Your Awesome Title

xAI-Proj-M: Master Project Explainable Machine Learning

Max Mustermann[*]
Otto-Friedrich University of Bamberg
96049 Bamberg, Germany
max.mustermann@stud.uni-bamberg.de

Baptiste Bony[†]
Otto-Friedrich University of Bamberg
96049 Bamberg, Germany
baptiste-patrice-francis.bony@stud.uni-bamberg.de

Jonas R. Amling[‡]
Otto-Friedrich University of Bamberg
96049 Bamberg, Germany
jonas-reinhold.amling@stud.uni-bamberg.de

Abstract

...

1 Introduction

The goal of this research project is to apply the CRISP-ML (Cross-Industry Standard Process for the development of Machine Learning applications with Quality assurance) methodology (Studer et al., 2021) to the problem of classifying rock, paper, or scissors from an image that includes exactly one hand. Specifically, we will focus on the CRISP-ML: Deep Learning Life Cycle and explore the three main steps in the process: Data Engineering, Model Engineering, and Model Evaluation. An short overview of the complete CRISP-ML cycle is presented in Figure 1.

Data Engineering is the process of understanding and preparing the data. This involves selecting the data to use, determining the amount of data needed, cleaning the data, and augmenting the data if necessary. The first research question for this project is whether removing the background during the image preprocessing phase benefits the image classification task at hand. This question aims to understand the impact of background removal on the accuracy of the model.

Model Engineering involves building a model that meets the desired requirements, such as the model architecture, training strategies, handling overfitting, and hyperparameter tuning. Our second research question is whether dropout layers prevent overfitting and what the ideal position is for these layers in the model. This question aims to optimize the model architecture and avoid overfitting.

Finally, Model Evaluation involves validating the model performance on a test set, evaluating its robustness and generalization, and determining the correct measures to use. Our third research question is how the model performs on distorted data and whether the use of distorted test data leads to worse model performance compared to the same test data without distortion. This question explores the robustness and generalization capabilities of the model.

[*]Degree: M.Sc. AI, matriculation #: 12345678
[†]Degree: Engineering Master, matriculation #: 2117568
[‡]Degree: M.Sc. AI, matriculation #: 1867301

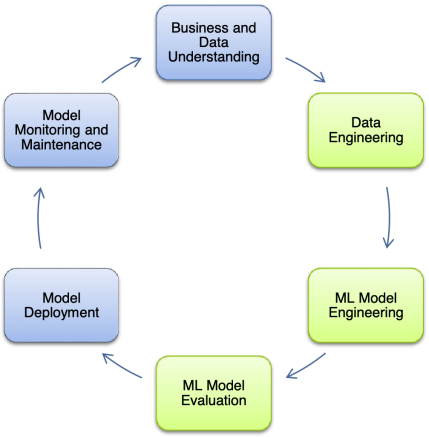


Figure 1: The CRISP-ML: Deep Learning Life Cycle as presented in the introductory presentation by Ines Rieger.

In this report, we will present our findings for each research question, explain the rationale behind each question, and set up a simple experiment to test each research question. We will discuss the implications of the findings, provide limitations of our approach, and suggest areas for future improvements. An overview of which parts of the report was written by which author is presented in Table 3.

2 Data Engineering

During ourur data engineering process we wanted to deal with problems that arise when combining data from multiple sources into a single dataset for model training. Specifically, we identified three key challenges in working with different datasets: combining images from different resolutions and formats, accounting for variations in hand position within each dataset (e.g., hands always positioned in the middle versus positioned elsewhere in the image), and dealing with different contextual backgrounds for the hands (e.g., single-color backgrounds versus real-life image backgrounds). We assumed that reducing the complexity of the datasets was a crucial factor in generating a well-performing model. Thus, our research question was formulated as follows: Does removing the background during the image preprocessing phase benefit the image classification task at hand? This research question will guide our investigation throughout the data engeneering part of the project.

