

*By submitting this portfolio the authors certify that this is their original work, and they have cited all the referenced materials properly.*

Deep Learning report

Lucas Bolwidt, 1583773

Stijn Stuiver, 2156976

Denis Kozlov, 1633335

Minor: Vision and machine learning

Group: 3

Date: 17-12-2024

# INSTRUCTIONS FOR PROJECT submission

**When submitting your project results, please follow these steps:**

1. **Rename this file** 
   * **Replace {YOUR GROUP\_NUMBER} with your group number**
   * **Replace {YOUR\_NAME} with your name**
   * **Replace {YOUR\_STUDENT\_NUMBER} with your student number**
2. **Delete this instruction page**
3. **Prepare the project results:**
   * **Report as a Word file**
   * **code in a zip file**
   * **deployment video**
4. **Per group, email the project results to the instructor**
5. **Upload your project results to HAND-IN** 
   * **Each student must upload INDIVIDUALLY**
   * **Use your renamed file**
   * **PLEASE DO NOT UPLOAD YOUR DATA AND MODEL.**

Contents

[1 Introduction 3](#_Toc88129256)

[2 Problem statement 4](#_Toc88129257)

[3 Data augmentation and preprocessing 5](#_Toc88129258)

[4 CNN architecture and training 6](#_Toc88129259)

[5 Deploy and test 7](#_Toc88129260)

[6 Conclusion 8](#_Toc88129261)

[7 References 9](#_Toc88129262)

[Code appendices 10](#_Toc88129263)

# Introduction

This report presents a collection of machine learning projects developed during the Vision and Machine Learning minor. These projects explore a range of tasks, including image classification where a combination of provided scripts and OpenCV to process and analyze visual data was utilized. Since none of the team members had experience with machine learning or computer vision before starting this minor, these projects offered an invaluable opportunity to build foundational knowledge and gain hands-on experience in his field.

The machine learning class introduced us to fundamental concepts, including supervised and unsupervised learning. Supervised learning involves training models on labeled datasets, allowing the system to make predictions or classifications based on input data. Unsupervised learning, on the other hand, focuses on identifying patterns and relationships within unlabeled data. By combining machine learning with computer vision techniques, an implementation of object recognition systems, where feature extraction is employed to identify and classify different objects in an image. This integration of machine learning and vision is a critical component of the EVML minor, as it enables us to develop systems capable of recognizing patterns, objects, or specific features in real-time.

Machine learning is revolutionizing industries such as healthcare and manufacturing, both of which are areas of interest for members the team. In healthcare, it can be used for predictive diagnostics, while in manufacturing, machine learning enhances quality control and process optimization. These transformative applications of machine learning underscore its importance in modern industry, aligning well with the skills that are developed in this minor.

Initially, the main project of the group focused on was the ABB-beamer system. This project required the group to recognize multiple steps in the assembly process of an ABB product and provide instructions to the user via a projector. The system is aiming to create an interactive environment where workers receive real-time guidance and corrections during the assembly process. To achieve this, machine learning needs to be used to identify and verify different parts of the assembly and ensure that they were correctly assembled.

However, after encountering technical challenges with segmentation and feature extraction with the project data, it was decided to shift focus. Despite multiple attempts, the desired results couldn’t be achieved within the time constraints, leading the group to pivot to a new project: coin recognition. For this new project, the focus was put on recognizing different coins, specifically the 5-cent, 1-euro, and 2-euro coins. The existing scripts were rewritten to adapt to this new objective, using machine learning and computer vision techniques to differentiate between features of the coins.

The primary learning objective of this project is to gain a deeper understanding of machine learning principles, particularly in how it integrates with vision systems. The aim to learn how to effectively apply feature extraction and object recognition techniques to the dataset of coins, and to refine skills in differentiating objects based on their visual features. This experience will not only contribute to academic development but also provide practical skills applicable to real-world problems in industries where machine learning plays an increasingly vital role.

