Data Engineering – Background Removal

Explainable Machine Learning - Deep Learning Life Cycle

Jonas Amling Baptiste Patrice Francis Bony Benedikt Markus Marsiske

February 2, 2023

University of Bamberg

Table of contents

Research Question

Research Question and Introduction

Our main Data Engineering Problems:

- Combining different datasets
- Different hand positions in different datasets
- Hands in different contexts in each dataset

Research Question and Introduction

Our main Data Engineering Problems:

- Combining different datasets
- Different hand positions in different datasets
- Hands in different contexts in each dataset
- $\Longrightarrow$ Reducing complexity of datasets is key

Research Question and Introduction

Our main Data Engineering Problems:

- Combining different datasets
- Different hand positions in different datasets
- Hands in different contexts in each dataset
- $\Longrightarrow$ Reducing complexity of datasets is key

Research Question: **Does removing the background during the image preprocessing phase benefit the image classification task at hand?**

Data Engeneering Process

Test, Train and Validation Datasts

Combining Datasets of different sources:

Training Data data combined from different datasets

custom	Self produced images
cgi	Computer-generated images [1]
webcam	Existing Dataset from Kaggle (hands with bodies) [2]
hands	Existing Dataset from Kaggle (only hands from top) [3]

[1] https://www.tensorflow.org/datasets/catalog/rock_paper_scissors
[2] https://www.kaggle.com/datasets/drgfreeman/rockpaperscissors
[3] https://www.kaggle.com/datasets/glushko/rock-paper-scissors-dataset

Test, Train and Validation Datasts

Combining Datasets of different sources:

Training Data data combined from different datasets

custom	Self produced images
cgi	Computer-generated images [1]
webcam	Existing Dataset from Kaggle (hands with bodies) [2]
hands	Existing Dataset from Kaggle (only hands from top) [3]

Validation Data Provided by the project (created by individual groups)

[1] https://www.tensorflow.org/datasets/catalog/rock_paper_scissors
[2] https://www.kaggle.com/datasets/drgfreeman/rockpaperscissors
[3] https://www.kaggle.com/datasets/glushko/rock-paper-scissors-dataset

Test, Train and Validation Datasts

Combining Datasets of different sources:

Training Data data combined from different datasets

custom Self produced images

cgi Computer-generated images [1]

webcam Existing Dataset from Kaggle (hands with bodies) [2]

hands Existing Dataset from Kaggle (only hands from top) [3]

Validation Data Provided by the project (created by individual groups)

Testing Data Provided by the project (created by individual groups)

[1] https://www.tensorflow.org/datasets/catalog/rock_paper_scissors
[2] https://www.kaggle.com/datasets/drgfreeman/rockpaperscissors
[3] https://www.kaggle.com/datasets/glushko/rock-paper-scissors-dataset

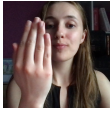


Figure 1: Distribution of individual Datasets

Existing libraries I

Searching the WWW we found some interesting libraries:

- YOLO-Hand-Detection: find hand position in an image [4]
  - \+ works on real life images, open source
  - \- not included in Python Package Index

[4] https://github.com/cansik/yolo-hand-detection
[5] https://pypi.org/project/rembg/

Existing libraries I

Searching the WWW we found some interesting libraries:

- YOLO-Hand-Detection: find hand position in an image [4]
  - \+ works on real life images, open source
  - \- not included in Python Package Index
- rembg: model that automatically removes image background [5]
  - \+ comes as library in Python Package Index
  - \- not works in all cases, has some strange edge cases

[4] https://github.com/cansik/yolo-hand-detection
[5] https://pypi.org/project/rembg/

Existing libraries II

Searching the WWW we found some interesting libraries:

- MediaPipe Hands: generates a 3d hand model from a 2d image [6] [1]
  - \+ works quite well and comes as library in Python Package Index
  - \- developed by google

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

The Preprocessor

Parameters for Image Processing:

- desired dimensions of preprocessed image
- crop image, based on the hand position within the image (Mediapipe Hands)
- remove background (rembg)
- greyscale: convert images to one-channel greyscale images

The Preprocessor

Parameters for Image Processing:

- desired dimensions of preprocessed image
- crop image, based on the hand position within the image (Mediapipe Hands)
- remove background (rembg)
- greyscale: convert images to one-channel greyscale images

Preprocessing steps:

1. read image using cv2
2. crop image based on bounding-box found with MediaPipe
3. remove left over background using rembg library
4. resize image and add padding if necessary
5. use cv2 to convert images to greyscale

