# Your Awesome Title

## xAI-Proj-M: Master Project Explainable Machine Learning

**Benedikt M. Marsiske**[*]
Otto-Friedrich University of Bamberg
96049 Bamberg, Germany
benedikt-markus.marsiske@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

During this project we applied the CRISP-ML methodology to the problem of classifying rock, paper, or scissors from an image that includes exactly one hand. The Deep Learning Life Cycle of CRISP-ML is explored, focusing on Data Engineering, Model Engineering, and Model Evaluation. This report sets out to answer three research questions: (1) Does removing the background during image preprocessing improve classification accuracy? (2) Do dropout layers prevent overfitting and what is the ideal position for these layers in the model? (3) How does the model perform on distorted data and does the use of distorted test data lead to worse model performance compared to the same test data without distortion? The report presents the findings of the experiments, including the implications, limitations, and areas for future improvements. The findings suggest that removing the background leads to a better image classification, while the optimal architecture included dropout layers but no batch normalization. The model that was trained during this project performed poorly on distorted images, but the incorporation of noisy training data was identified as a possible solution for this problem.

## 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.

[*] Degree: M.Sc. CitH, matriculation #: 2045635

[†] Degree: Engineering Master, matriculation #: 2117568

[‡] Degree: M.Sc. AI, matriculation #: 1867301

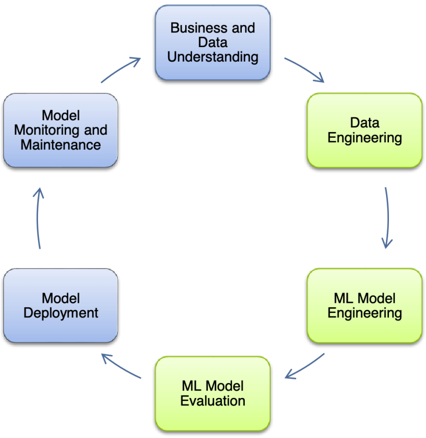


Figure 1: The CRISP-ML: Deep Learning Life Cycle as presented in the introductory presentation by Ines Rieger.

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.

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 13.

## 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

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 |

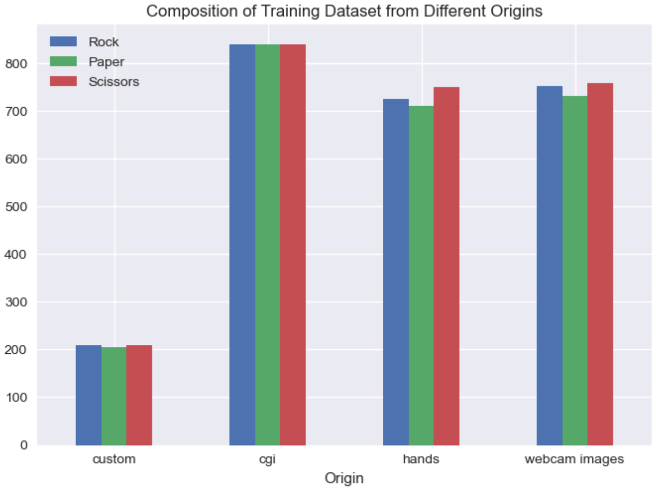


Figure 2: Individual dataset distribution showing the number of images in each dataset

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 1 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 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]

[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`

[7] https://github.com/cansik/yolo-hand-detection

[8] https://pypi.org/project/rembg/

[9] https://google.github.io/mediapipe/solutions/hands.html

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 7 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.