In order to address the data engineering challenges we identified, we combined data from multiple datasets into a single dataset for model training. The training data was created by combining data from several sources, including self-produced images, computer-generated images, and existing datasets from Kaggle. The existing datasets from Kaggle included a dataset of hands with bodies and a dataset of hands from the top. [4] [5] The computer-generated images were obtained from the TensorFlow datasets catalog. [6] We chose to combine both computer-generated images and real-life images to increase the variety of the training data. Since the testing data was unknown during the data engineering phase of the project, we created a separate validation dataset that was provided by the project, as well as a testing dataset that was also provided by the project. We will refer to these datasets in future discussions as follows: the self-produced images as "custom," the computer-generated images as "cgi," the Kaggle dataset of hands with bodies as "webcam," and the Kaggle dataset of hands from the top as "hands." An overview of the total number of images is presented in Table **??** and the distributions are shown in Figure 2.

To streamline our dataset and remove extraneous background images, we set out to identify Python libraries capable of detecting hands in an image and eliminating the background. After thorough research, we narrowed our focus to three potential options: YOLO-Hand-Detection, rembg, and

[4] https://www.kaggle.com/datasets/drgfreeman/rockpaperscissors
[5] https://www.kaggle.com/datasets/glushko/rock-paper-scissors-dataset
[6] https://www.tensorflow.org/datasets/catalog/rock_paper_scissors

Table 1: Total number of images by source

Origin	Rock	Paper	Scissors	Total
custom	210	205	210	625
cgi	840	840	840	2520
hands	726	712	750	2188
webcam	752	733	760	2245
Total	2528	2490	2560	7578

Figure 2: Individual dataset distribution showing the number of images in each dataset

MediaPipe Hands. Although YOLO-Hand-Detection was able to detect hand positions in real-life images, it was not easily set up and ran the risk of technical difficulties.[7] Rembg, on the other hand, was readily available and easy to set up, but had shortcomings that resulted in erroneous background removal.[8] The purpose of using rembg was to extract the hand from the background and create a uniform image background. Ultimately, we determined that MediaPipe Hands was the most suitable solution, as it can generate a 3D hand model from a 2D image and proved to be highly effective for our purposes.[9]

MediaPipe Hands is part of MediaPipe, a library developed by Google, which is a versatile tool that provides various functionalities, such as face detection, object recognition, hand detection, and selfie segmentation. We specifically utilized MediaPipe Hands due to its ability to accurately perceive the shape and motion of hands, making it ideal for applications like sign language understanding and hand gesture control. The library is capable of high-fidelity hand and finger tracking, using machine learning to infer 21 3D landmarks of a hand from a single frame Zhang et al. (2020), which allows it to predict a hand skeleton. This is achieved using two models simultaneously: a palm detector that operates on a full input image to locate palms via an oriented hand bounding box, and a hand landmark model that operates on the cropped hand bounding box provided by the palm detector and returns high-fidelity landmarks. Table **??** demonstrates the performance of MediaPipe Hands with a hand confidence threshold of 0.1. Based on the results, we determined the performance to be satisfactory for our intended use.

For our data engineering problem, we have opted to utilize MediaPipe Hands as the foundation for our preprocessor. This preprocessor is tasked with finding the bounding box around a hand in an image and cropping the image accordingly. Our parameters for image processing include specifying

[7] https://github.com/cansik/yolo-hand-detection
[8] https://pypi.org/project/rembg/
[9] https://google.github.io/mediapipe/solutions/hands.html

Algorithm 1 Find the Bounding Box from the MediaPipe Hands Landmarks

$xHandCords \leftarrow \emptyset$
$yHandCords \leftarrow \emptyset$
for $landmark$ in $handLandmarks$ **do**
    $xHandCords \leftarrow xHandCords \cup landmark$.x
    $yHandCords \leftarrow yHandCords \cup landmark$.y
end for
$minX$ = min($xHandCords$)
$minY$ = min($yHandCords$)
$maxX$ = max($xHandCords$)
$maxY$ = max($yHandCords$)
return { $minX$,$minY$,$maxX$,$maxY$ }