Preprocessing Examples

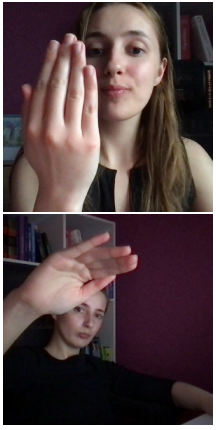


Figure 2: original

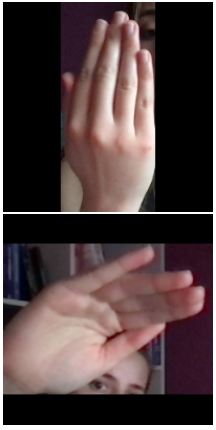


Figure 3: cropped

Figure 4: background removal

Parameter Selection for Preprocessor

Chosen parameters:

dimensions (300,300), and again scaled down for model to (64,64)

crop images True with a hand detection confidence of 0.1

remove background False, since rembg did perform very poorly

greyscale True

Media Pipe Hands Performance on Test Dataset

Evaluation the performance of hand detection with Mediapipe Hands with a confidence of 0.1

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

Table 1: Total number of images per origin

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 2: Percentage of detected hands in images

Experiment

Same Model, Same Data, Different Processing, Same Result?

Here the basic Idea is to run the exactly same training simply with different preprocessed Datasets

> **H0** : **Reagardless of the preprocessing used, the (blackbox) model should perform equally on the accuracy on the validation and test dataset in terms of accuracy**

- Preprocessor parameters are set as before, only difference is the use of cropping images based on Mediapipe Hands
- Model parameters: dropout probability: 0.5, no batch normalization, 100 epoches of training and a batch size of 64, Adam optimizer with learning-rate of 0.001 and CrossEntropy as criterion
- Compare the model performance on Train, Validation and Test Data after each 10 epoches of training

Schematic of Experiment Setup

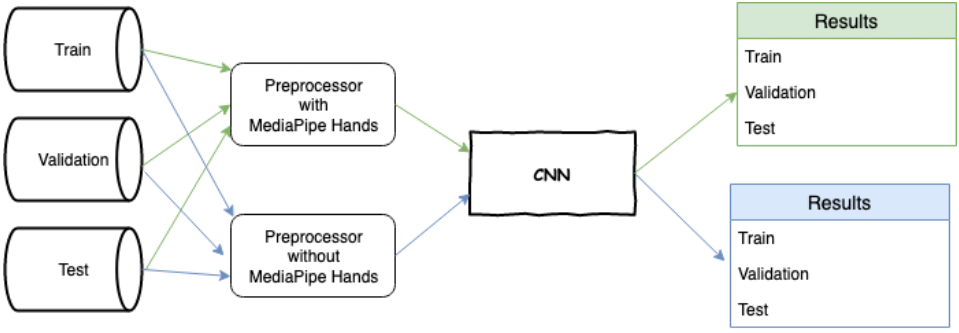


Figure 5: Experiment Setup

Results Training Data

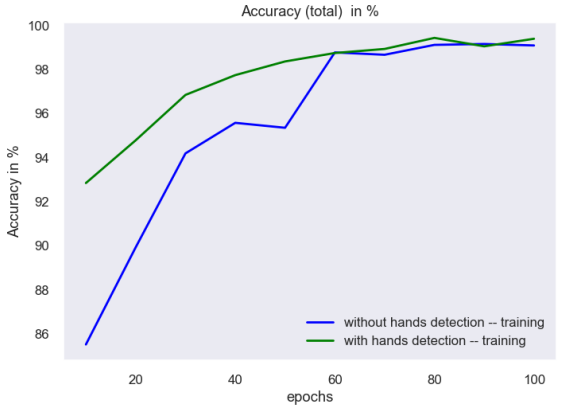
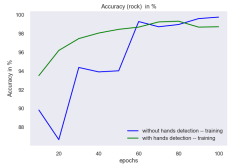
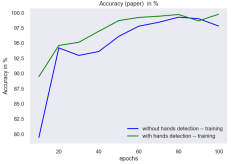
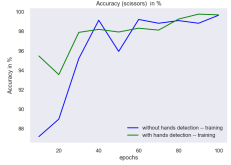


Figure 6: Total Accuracy on Training Data







Results Validation Data

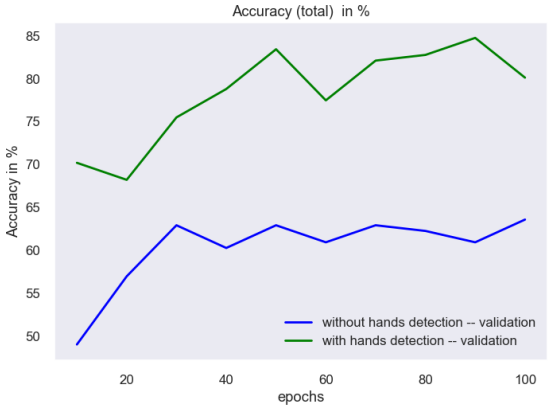
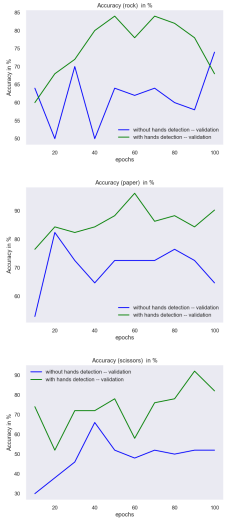