**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 $\{$ $minX$,$minY$,$maxX$,$maxY$ $\}$
```

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 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

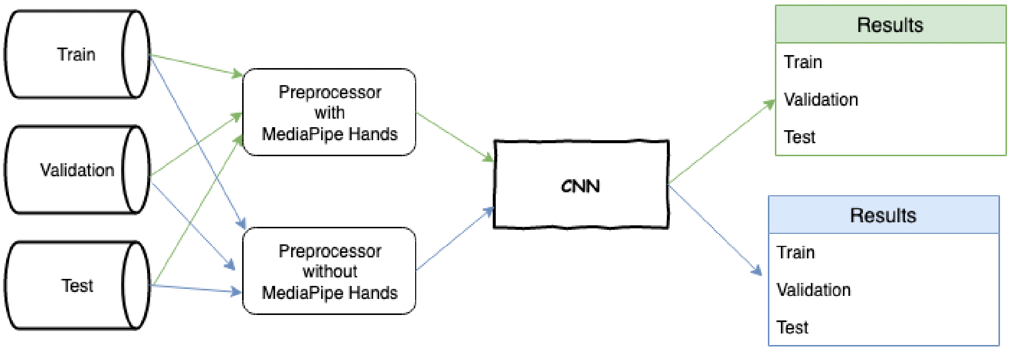


Figure 3: Data Engineering experimental setup

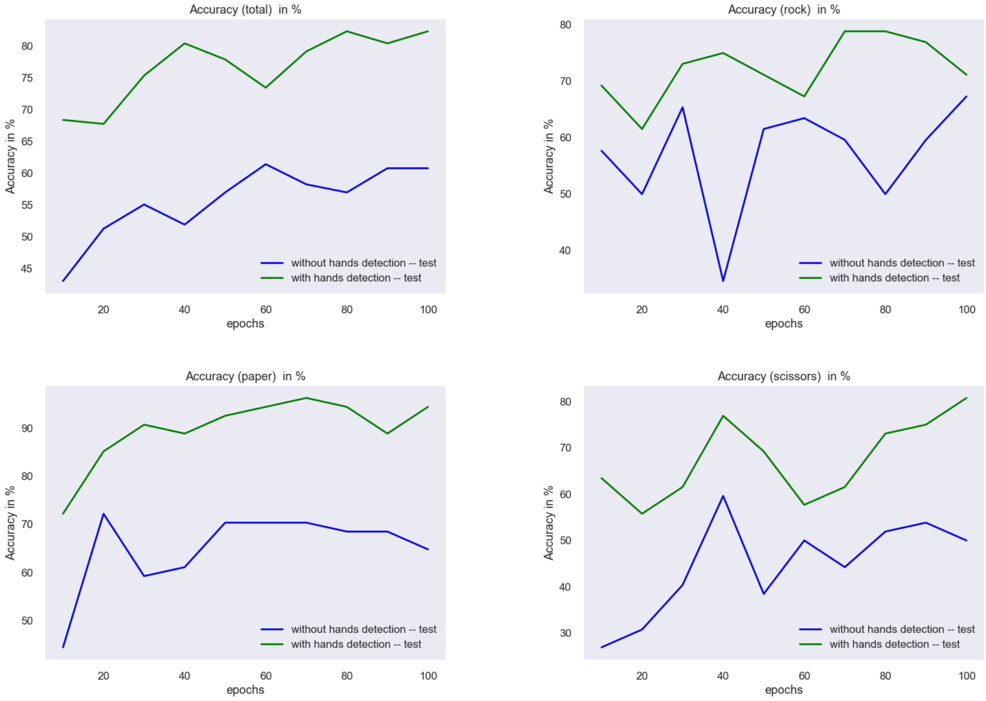


Figure 4: Model performance on test data

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 8, 9 and 4:

- 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)).

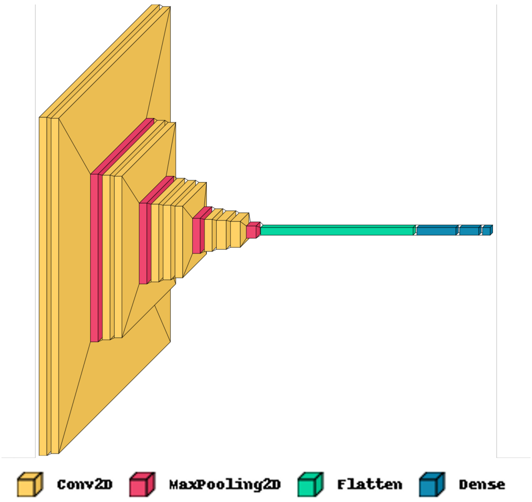


Figure 5: Visualization of our model

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

### 3.2 Introduction

The purpose of this experiment is to investigate the impact of Dropout and Batch Norm on overfitting. As explained, overfitting is a common problem in deep learning models, where the model performs well on the training dataset but poorly on the validation and test sets. Dropout and Batch Norm are two popular regularization techniques used to prevent overfitting. Dropout randomly drops out some neurons during training, while Batch Norm normalizes the output of each layer to ensure that the distribution of the input to each layer is consistent. In this experiment, we aim to compare the performance of a neural network trained with and without these regularization techniques.

### 3.3 Method

We trained our neural network for 100 epochs in four different configurations: without any regularization, with Dropout, with Batch Norm, and 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 set to 0.5, and the batch size was set to 64. Loss and accuracy curves on training and validation sets recorded during training are available here: Figures 10, 11, 12, and 13.

### 3.4 Results

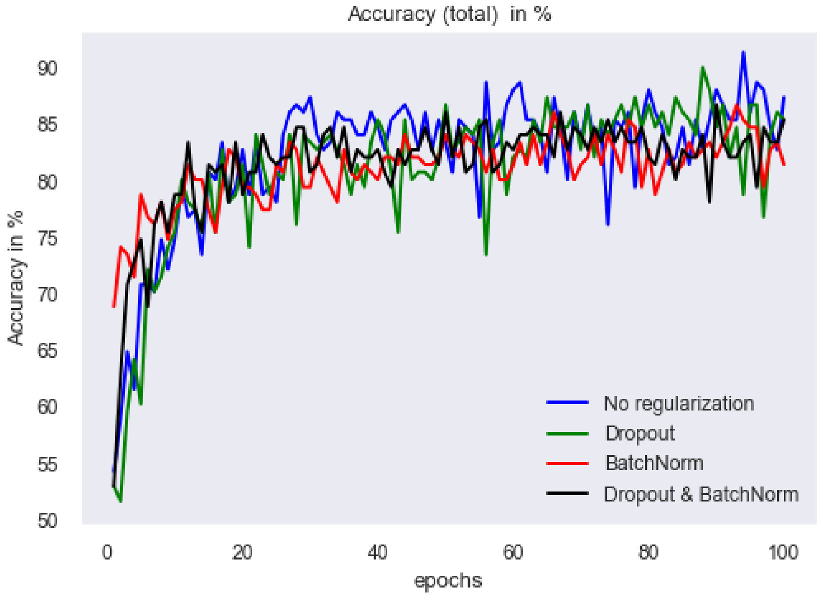


Figure 6: Comparison between the four accuracies on validation set

Figure 6 shows that all accuracies on the validation set are similar, regardless of the regularization technique used. However, Figure 7 reveals that the accuracies on the test set are different from each other. The model with Dropout achieves slightly better overall accuracy on the test set than the original model, indicating that Dropout helps to improve the model's generalization ability.

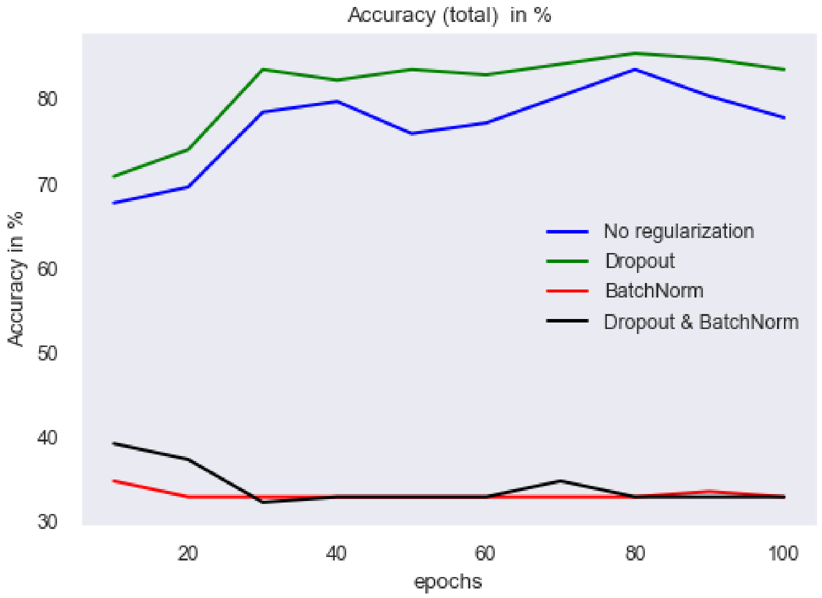


Figure 7: Comparison between the four accuracies on test set

On the other 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 the generalization ability of the model in our case. We came up with several hypotheses 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.

### 3.5 Discussion

The results of this experiment suggest that Dropout is an effective regularization technique that can help to improve the generalization ability of neural networks. On the other hand, Batch Norm may not always be beneficial, especially when the data distribution of the test set differs substantially from that of the training set.

Note well that the results of this experiment may not generalize to other datasets or models. Further experimentation is necessary to confirm these findings and to explore the effectiveness of other regularization techniques.