the desired dimensions of the preprocessed image, cropping the image based on the hand position within the image using MediaPipe Hands, removing the background with the rembg library, and converting the images to one-channel greyscale images. The preprocessing steps consist of reading the image using cv2, cropping the image based on the bounding box found with MediaPipe Hands, resizing the image, adding padding if necessary, and converting the image to greyscale using cv2. The method for determining the bounding box to crop the image can be found in Algorithm 1. After conducting tests, we selected (300,300) as the desired dimensions which are then scaled down to (64,64) for our final model. Our preprocessor crops images using a hand detection confidence of 0.1, does not remove the background using rembg due to poor performance, and converts the images to greyscale to facilitate one-dimensional model input.

In the final step of our model data engineering, we decided to incorporate data augmentation techniques to improve the generalization of our model. We used simple forms of data augmentation, including random rotations, horizontal and vertical flips, and random affine transformations with shearing and scaling. We implemented these transformations using the standard transformation function provided by PyTorch Data Loader. Our data augmentation function is composed of various transformations, including ToTensor, RandomAffine (shear, scale), RandomRotation, RandomHorizontalFlip, and RandomVerticalFlip. This data augmentation will enable our model to be more robust to variations in the data and increase its performance during testing.

2.1 Experiment

Based on the research question, we want to investigate if removing the background (by cropping the images based on the bounding box generated from MediaPipe Hands outcome) during the image preprocessing phase benefits the image classification task. To test this, we will train the same model with and without preprocessed data and then test the performance on a test dataset, both preprocessed and not preprocessed. We hypothesize that the model trained on preprocessed data will perform better than the model trained on non-preprocessed data, as removing the background can reduce noise and improve the model's ability to learn the hand features. Our null hypothesis is that regardless of the preprocessing used, the blackbox model should perform equally on the validation and test dataset in terms of accuracy.

In order to test our research question, we will conduct an experiment where we train a deep learning model on both preprocessed and non-preprocessed data, and then compare their performance on a test dataset. We will use the same preprocessor parameters as before, but will crop images based on MediaPipe Hands with an interval of 0.001 for the validation and training data to obtain the best hand detection. The model parameters will be set with a dropout probability of 0.5, no batch normalization, 100 epochs of training, and a batch size of 64. We will use Adam optimizer with a learning rate of 0.001 and CrossEntropy as the criterion. To compare the models' performances, we will calculate the accuracy of the models on the train, validation, and test data after every 10 epochs of training. We will then compare the differences in accuracy between the two models. The experimental setup is illustrated in Figure 3.

The results of the experiment are as follows and presented in Figures 12, 13 and 4:

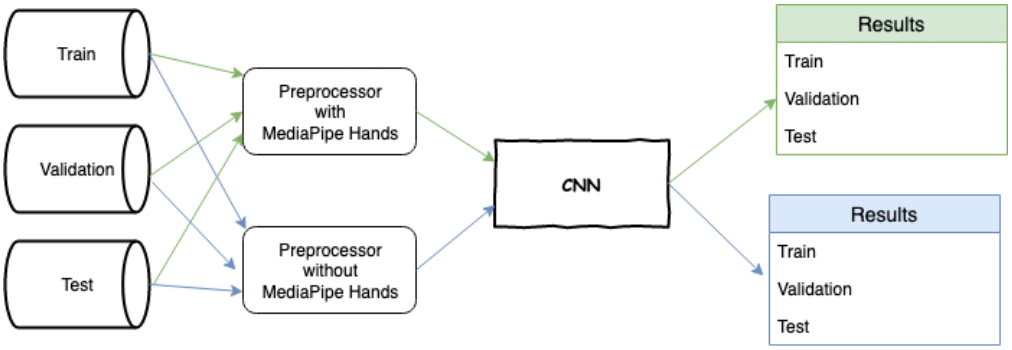


Figure 3: Data Engineering experimental setup

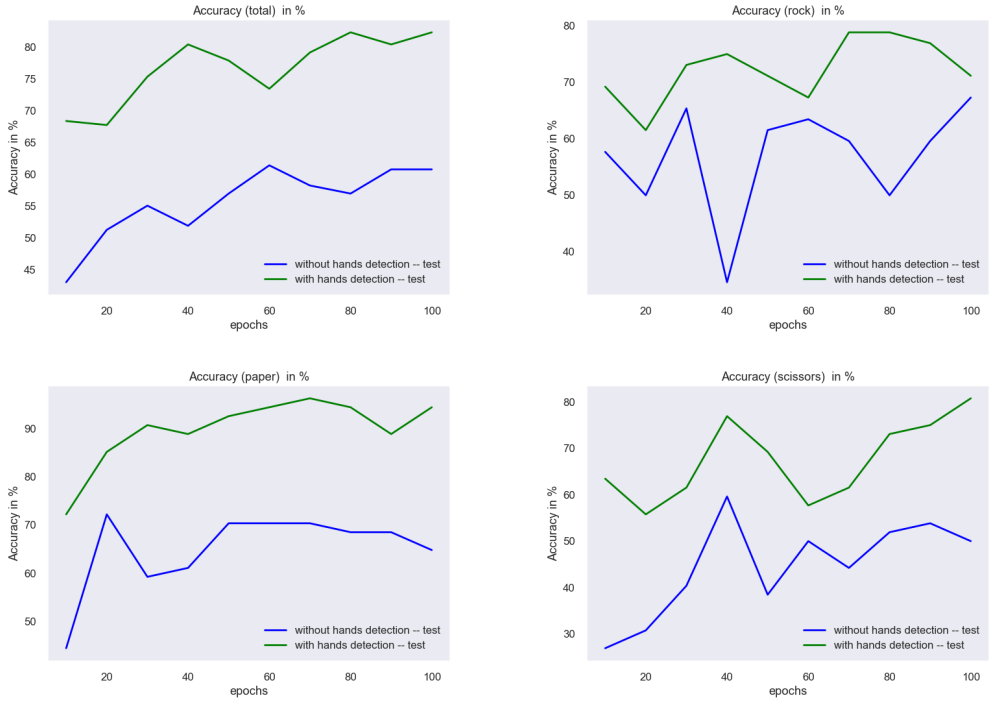


Figure 4: Model performance on test data

- Training Data: The model trained on cropped images outperformed the model trained on non-cropped images in terms of accuracy at less training steps. After 60 epochs, the total and category-wise performances of both models were comparable.
- Validation Data: The model trained on cropped images consistently outperformed the model trained on non-cropped images in each test model iteration. The total performance of the model trained on cropped images was always better, except for the "rock" category where the model trained on non-cropped images performed slightly better after 100 epochs of training.
- Testing Data: The model trained on cropped images consistently outperformed the model trained on non-cropped images in each test model iteration. The model trained on cropped images had much better performance at fewer training steps and remained better over all 100 training epochs. The overall performance of the model trained on cropped images was better.

Based on these results, it can be assumed that reducing the complexity of the dataset by removing the background (i.e. unimportant parts of the image) leads to better model performance.

2.2 Discussion

The results of our preprocessing pipeline indicate that it was effective in enhancing the quality of the input images for our hand gesture recognition model. However, since we did not apply any statistical tests to our findings, the results only suggest a tendency towards better performance rather than scientifically correct findings.

One issue that arose during the preprocessing stage was the possibility that specific preprocessing techniques could limit the overall performance of the model to the performance of the preprocessing itself. This suggests that more sophisticated preprocessing techniques might be required to improve the performance of the model beyond a certain point. On the other hand, even no preprocessing might be a valid approach but therefore the training data should be increased significantly.

Furthermore, the evaluation of the model on validation and test data revealed that the training data did not sufficiently model the testing and validation data. This might be due to the fact that the training data consisted of CGI and hands with a green background, which is not a common occurrence in real-life scenarios. As a result, it might have been more beneficial to generate much more custom data that better represented the real-life scenarios to improve the performance of the model.

2.3 Conclusion

In conclusion, while our preprocessing pipeline was effective in improving the quality of the input images, the results should be interpreted with caution due to the lack of statistical testing. Additionally, the limitation of the preprocessing techniques and the need for more representative training data suggest that another approach should be considered in order to improve the overall performance of the hand gesture recognition model.