Figure 7: Total Accuracy on Validation Data



Results Test Data

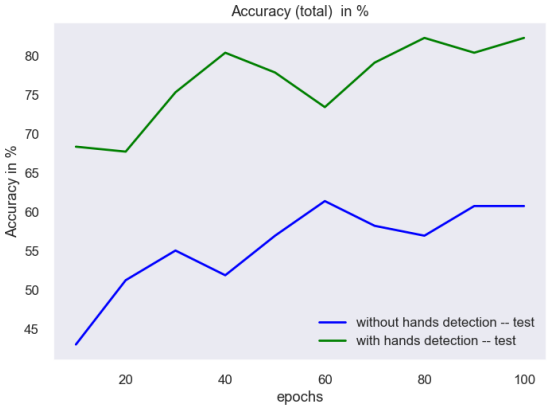
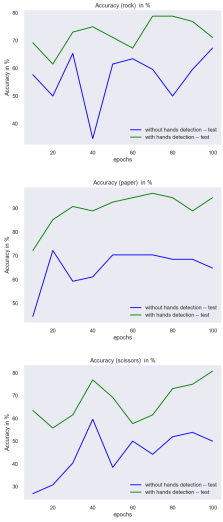


Figure 8: Total Accuracy on Test Data



Discussion

Results show that the preprocessor with image cropping based on Mediapipe Hands:

- Training Data: performed better at less training steps
- Validation Data: performed better for each test model iteration
- Testing Data: performed better for each test model iteration

Discussion

Results show that the preprocessor with image cropping based on Mediapipe Hands:

- Training Data: performed better at less training steps
- Validation Data: performed better for each test model iteration
- Testing Data: performed better for each test model iteration

$\Longrightarrow$ **H0** is probably **not correct** and an alternative has to be considered

Discussion

Results show that the preprocessor with image cropping based on Mediapipe Hands:

- Training Data: performed better at less training steps
- Validation Data: performed better for each test model iteration
- Testing Data: performed better for each test model iteration

$\Longrightarrow$ **H0** is probably **not correct** and an alternative has to be considered

$\Longrightarrow$ based on the presented results it is to assume that **reducing the complexity of the dataset** by removing the background (unimportant parts of the image) **leads to better model performance**.

Thank you!

References i

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.

Model Engineering – Dropout v BatchNorm

Explainable Machine Learning - Deep Learning Life Cycle

Jonas Amling Baptiste Bony Benedikt Marsiske

February 1, 2023

University of Bamberg

Table of contents

Research Question

Research Question

Our main Model Engineering challenges:

- Accuracy performance
- Prevent overfitting for the model to be generalizable to new data

Research Question: **Do dropout layers prevent overfitting ?**

In addition to Dropout, we tried another regularization technique, namely Batch Normalization.

