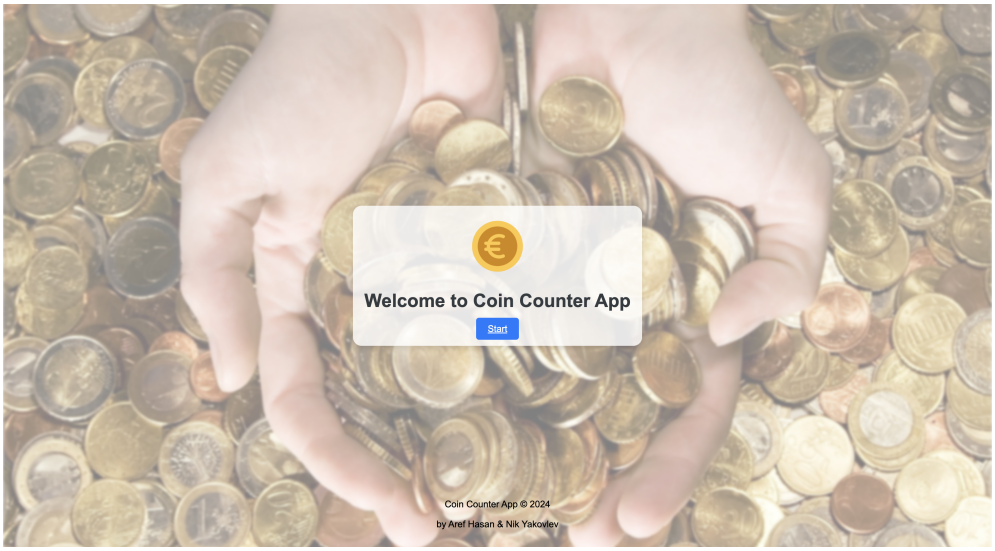

Figure 4.2: Home Page
Source: Own Figure (Screenshot)

Step 2: Upload Options Users are presented with three options for uploading an image:

- Choose File: Allows users to upload an image from their computer.
- Take a Photo: Opens the device's camera to capture a new photo.
- Scan QR Code: Generates a QR code that users can scan with their mobile devices to upload an image.

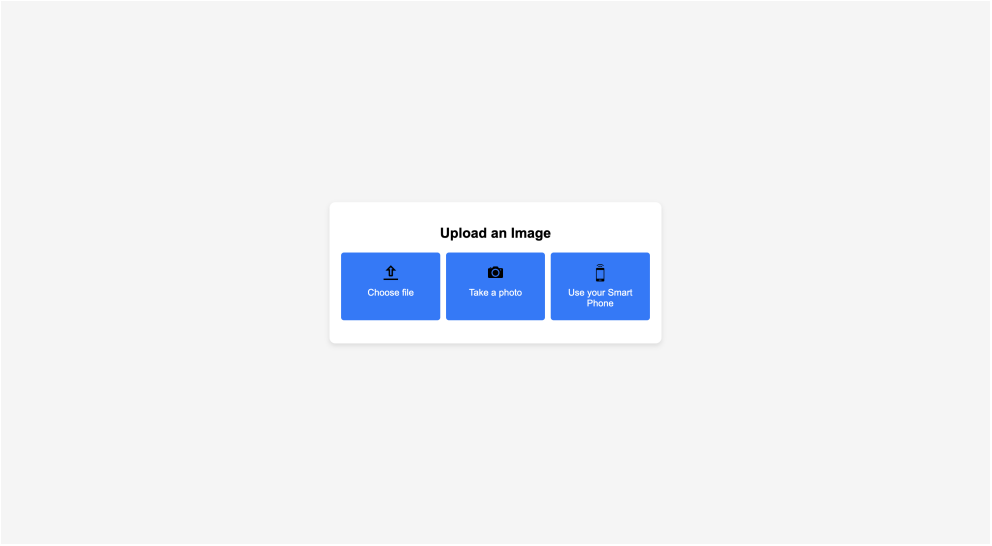

Figure 4.3: Upload Options Page
Source: Own Figure (Screenshot)

Step 3: Image Upload When users choose to upload an image from their computer, they are directed to a file upload page where they can select and upload an image file. The uploaded file is then processed by the server.