### 3.6 Conclusion

In this experiment, we compared the performance of a neural network trained with and without Dropout and Batch Norm. The results show that Dropout helps to improve the generalization ability of the model, while Batch Norm may not always be beneficial. The findings of this experiment can help to inform the choice of regularization techniques in deep learning models. Further research is needed to confirm these results and to explore the effectiveness of other regularization techniques.

# 4 Model Evaluation

## 4.1 Basic Model Evaluation

In this section, we will evaluate the previously presented model based on several criteria. For our evaluation, we first look at the performance of the model on the testset that was provided as part of this project. This dataset consists of 158 images, with 54 instances of paper, 52 instances of rock, and 52 instances of scissors.

The evaluation measures indicate that the model performs reasonably well on the classification task, with an overall classification accuracy of 83.54%. However, the evaluation also reveals some class imbalances, with the precision, recall, and F1-score varying across the three classes. For example, the model has a high precision for the rock class (0.911), indicating that when the model predicts the rock class, it is usually correct. However, the recall for the rock class is lower (0.788), indicating that the model misses some instances of the rock class.

Similarly, the model has a high recall for the paper class (0.963), indicating that it correctly identifies most instances of the paper class. However, here the precision for the paper class is lower at 0.754, indicating that the model tends to sometimes predict the paper class incorrectly. To thoroughly evaluate the performance of the model, we will use precision, recall, F1-score, and accuracy. Precision measures the proportion of true positives among the instances predicted as positive. Recall measures the proportion of true positives among the instances that are actually positive and the F1-score is the harmonic mean of precision and recall.

To evaluate the accuracy of individual classes we use the One-vs.-rest method, which involves treating the class of interest as the positive class and the remaining classes as the negative class. The model is then evaluated based on its ability to correctly classify instances belonging to the positive class, while ignoring instances belonging to the negative classes. The reported overall classification accuracy of 83.54%, however, measures the proportion of correctly classified instances among all instances.

To calculate the respective performance metrics, the following equations (Grandini et al. (2020)) were used:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

where TP is the number of true positives and FP is the number of false positives.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

where TP is the number of true positives and FN is the number of false negatives.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