Overfitting

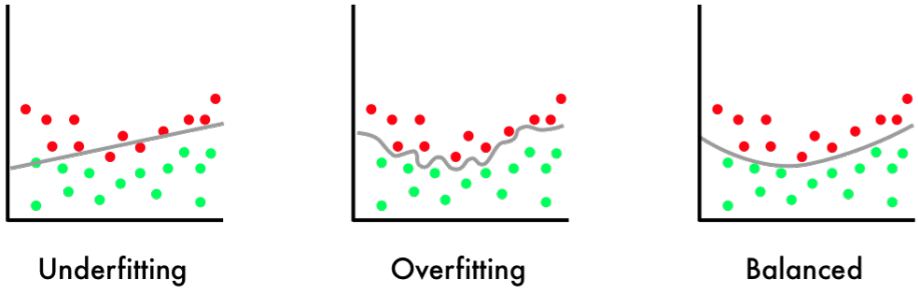


Figure 1: Visual explanation of Overfitting

Model Engineering Process

Overview of the different datasets

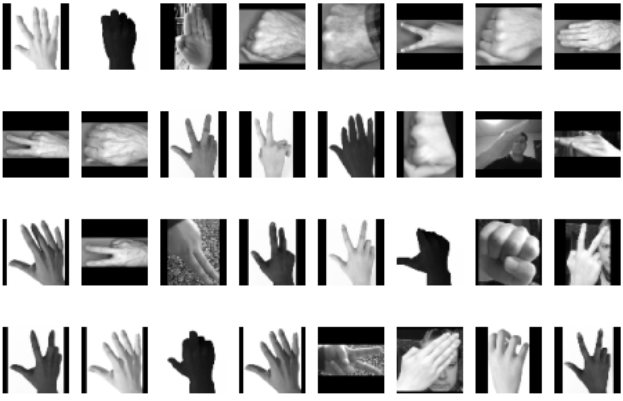


Figure 2: Sample of the Training Dataset

Overview of the different datasets

Figure 3: Sample of the Validation Dataset

Overview of the different datasets

Figure 4: Sample of the Testing Dataset

Our first model

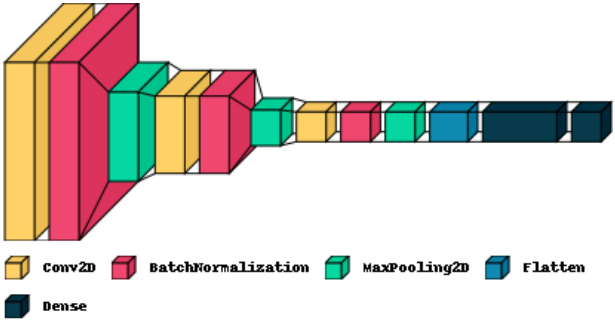


Figure 5: Visualization of our first model

VGG16 [1]