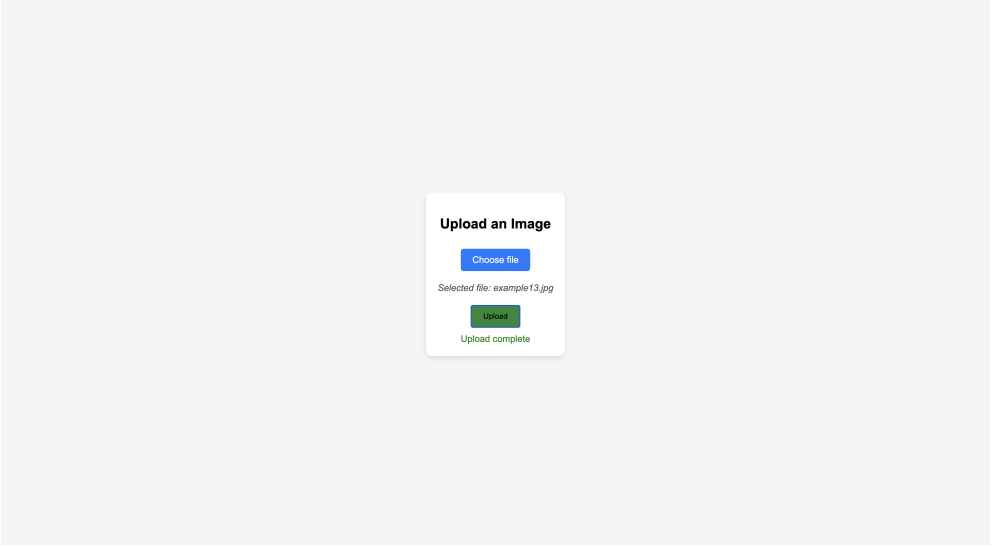

Figure 4.4: Image Upload Page
Source: Own Figure (Screenshot)

Step 4: Photo Capture If users select the photo capture option, the application activates the device's camera, allowing users to take a new photo, which is then uploaded and

processed.

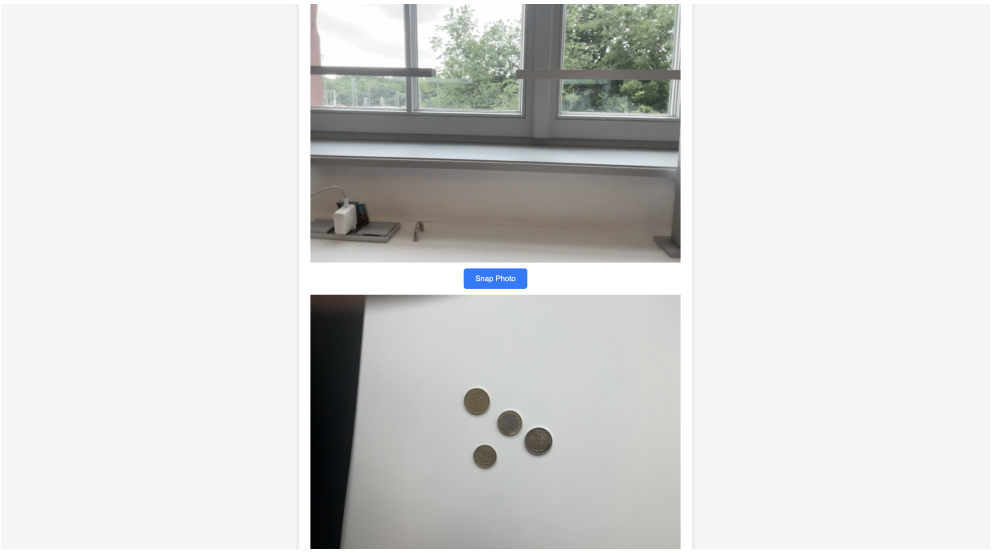

Figure 4.5: Titel
Source: Own Figure (Screenshot)

Step 5: QR Code Scanning Users can also scan a QR code generated by the application, which allows them to upload an image from their mobile device directly to the web application.

Figure 4.6: QR Code Page
Source: Own Figure (Screenshot)

Step 6: Phone Page After scanning the QR code, users are directed to a mobile-optimized

upload page, allowing them to upload an image from their smartphone.

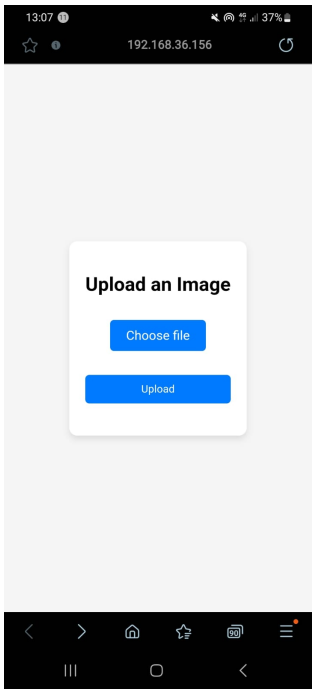

Figure 4.7: Upload Page - Smart Phone
Source: Own Figure (Screenshot)

Step 7: Result Page The Result Page displays the results of the image processing. It shows the original image, the processed image with detected coin bounding boxes, and a summary of the detected coins and their total value.