where Precision and Recall are as defined above.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

We report the evaluation measures for each individual class in Table 2, which includes precision, recall, F1-score, and accuracy.

The overall accuracy of the model is lower than the accuracies for the individual classes, since here only the cases in which the actual and predicted classes match exactly are considered correct. The One-vs.-rest method used for the individual classes does not account for errors that don’t involve the current positive class.

Table 2: Performance metrics of each individual class

| Class | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| Rock | 0.911 | 0.788 | 0.845 | 0.905 |
| Paper | 0.757 | 0.981 | 0.855 | 0.886 |
| Scissors | 0.884 | 0.731 | 0.800 | 0.880 |

To further evaluate, we can look at the confusion matrix of our model's performance on the testset:

Table 3: Confusion matrix of our model on the test data

| | | **Prediction** | | |
|---|---|---|---|---|
| | | **rock** | **paper** | **scissor** |
| **Actual** | **rock** | 41 | 7 | 4 |
| | **paper** | 0 | 53 | 1 |
| | **scissor** | 4 | 10 | 38 |

The confusion matrix (Table 3) shows that the model correctly classified 41 instances of the Rock class, 52 instances of the Paper class, and 38 instances of the Scissors class. The most common error made by the model is to mistake scissors for paper, which occurs in 6.3% of the cases. This error is understandable because the paper and scissors signs are closely related, and a similar hand gesture can be used to represent both signs. However, this error occurred only in 10 instances, which is a relatively small number of misclassifications. One should note that this result should be interpreted with caution due to the limited size (150 images) of the test set. For a more thorough evaluation a larger and more diverse test set is needed to fully evaluate the model's performance and identify more potential imbalances.

Despite these limitations, the evaluation suggests that the model is reasonably effective at predicting the correct hand sign. In the next section, we describe an experiment that tests the performance of the model on a noisy test set to further evaluate our model and assess its robustness.

## 4.2 Experiment - Evaluating Robustness On Distorted Images

### 4.2.1 Introduction

Robustness is a critical aspect of any machine learning model, particularly for image classification tasks. It can be defined as the ability of a model to maintain its performance under naturally-induced image corruptions or alterations (Drenkow et al. (2021)).

Convolutional neural networks (CNNs) are commonly used for image classification tasks, but they are susceptible to corruption (Dodge and Karam (2017)). These corruptions can often occur in real-world scenarios due to factors such as a bad internet connection, bad camera quality, or other external factors that cannot be easily controlled. In contrast, humans can easily recognize mildly transformed images caused by noise or blur (Dodge and Karam (2017)). Therefore, to build a model that can rival human performance, a robust model is required.