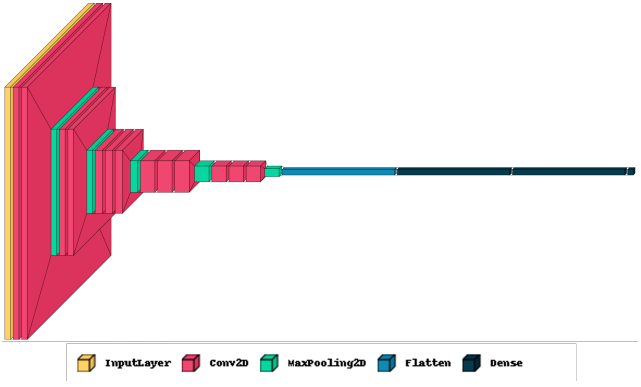


Figure 6: Visualization of the VGG16 model

Our custom model

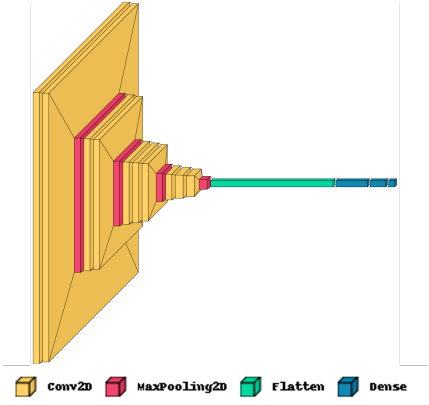


Figure 7: Visualization of our model

Potential Solution: Dropout

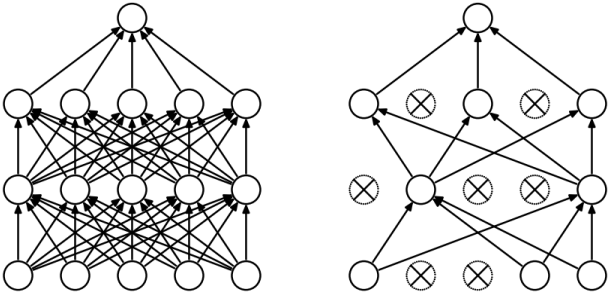


Figure 8: Scheme explaining the principle of Dropout Layers

Potential Solution: Batch Normalization

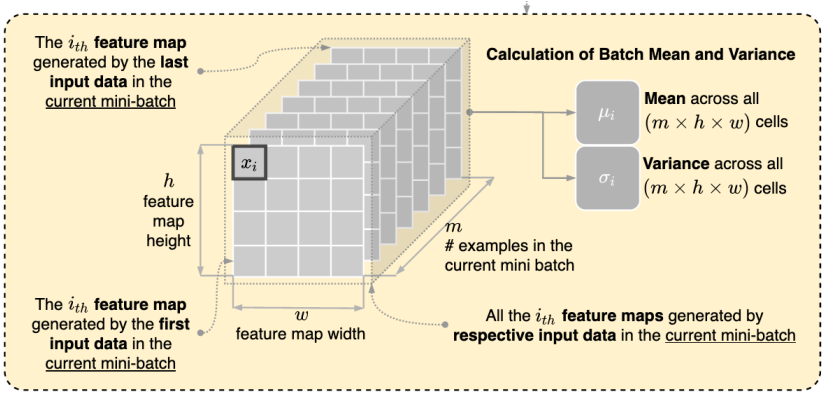


Figure 9: Scheme explaining the principle of Batch Normalization

Experiment

Model performance without any Regularization

Figure 10: Training v Validation Loss

Figure 11: Training v Validation Accuracy

Model performance without any Regularization

Figure 12: Validation Accuracy in detail

Model performance without any Regularization

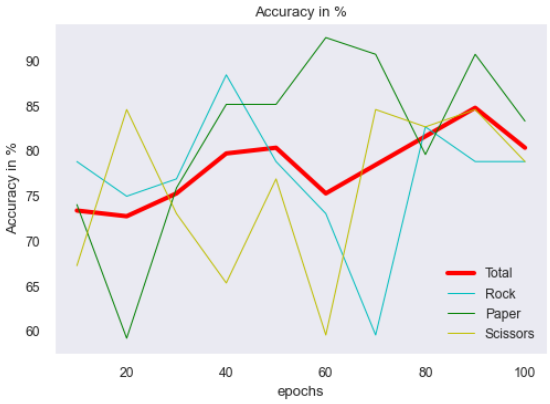


Figure 13: Testing Accuracy in detail

Model with Dropout

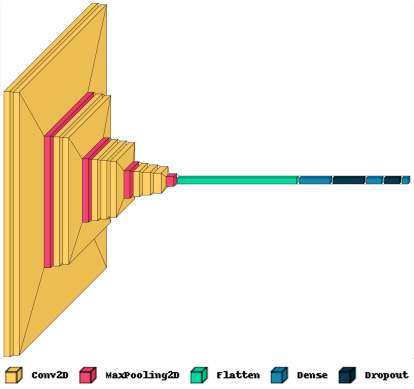


Figure 14: Visualization of our model with Dropout

Model performance with Dropout

Figure 15: Training v Validation Loss

Figure 16: Training v Validation Accuracy

Model performance with Dropout

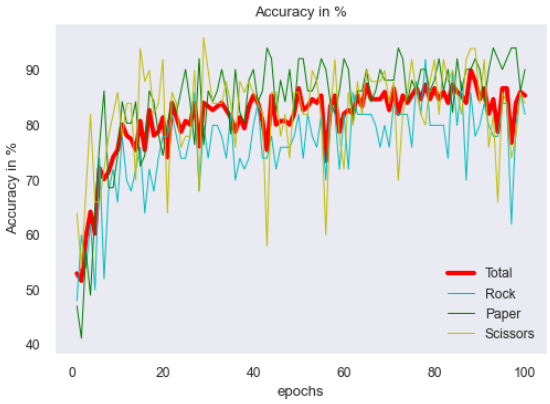


Figure 17: Validation Accuracy in detail

Model performance with Dropout

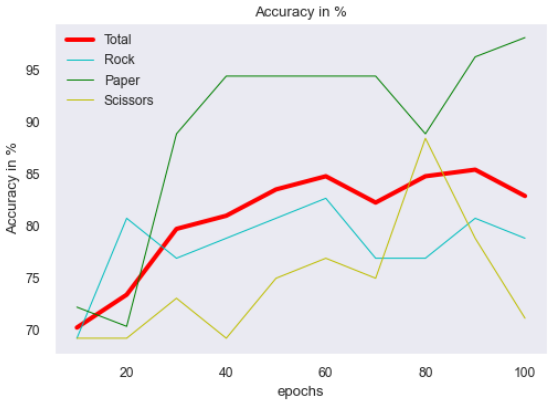


Figure 18: Testing Accuracy in detail

Model with Batch Normalization

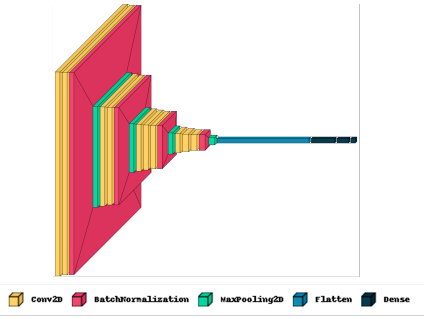


Figure 19: Visualization of our model with Batch Normalization

Model performance with Batch Normalization

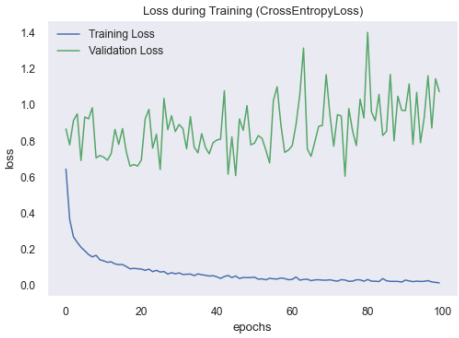


Figure 20: Training v Validation Loss

Figure 21: Training v Validation Accuracy