3 Model Engineering

The performance metric used in the leaderboard was the accuracy on the test set. Thus, one of our main model engineering challenges was to prevent overfitting for the model to be generalizable to unseen images submitted by each team. The problem of overfitting in machine learning is a common one, especially in deep learning models such as convolutional neural networks. To address this issue, we tried two regularization techniques, namely Dropout and Batch Normalization. This section aims to investigate the effectiveness of Dropout and Batch Norm in preventing overfitting in our image classification task.

Our first model was a simple Convolutional Neural Network (CNN). We obtained insufficient results in terms of accuracy, especially on the scissors class. Our hypothesis is that our model was unable to recognize scissors because of the image input size, as we were using 32 x 32 pixels images. Indeed, more pixels are needed to recognize the outline of the fingers making a scissors shape as compared to rock or paper. Therefore, we decided to make input images 64 x 64 pixels instead. We also looked at the literature to improve our model and took inspiration from VGG16 (Simonyan and Zisserman (2014)).

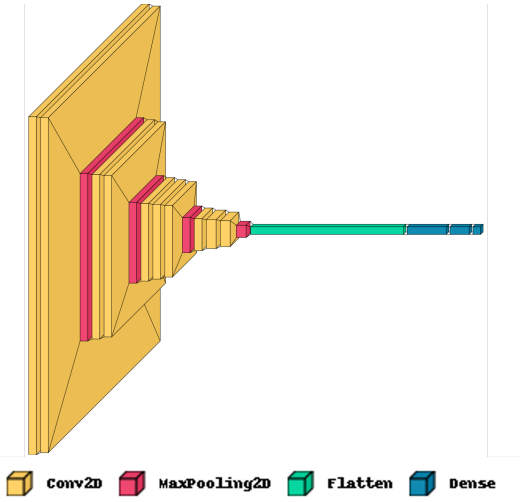


Figure 5: Visualization of our model

Overfitting is a common problem in machine learning where a model is trained to fit the training data too closely, to the point that it becomes overly specialized and loses its ability to generalize to new, unseen data. This undesirable behavior happens when a model is too complex or has too many parameters relative to the size of the training data. As a result, the model may perform well on the training data, but its performance on the test data is significantly worse. Therefore, overfitting can be diagnosed by splitting the dataset into train, validation and tests sets, and comparing loss and accuracy curves.

Dropout is a regularization technique used in machine learning to prevent overfitting. It works by randomly dropping out a certain percentage of neurons in a neural network during training, which helps to reduce co-adaptation between neurons and makes the network more robust. This means that the network is forced to learn more generalizable features and is less likely to overfit to the training data. Dropout can be applied to any layer in a neural network, including input, hidden, and output layers, and is typically implemented by setting a dropout rate that determines the probability of dropping out each neuron.

Batch Normalization is a technique used to improve training stability and speed and has also been shown to improve the generalization performance of neural networks. It works by normalizing the input data to each layer of the network so that it has zero mean and unit variance. This helps to alleviate the problem of covariate shift, which occurs when the distribution of input features changes as the network trains. By normalizing the input data, batch normalization helps to ensure that the network can learn more effectively and with fewer training iterations. Batch Norm can be applied to any layer in a neural network, including input, hidden, and output layers, and is typically implemented as a layer in the network architecture.

3.1 Experiment

To investigate the impact of Dropout and Batch Norm on overfitting, we trained the network during 100 epochs in four different configurations:

1. Without any regularization
2. With Dropout
3. With Batch Norm
4. With both Dropout and Batch Norm

We recorded the classification accuracy and loss on both the training and validation datasets, as well as the accuracy on the test set. The dropout rate was 0.5 and the batch size was 64.