# Problem statement

## Problem definition

|  |  |
| --- | --- |
| Specific Measurable Achievable Relevant Time-bound | |
| **S** | The problem this project tackles is the differentiation of different coins. |
| **M** | The solution must be able to work with 95 to 99% performance to recognize 3 different coins. 5 cent, 1 euro and 2 euro. |
| **A** | The problem must be solved using Machine Learning approach. |
| **R** | The implementation must include modules and theory background from lectures and provide solutions to the given assignments. |
| **T** | The problem must be solved within the first Term period during EVML Minor. |

Tabel , problem definition

## Requirements

|  |  |
| --- | --- |
| **Requirement** | **Description/Value** |
| **Functional requirements** | |
| Data Acquisition | The system should be able to capture images of 5 Cent, 1 Euro and 2 Euro coins. |
| Preprocessing | The system should preprocess images to enhance quality, including resizing, noise reduction, and background removal. |
| Segmentation | The system should segment the coin from the background to isolate it for further analysis. |
| Feature Extraction | The system should extract relevant features from the coin images, such as extent, perimeter length and dominant hue. |
| Classification | The system should classify the coins into three different categories using a supervised machine learning model. |
| Model Training and Validation | The system should include a process for training the model with labeled data and validating its performance with a separate validation set. |
| Accuracy and Performance | The system should perform with accuracy and  precision within 90-99%. |
| Non-functional requirements | |
| Documentation | The project should include comprehensive documentation for users and developers, detailing the setup, usage, and maintenance of the system. |
| Operating distance | 85mm |
| Camera model | OnePlus Nord 3 5G |
| Camera angle | 90 Degrees |
| Environment | The tests are performed under a normal room lighting using a piece of plain color cloth as a background. |

Tabel , requirements

# Data augmentation and preprocessing

|  |  |
| --- | --- |
| Assignment | 1. Increase your data set Start with your ML image set Choose data augmentation methods Apply these methods to create new images Explain why you chose these methods 2. Prepare images for CNNs Choose preprocessing steps to: - Make patterns clearer for CNNs, or - Reduce CNN complexity Consider these factors: - Image size - Color depth - Image enhancement -Normalization   Build a preprocessing pipeline  Explain each step in your pipeline   1. Test your preprocessing Run some images through your pipeline Show before and after examples Explain how each step improves the images |
| Acceptance criteria | Data augmentation methods are used and explained  Preprocessing pipeline is implemented  Each preprocessing step is explained and justified |
| Size | Max 3 A4 |

To enhance the dataset for our machine learning assignment, a custom script has been developed to perform various data augmentation techniques. These augmentations include:

* **Rotation**
* **Shearing**
* **Scaling**
* **Intensity adjustment**
* **Contrast and brightness modifications**

**Rationale for Selected Augmentations**

* **Rotation and Shearing:** These augmentations simulate changes in the camera angle, ensuring the dataset is not constrained to a fixed viewpoint. This improves the model's robustness to perspective variations.
* **Contrast, Brightness Adjustments and Intensity adjustment:** These modifications mimic different lighting conditions, helping the model handle varying illumination levels and improving its colour detection performance.
* **Scaling:** Scaling introduces a zoom effect, addressing the issue of the model's sensitivity to working distance. Previously, slight changes in working distance caused the model to fail, and scaling ensures the model can generalize better across different distances.

**Why a Custom Script?**

A custom script was chosen over TensorFlow's built-in augmentation methods due to the complex texture of the background in our images. TensorFlow's augmentations struggled with the intricate patterns in the background, leading to suboptimal results. This issue is illustrated in **Figure 2**, where TensorFlow's augmentations produce noticeable artifacts. In contrast, Figures **3 to 5** demonstrate the results of our custom script, which handles the background texture more effectively and preserves the quality of the augmented images.

By implementing these augmentations with a tailored approach, we have successfully improved the dataset's diversity and addressed key challenges, such as camera angle, lighting conditions, and working distance, ultimately enhancing the model's performance and robustness.

A computer screen shot of a program code

Description automatically generated

A close up of a button

Description automatically generatedA coin on a purple surface

Description automatically generated A coin on a purple surface

Description automatically generatedA coin on a purple surface

Description automatically generated

To reduce the computational complexity for the CNN, the images will be resized to a resolution of **224 x 224**. This resizing is done alongside the same segmentation approach used in the machine learning model, where the **purple background is removed**.