Model performance with Batch Normalization

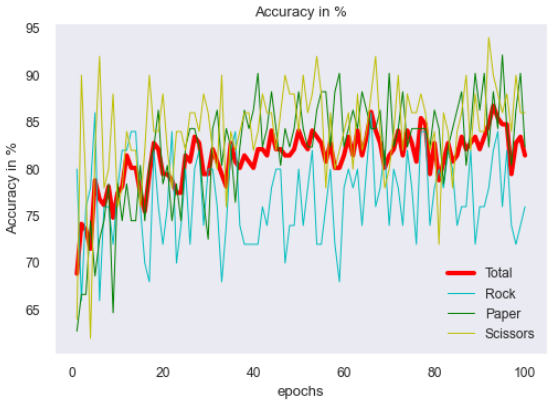


Figure 22: Validation Accuracy in detail

Model performance with Batch Normalization

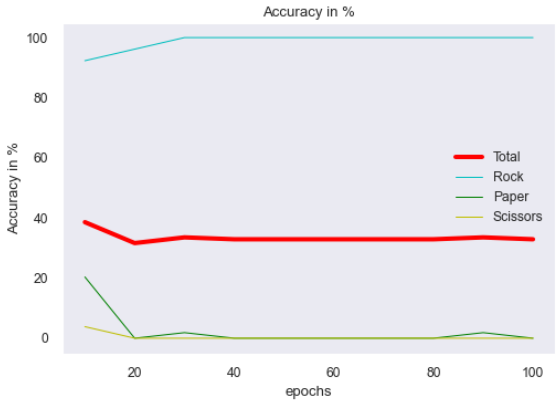


Figure 23: Testing Accuracy in detail

Model with both Dropout & Batch Normalization

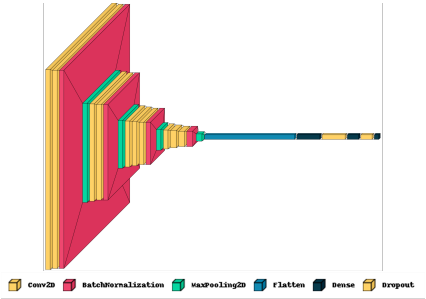


Figure 24: Visualization of our model with both Dropout & Batch Normalization

Model performance with both Dropout & Batch Normalization

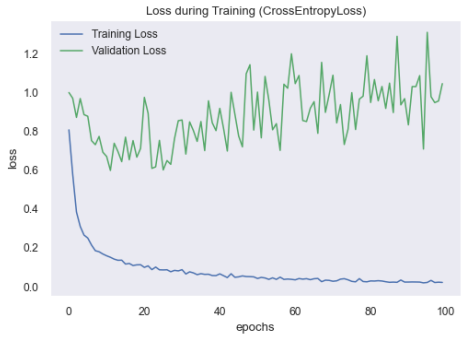


Figure 25: Training v Validation Loss

Figure 26: Training v Validation Accuracy

Model performance with both Dropout & Batch Normalization

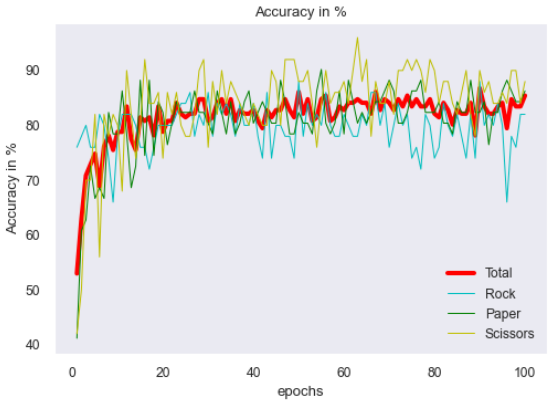


Figure 27: Validation Accuracy in detail

Model performance with both Dropout & Batch Normalization

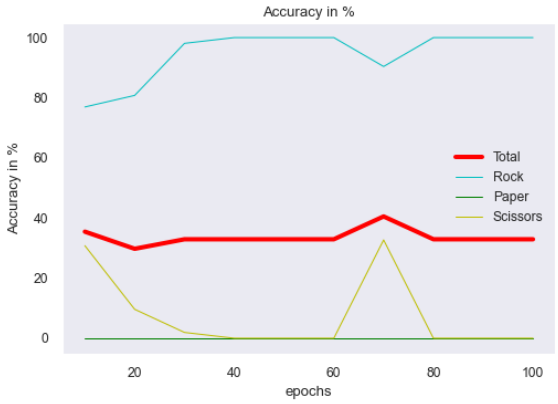


Figure 28: Testing Accuracy in detail

Comparison between the four cases

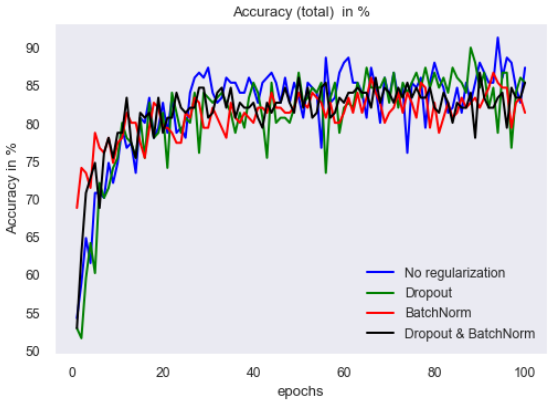


Figure 29: Comparison between all Validation Accuracies

Comparison between the four cases

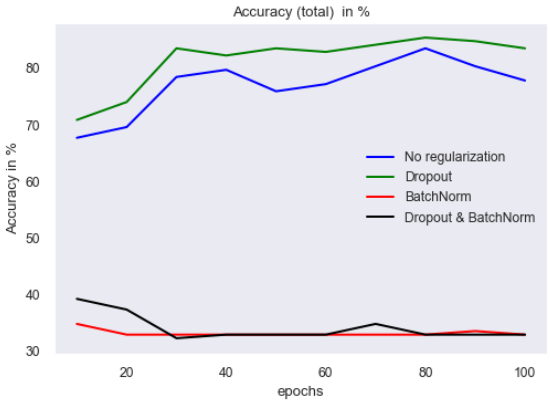


Figure 30: Comparison between all Testing Accuracies

Final results (with Dropout)

	Rock	Paper	Scissors	Total
Val Set	86,00%	90,20%	78,00%	84,77%
Test Set	78,85%	98,15%	71,15%	82,91%

Table 1: Accuracies on Val Set and Test Set for the Leaderboard

Thank you!

References i

K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

Model Evaluation – The Effect of Noisy Data

Explainable Machine Learning - Deep Learning Life Cycle