(a) Training v Validation Loss

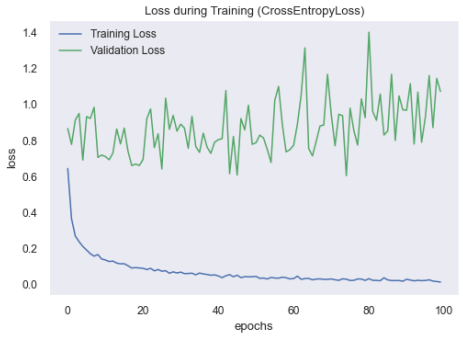
(b) Training v Validation Accuracy

Figure 6: Model performance without any regularization

(a) Training v Validation Loss

(b) Training v Validation Accuracy

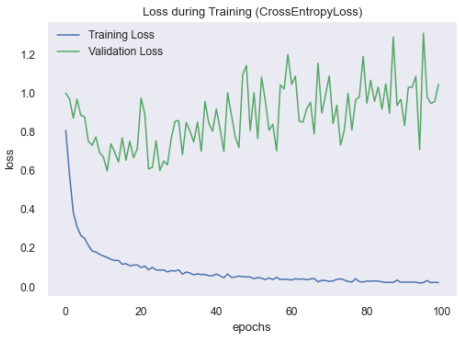
Figure 7: Model performance with Dropout