A robust model is not only important for achieving high accuracy, but it also ensures the trustworthiness of the model. There is a general scepticism towards artificial intelligence (AI) from the general public (Juravle et al. (2020)) and a robustly performing model is an integral part of increasing the trust in AI systems.

To test whether our model is robust and answer our third research question, we conducted an experiment that evaluated the performance of the model on a noisy test set. Specifically, we used two different image modification techniques - Gaussian distortion and random noise - to simulate naturally-induced image corruptions. We compared the performance of our model on our test set with the performance on these noisy datasets. Additionally, we briefly looked into techniques to increase the robustness of our model.

In this experiment, we aimed to determine the impact of image corruptions on the performance of our model and whether it could maintain its performance in real-world scenarios. The results of this experiment will provide valuable insights into the robustness of our model and the potential need for additional optimization to achieve higher performance in real-world settings.

### 4.2.2 Method

To test the robustness of our rock, paper, scissor image classifier model and answer our research question, we used two different image transformations: Gaussian Noise and Random Distortion, which we implemented using the cv2 Python library.

Gaussian Distortion is a common image transformation technique that applies a Gaussian blur to an image. The amount of blur applied is determined by a standard deviation (SD) parameter, which controls the spread of the normal distribution that is used to generate the blur kernel. We applied a SD of 25 on our preprocessed grayscale images, which resulted in a visible but not overly distorted version of the original image.

Random Noise is a different type of transformation that involves randomly removing pixels from an image. To implement this transformation, we assigned each pixel a set chance to be removed. We chose a value of 25%, which meant that 25% of the pixels of each image got randomly removed and replaced by a black pixel.

By using these two transformations on our preprocessed test dataset, we created two additional datasets that included the same images as the original test set, but with the added distortions. This way we can compare the performance of our model on the undistorted dataset with the performance on the two distorted datasets.

To further test whether we could improve the robustness of our model, we also trained the model with noisy data. Specifically, we used the same architecture as described in section 3 and used the same training set as before but this time it consisted of 70% non-distorted images and 30% distorted images with the Random Noise Transformation. This new model allowed us to briefly evaluate the effect of incorporating noisy images to our training data.