Jonas Amling Baptiste Patrice Francis Bony Benedikt Markus Marsiske

February 2, 2023

University of Bamberg

Table of contents

Research Question

Research Question and Introduction

Focus for evaluating our trained model:

- How well does the model perform?
- Is there a difference between the classes?
- How robust is our model?

Research Question and Introduction

Focus for evaluating our trained model:

- How well does the model perform?
- Is there a difference between the classes?
- How robust is our model?

Specific Research Question: **How does the model perform on distorted data? Does the usage of distorted test data lead to a worse model performance compared to the same test data without distortion?**

Basic Model Evaluation

Model Performance - Training Data

How well did our model perform on our training data?

```
testing against the training dataset
Accuracy of the network on the test set: 98.91792029559251%
Accuracy of rock : 99.0506329113924%
Accuracy of paper : 98.99598393574297%
Accuracy of scissors : 98.7109375%
```

Figure 1: Accuracy at the end of training

Confusion Matrix - Training Data

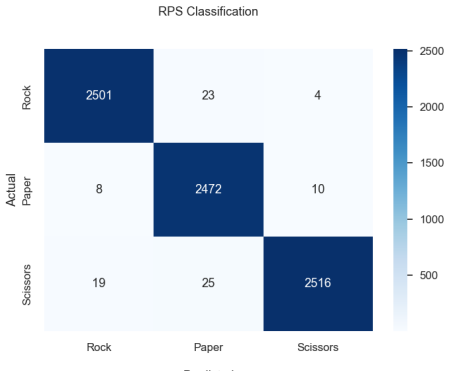


Figure 2: Numeric CM of Training Data

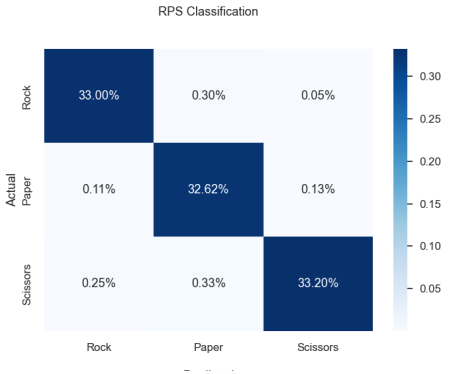


Figure 3: CM of Training Data (in %)

Model Performance - Validation Data

How well did our model perform on the provided validation data?

- custom made data
- no images of big datasets (significant portion of training data)
- incorporated in our training

Model Performance - Validation Data

How well did our model perform on the provided validation data?