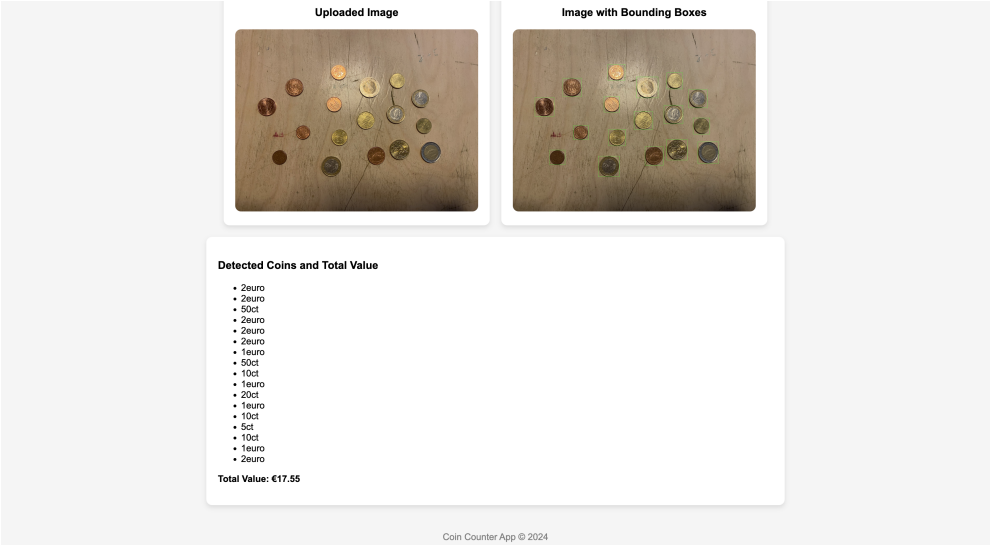

Figure 4.8: Results Page
Source: Own Figure (Screenshot)

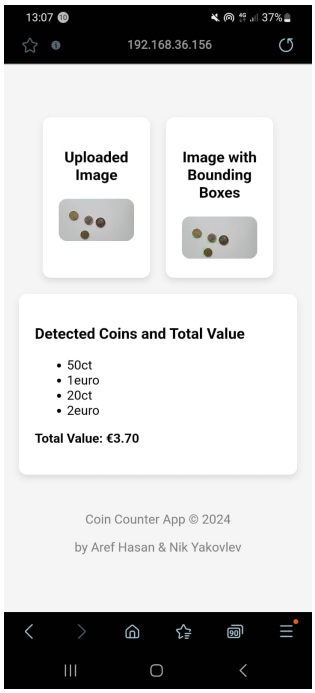

Figure 4.9: Results Page - Smart Phone
Source: Own Figure (Screenshot)

These steps demonstrate the functionality and user interface of the Coin Counter application, highlighting its capabilities in handling different methods of image input and providing a user-friendly experience.

5 Conclusion

In conclusion, the development of the Coin Counter application marks a significant achievement in utilizing advanced machine learning techniques for practical and efficient financial transaction management. By integrating image recognition technologies such as ResNet-18 and YOLOv8, the application successfully provides a robust solution for the rapid and accurate counting of coins, substantially reducing the inefficiency associated with manual counting.

Looking forward, there are several avenues for further enhancing the application's capabilities and extending its utility. Future improvements could focus on increasing the model's accuracy and reliability under more diverse real-world conditions, such as varied lighting and coin orientations. Additionally, expanding the application to recognize and count coins from multiple currencies could significantly broaden its applicability on a global scale. Another promising development could be the introduction of live recognition features, allowing the application to provide real-time coin counting during transactions.

Bibliography

[1] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385.

[2] kaa. *coins-dataset*. https://github.com/kaa/coins-dataset. Accessed: 2024-06-28. 2018.

[3] Pitrified. *coin-dataset*. https://github.com/Pitrified/coin-dataset. Accessed: 2024-06-28. 2019.

[4] PyTorch Contributors. *PyTorch Vision Documentation*. https://pytorch.org/vision/stable/index.html. Accessed: 2024-06-28. 2017.

[5] PyTorch Contributors. *ResNet18 - torchvision.models*. https://pytorch.org/vision/master/models/generated/torchvision.models.resnet18.html. Accessed: 2024-06-28. 2017.

[6] PyTorch Contributors. *torch.nn - PyTorch Documentation*. https://pytorch.org/docs/stable/nn.html. Accessed: 2024-06-28. 2023.

[7] PyTorch Contributors. *torch.optim - PyTorch Documentation*. https://pytorch.org/docs/stable/optim.html. Accessed: 2024-06-28. 2023.

[8] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640.

[9] S. Mohamed Mansoor Roomi and R. B. Jayanthi Rajee. "Coin detection and recognition using neural networks". In: *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*. 2015, pp. 1–6. DOI: 10.1109/ICCPCT.2015.7159434.

[10] shane testing. *coin counting 2 Dataset*. https://universe.roboflow.com/shane-testing/coin-counting-2. Open Source Dataset. visited on 2024-06-25. Dec. 2023. URL: https://universe.roboflow.com/shane-testing/coin-counting-2.

[11] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. arXiv: 2207.02696.