Due to issues during the training process and time restraints of the project we stopped the training and used the model produced after only 20 training epochs. This model was the best performing one out of the models trained on noisy data and still provided insight into the effect of using such techniques.

### 4.2.3 Results

Comparing the results presented in section 4.1 of the model's performance on the undistorted dataset with the performance on the distorted datasets, we notice a drastic decrease in the performance.

When tested on the dataset with random distortion, the model's accuracy decreased to 0.43. The most noticeable decrease in performance was observed in the scissors class, where the model had difficulty identifying any images as scissors (precision: 0.417, recall: 0.096, F1-score: 0.156). A more detailed overview of the classification distribution can be found in Table 5.

When tested on the dataset with Gaussian distortion, the model's total accuracy was 0.49. Here we could observe that the model strongly preferred the paper class and classified most of the images as paper, leading to a recall of 0.907 with a precision of only 0.412. The confusion matrix for this testset can be found in Table 6.

The evaluation of the model trained on noisy data showed that it performed worse on the undistorted testset (total accuracy: 0.589). However, the model was robust against both distorted datasets, with the same accuracy or higher accuracy than on the undistorted testset.

All the total accuracies are presented in Table 4 below. The exact performance metrics of our standard model on the undistorted testset are shown in Table 2 while the remaining metrics on the different datasets can be found in the appendix.

Table 4: Accuracy of our models on different test data

| Data | Standard Model | Model Trained On Noisy Data |
|---|---|---|
| Undistorted Testset | 0.835 | 0.589 |
| Random Distortion | 0.430 | 0.652 |
| Gaussian Noise | 0.494 | 0.595 |

Table 5: Confusion matrix of our model on the randomly distorted testset

| | | **Prediction** | | |
|---|---|---|---|---|
| | | **rock** | **paper** | **scissor** |
| **Actual** | **rock** | 34 | 15 | 3 |
| | **paper** | 21 | 29 | 4 |
| | **scissor** | 23 | 24 | 5 |

Table 6: Confusion matrix of our model on the testset with Gaussian Noise

| | | **Prediction** | | |
|---|---|---|---|---|
| | | **rock** | **paper** | **scissor** |
| **Actual** | **rock** | 18 | 31 | 3 |
| | **paper** | 5 | 49 | 0 |
| | **scissor** | 2 | 39 | 11 |

### 4.2.4 Discussion

The results of our experiment showed that the model's performance varied considerably across different datasets, highlighting the importance of robustness to noise and other types of distortion.

We observed that the model trained on undistorted data did not perform well on noisy data, indicating that the model is not robust against noisy data. However, we found that training the model on noisy data improved its robustness to noise. This suggests that training with more diverse data, including noisy data, can help improve the model's robustness.