**Improvements to the Segmentation Script**

The segmentation script has been updated to account for a broader range of purple shades, ensuring better segmentation performance, especially when working with the newly augmented dataset. This refinement guarantees more robust separation of the coin from the background, regardless of subtle variations in color.

**Normalization and Input**

Once the segmentation is complete, the images are normalized to ensure compatibility with TensorFlow. The **final input to the CNN** consists solely of the segmented coin, as illustrated in **Figure 5**.

By focusing on the segmented coin and resizing the images, the dataset becomes more streamlined, reducing complexity while maintaining high-quality inputs for the CNN.

A collage of different colored objects

Description automatically generated

# CNN architecture, training and validation

|  |  |
| --- | --- |
| Assignment | 1. Design CNN architecture and/or use transfer learning Design a custom CNN:  - Decide number of layers  - Choose number of neurons per layer  - Select pooling methods  - Pick activation functions Use transfer learning:  - Select a pretrained model - Decide which layers to freeze  - Design new top layers for your specific task Explain each choice in your design 2. Train and optimize CNN Choose appropriate performance measures Consider accuracy, precision, recall, F1-score and explain Train your CNN using the training set Use cross-validation to check performance Optimize hyperparameters, e.g.: try different learning rates, adjust batch sizes, experiment with optimizer types   Prevent overfitting (explain which method you used and why)   1. Test your model Use the test set to check model performance Create confusion matrix Compute performance measures Check for overfitting or underfitting Discuss trade-offs, e.g. precision vs. recall or bias vs. variance 2. Visualize CNN learning Show how an input image transforms through your network Visualize at least 3 different layers Explain what features each layer detects |
| Acceptance criteria | Architecture is designed and argued.  Data is split into stratified subsets and checked.  CNN is trained, cross-validated, and fine-tuned.  Performance is evaluated using appropriate methods.  Visualization of network's internal representations is provided |
| Size | Max 5 A4 |

# Deploy and test

|  |  |
| --- | --- |
| Assignment | 1. Deploy your CNN model  Set up preprocessing and prediction pipeline Choose where to run your model: - On your target machine (e.g., Raspberry Pi) - Or on your training machine 2. Make a test plan Review your SMART problem definition List requirements to measure, such as: - Model performance (e.g., accuracy, precision) - Inference speed (e.g., frame rate) - Technical factors (e.g., camera angles, distances, lighting conditions) Set target levels for each measure Explain how you will test each measure 3. Conduct tests Run tests based on your plan Record all test results Compare results to your targets Note any unexpected behaviors or limitations 4. Document your work Write down your test plan Record all test results Explain any differences between results and targets |
| Acceptance criteria | Preprocessing and prediction pipeline deployed.  Test plan present.  Documentation of test results. |
| Size | Max 5 A4 |

# Conclusion

|  |  |
| --- | --- |
| Assignment | 1. Summarize your project List the main steps you took Explain your key decisions 2. Evaluate your results Compare your results to your initial goals Discuss if you met your SMART objectives Explain any differences between goals and results 3. Reflect on generalization performance Discuss how well your model works on new, unseen data Compare performance on training, validation, and test sets Explain any differences in performance across these sets 4. Analyze your approach Identify what worked well Point out areas for improvement Suggest changes for future projects |
| Acceptance criteria | Results are compared to initial goals and SMART objectives  Generalization performance is analyzed |
| Size | Max 1 A4 |

# References

|  |  |
| --- | --- |
| Assignment | Give references to the sources that you have used. |

|  |  |
| --- | --- |
| [1] | „SMART criteria,” 14 05 2020. [Online]. Available: https://en.wikipedia.org/wiki/SMART\_criteria. |
| [2] | A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, Sebastopol, Canada.: O’Reilly Media, 2019. |

# Code appendices

|  |  |
| --- | --- |
| Assignment | 1. Select key code snippets Choose important parts of your code Include snippets for: - Data preprocessing - Model creation - Training process - Evaluation methods 2. Explain your code Add comments to each snippet Explain what each part does Describe why you made specific coding choices 3. Show coding best practices  Use clear variable names Structure your code logically Follow Python style guidelines (PEP 8) |
| Acceptance criteria | Code snippets are provided for key parts of the project  Code quality is sufficient |