(a) Training v Validation Loss

(b) Training v Validation Accuracy

Figure 8: Model performance with Batch Norm



(a) Training v Validation Loss

(b) Training v Validation Accuracy

Figure 9: Model performance with both Dropout and Batch Norm

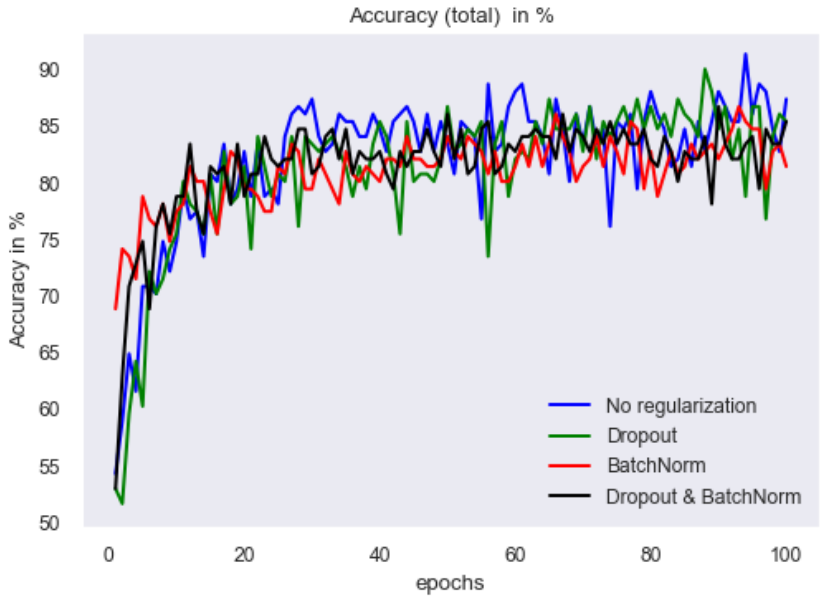


Figure 10: Comparison between the four accuracies on validation set

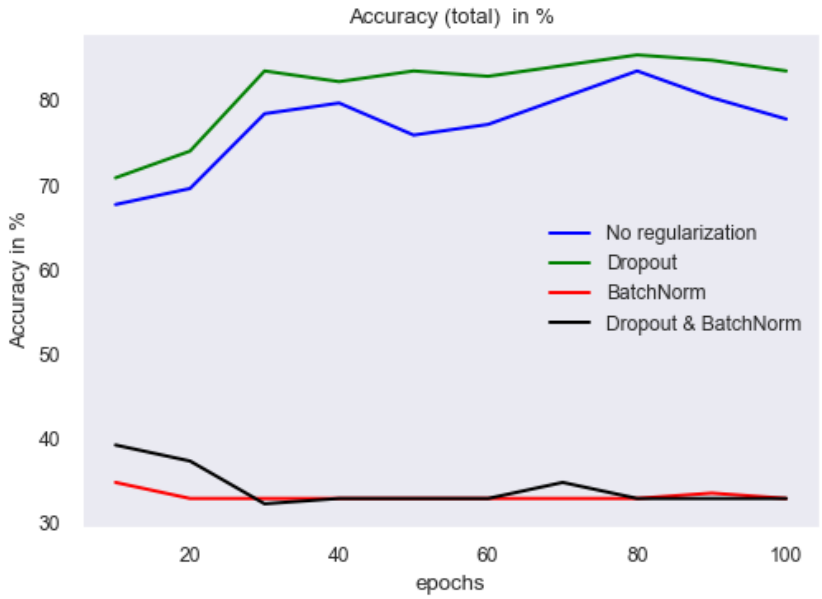


Figure 11: Comparison between the four accuracies on test set

All accuracies on the validation are similar, whatever the combination of regularization technique is.

However, accuracies on the test set are different from each other.

On the one hand, models incorporating Batch Norm achieve an accuracy level that is notably low, even failing to surpass random chance. This trend shows that Batch Normalization hinders generalization ability of the model in our case. We came up with several hypothesis to explain this unexpected behavior. During inference, Batch Norm resorts to parameters that were learnt during training based on the statistics of the training data. Therefore, if the discrepancy is substantial between the data distribution of the training set and the test set, then the Batch Normalization layer may not work as well during inference, leading to poor performance. In conclusion, the out-of-domain data contained in the test set explains why Batch Norm worsens model robustness, according to the test accuracy metric.

On the other hand, the model with Dropout achieves slightly better overall accuracy on the test set than the original model.

4 Model Evaluation

5 Final Conclusion

References

References

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL `https://arxiv.org/abs/1409.1556`.

S. Studer, T. B. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters, and K.-R. Müller. Towards crisp-ml (q): a machine learning process model with quality assurance methodology. *Machine learning and knowledge extraction*, 3(2):392–413, 2021.

F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann. Mediapipe hands: On-device real-time hand tracking, 2020. URL `https://arxiv.org/abs/2006.10214`.

Table 2: Percentage of detected hands in images

Origin	Rock	Paper	Scissors	Total
custom	95.2%	90.7%	96.2%	94.1%
cgi	89.0%	100%	100%	96.3%
hands	95.9%	99.6%	94.5%	96.6%
webcam	93,5%	96.2%	91.8%	93.8%
Total	92.8%	98.0%	95.6%	95.5%

Table 3: Who wrote which part

Part	Baptiste	Benedikt	Jonas
Abstract		√	
Introduction			√
Data Engineering			√
Model Engineering	√		
Model Evaluation		√	
Final Conclusion	√		

Declaration of Authorship

All final papers have to include the following 'Declaration of Authorship':

Declaration of Authorship

Ich erkläre hiermit gemäß § 9 Abs. 12 APO, dass ich die vorstehende Projektarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass die digitale Fassung der gedruckten Ausfertigung der Projektarbeit ausnahmslos in Inhalt und Wortlaut entspricht und zur Kenntnis genommen wurde, dass diese digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann.

Bamberg, February 23, 2023

(Place, Date) (Signature)

Bamberg, February 23, 2023

(Place, Date) (Signature)

Bamberg, February 23, 2023

(Place, Date) (Signature)