While the model trained on distorted data was more robust against both types of distortion, we have to note that it did not perform well on any of the test sets. When incorporating noisy data in the training, there are many factors to consider that are beyond the scope of this project. Our attempt at training a model on noisy data was more of a proof of concept, showing that it is possible to increase the model's robustness even though the overall performance was poor. Further investigation is necessary to determine the optimal approach for incorporating noisy data in the training process to achieve both robustness and high performance.

Although we only used two different types of distortion in this study, our results suggest that our model's robustness against one type of distortion can transfer to other types of distortion. Additionally, we acknowledge that our distortion may not be entirely natural, but we attempted to produce noise that is close to naturally occurring noise.

## 5 Final Conclusion

In conclusion, this research project applied the CRISP-ML methodology to the problem of classifying rock, paper, or scissors from images that contain exactly one hand. The project explored the three main steps in the process: Data Engineering, Model Engineering, and Model Evaluation. The findings suggest that preprocessing techniques can enhance the quality of input images, although sophisticated techniques might be required to improve performance beyond a certain point. Furthermore, the limitation of the preprocessing techniques and the need for more representative training data suggest that another approach should be considered to improve the overall performance of the hand gesture recognition model. The results of the Model Engineering experiment suggest that Dropout is an effective regularization technique that can help improve the generalization ability of neural networks, while Batch Norm may not always be beneficial. The Model Evaluation experiment showed that training with more diverse data, including noisy data, can help improve the model's robustness. However, further investigation is necessary to determine the optimal approach for incorporating noisy data in the training process to achieve both robustness and high performance. Overall, this project highlights the importance of applying the CRISP-ML methodology to ensure quality assurance in developing machine learning applications.

## References

## References

S. Dodge and L. Karam. A study and comparison of human and deep learning recognition performance under visual distortions. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, 2017. doi: 10.1109/ICCCN.2017.8038465.

N. Drenkow, N. Sani, I. Shpitser, and M. Unberath. Robustness in deep learning for computer vision: Mind the gap? *CoRR*, abs/2112.00639, 2021. URL `https://arxiv.org/abs/2112.00639`.

M. Grandini, E. Bagli, and G. Visani. Metrics for multi-class classification: an overview, 2020. URL `https://arxiv.org/abs/2008.05756`.

G. Juravle, A. Boudouraki, M. Terziyska, and C. Rezlescu. Chapter 14 - trust in artificial intelligence for medical diagnoses. In B. L. Parkin, editor, *Real-World Applications in Cognitive Neuroscience*, volume 253 of *Progress in Brain Research*, pages 263–282. Elsevier, 2020. doi: https://doi.org/10.1016/bs.pbr.2020.06.006. URL `https://www.sciencedirect.com/science/article/pii/S0079612320300819`.

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`.

## 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 24, 2023

(Place, Date) (Signature)

Bamberg, February 24, 2023

(Place, Date) (Signature)

Bamberg, February 24, 2023

(Place, Date) (Signature)

# A Appendix

Table 7: 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% |

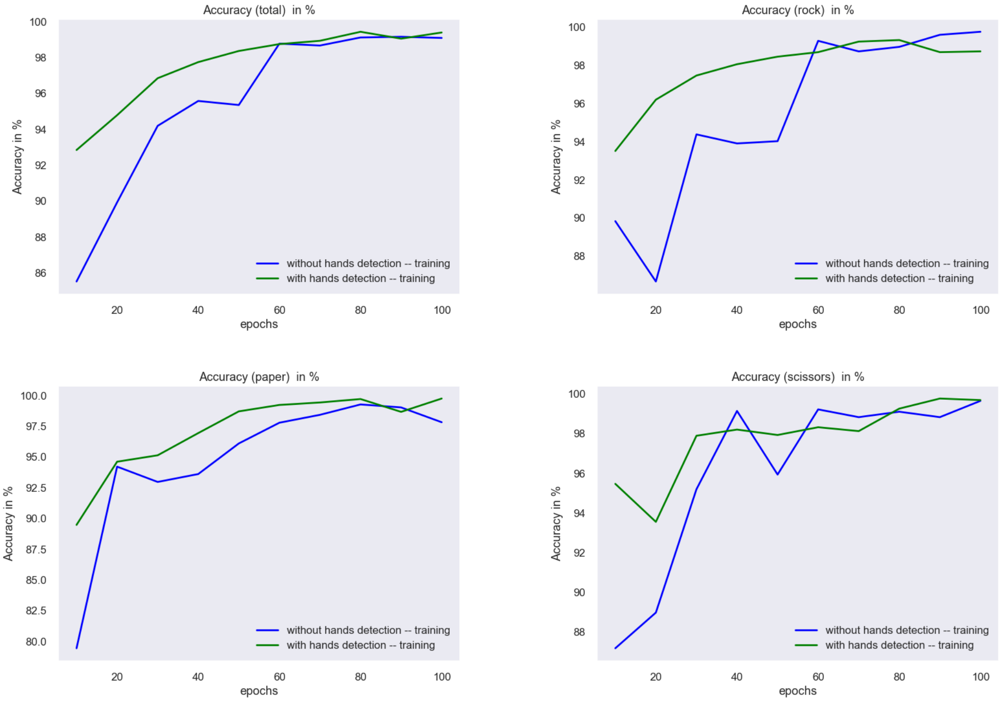


Figure 8: Model performance on training data

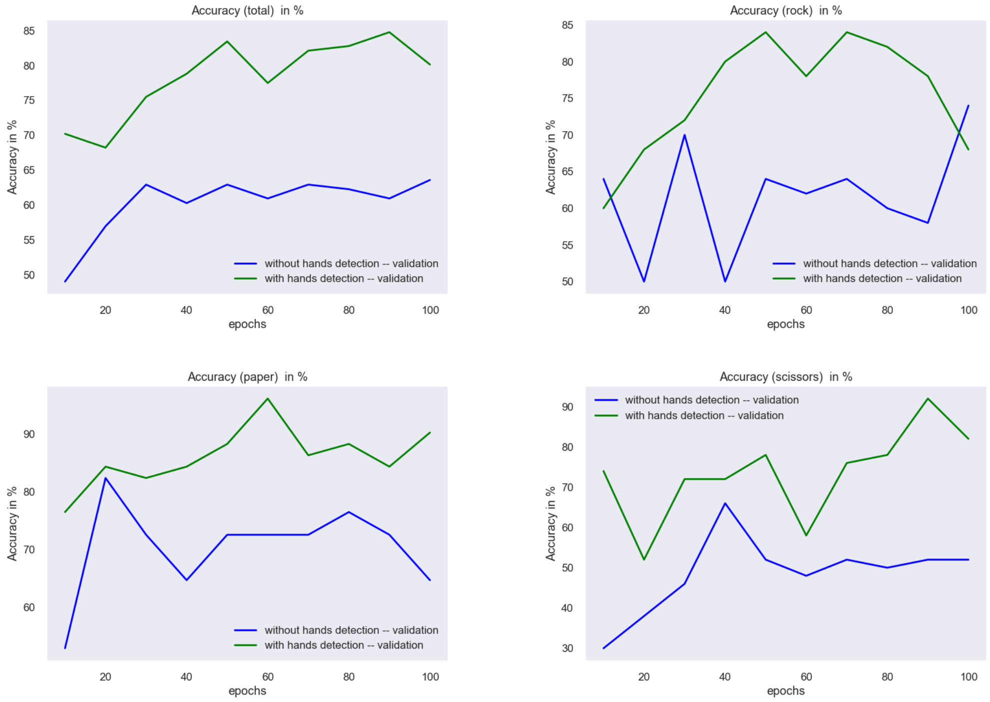
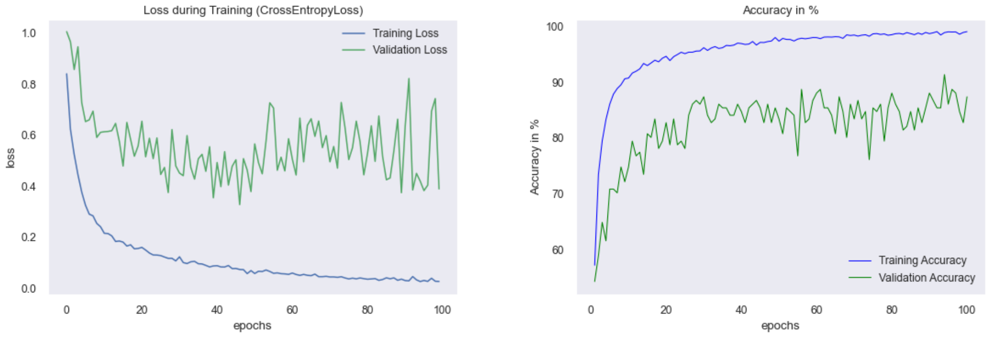


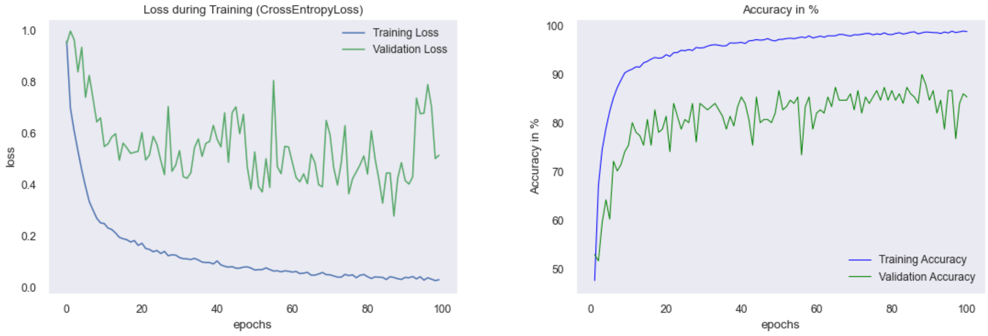
Figure 9: Model performance on validation data



(a) Training v Validation Loss

(b) Training v Validation Accuracy

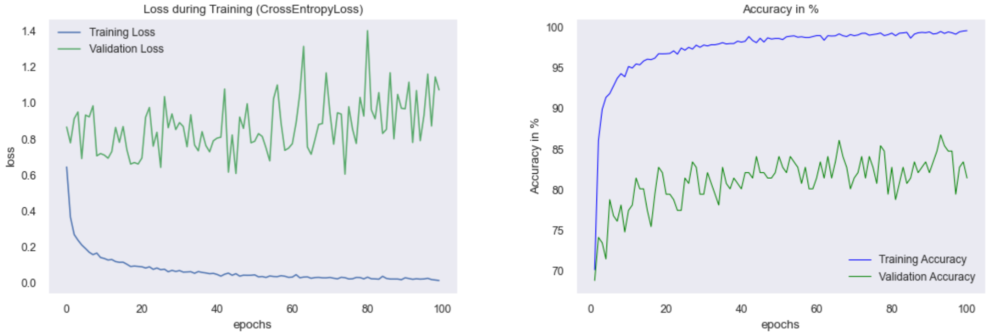
Figure 10: Model performance without any regularization



(a) Training v Validation Loss

(b) Training v Validation Accuracy

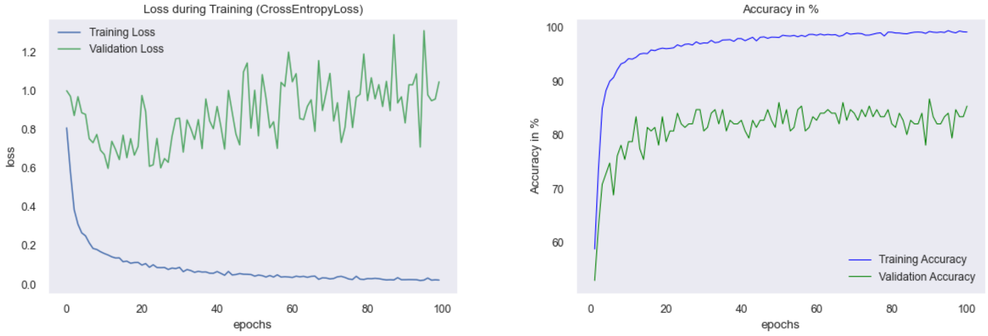
Figure 11: Model performance with Dropout



(a) Training v Validation Loss

(b) Training v Validation Accuracy

Figure 12: Model performance with Batch Norm



(a) Training v Validation Loss

(b) Training v Validation Accuracy

Figure 13: Model performance with both Dropout and Batch Norm

Table 8: Performance metrics for our standard model on the randomly distorted testset

| Class | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| Rock | 0.436 | 0.654 | 0.523 | 0.608 |
| Paper | 0.426 | 0.537 | 0.475 | 0.595 |
| Scissors | 0.417 | 0.096 | 0.156 | 0.658 |

Table 9: Performance metrics for our standard model on the testset distorted with Gaussian noise

| Class | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| Rock | 0.720 | 0.346 | 0.468 | 0.741 |
| Paper | 0.412 | 0.907 | 0.566 | 0.525 |
| Scissors | 0.786 | 0.212 | 0.333 | 0.722 |

Table 10: Performance metrics for the model trained on 30% noisy data on the undistorted testset

| Class | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| Rock | 0.542 | 0.865 | 0.667 | 0.715 |
| Paper | 0.556 | 0.370 | 0.444 | 0.684 |
| Scissors | 0.718 | 0.538 | 0.615 | 0.778 |

Table 11: Performance metrics for the model trained on 30% noisy data on the randomly distorted testset

| Class | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| Rock | 0.605 | 0.885 | 0.719 | 0.772 |
| Paper | 0.641 | 0.463 | 0.538 | 0.728 |
| Scissors | 0.744 | 0.615 | 0.674 | 0.804 |

Table 12: Performance metrics for the model trained on 30% noisy data on the testset distorted with Gaussian noise

| Class | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| Rock | 0.592 | 0.808 | 0.683 | 0.753 |
| Paper | 0.509 | 0.519 | 0.514 | 0.665 |
| Scissors | 0.750 | 0.462 | 0.571 | 0.772 |

Table 13: Who wrote which part

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