A Appendix

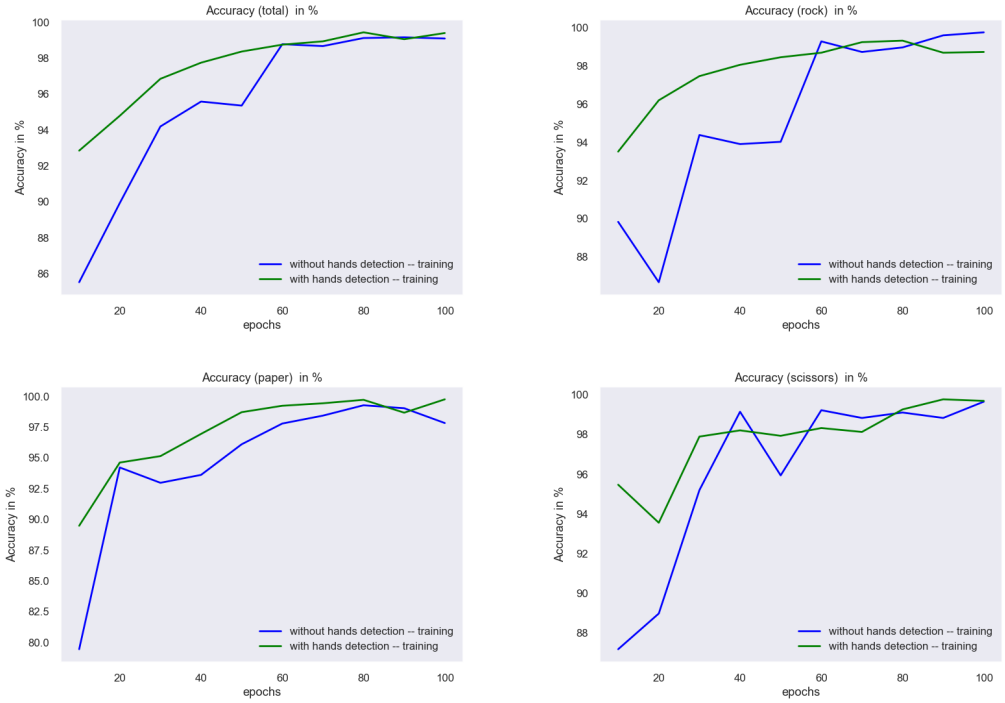


Figure 12: Model performance on training data

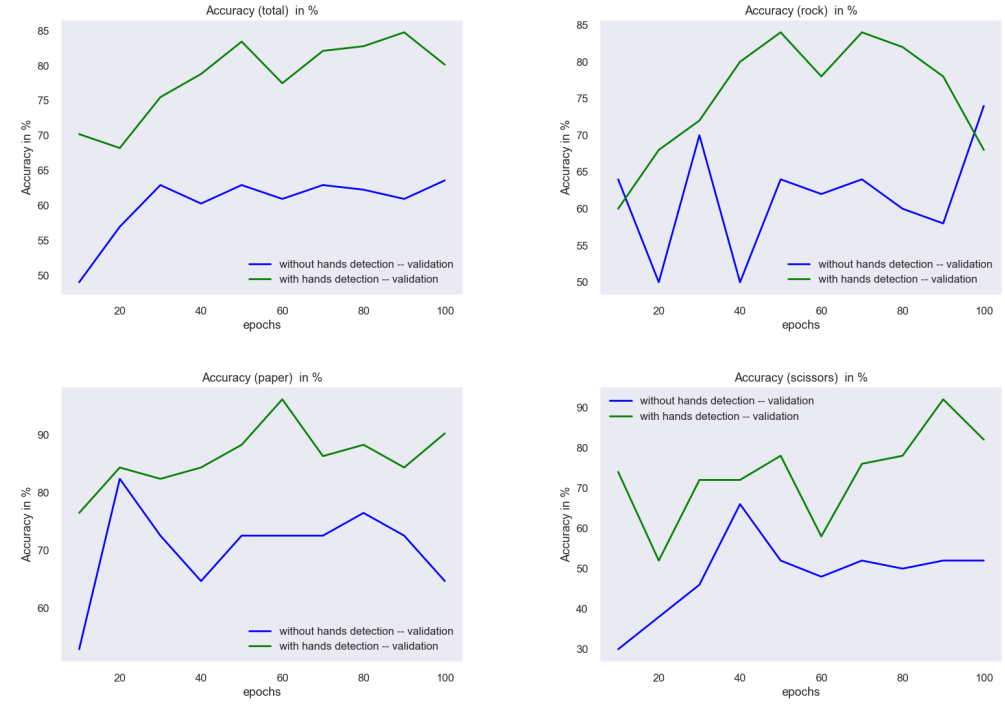


Figure 13: Model performance on validation data