- custom made data
- no images of big datasets (significant portion of training data)
- incorporated in our training
- model performance:

```
testing against the validation dataset
Accuracy of the network on the test set: 86.44067796610169%
Accuracy of rock : 84.21052631578948%
Accuracy of paper : 85.0%
Accuracy of scissors : 90.0%
```

Figure 4: Accuracy on validation dataset

Confusion Matrix - Validation Set

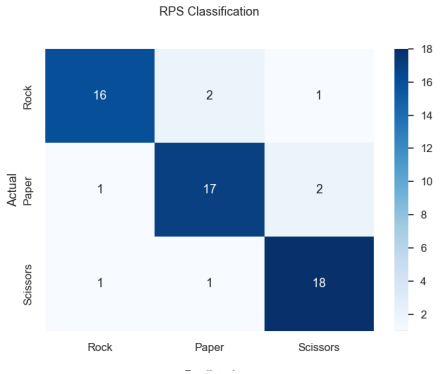


Figure 5: Numeric CM of testset

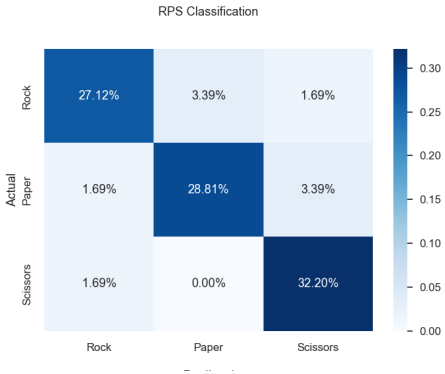


Figure 6: CM of testset (in %)

Model Performance - Test Data

How well did our model perform on the provided test data?

- more custom made data
- no images of big datasets (significant portion of training data)
- unseen

Model Performance - Test Data

How well did our model perform on the provided test data?

- more custom made data
- no images of big datasets (significant portion of training data)
- unseen
- model performance:

```
testing against the testing dataset
Accuracy of the network on the test set: 83.54430379746836%
Accuracy of rock : 80.76923076923077%
Accuracy of paper : 98.14814814814815%
Accuracy of scissors : 71.15384615384616%
```

Figure 7: accuracy on test dataset

Confusion Matrix - Unseen Testset

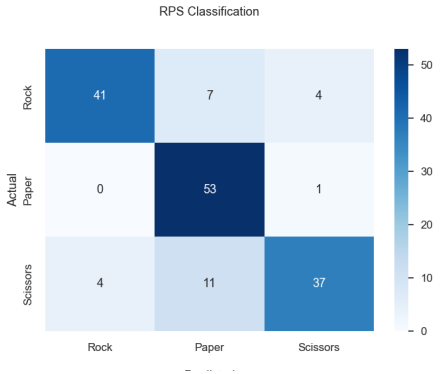


Figure 8: Numeric CM of unseen testset

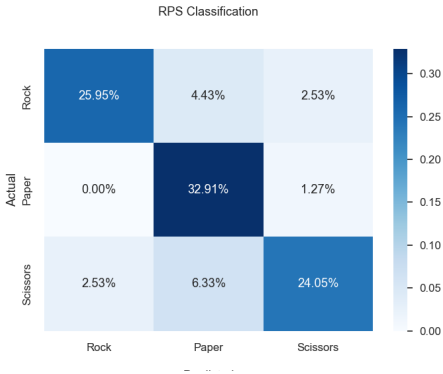


Figure 9: CM of unseen testset (in %)

Evaluating On Distorted Images

Image Distortion - Random Distortion

Each pixel has a chance to be removed (25%):

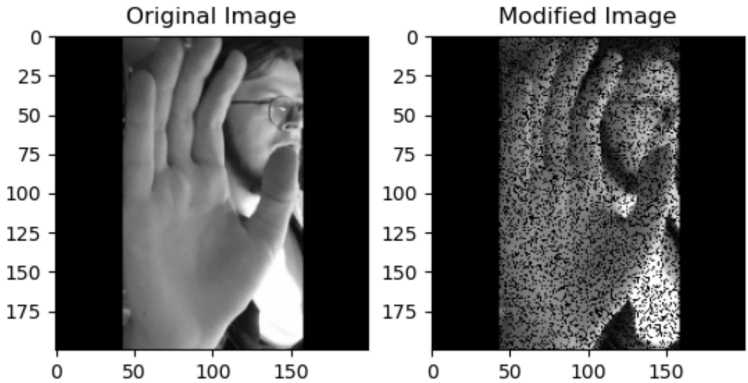


Figure 10: random distortion with a pixel elimination probability of 25%

Image Distortion - Gaussian Distortion

Gaussian Filter:

- follows normal distribution
- parameter: standard deviation (25 in our case)

Image Distortion - Gaussian Distortion

Gaussian Filter:

- follows normal distribution
- parameter: standard deviation (25 in our case)

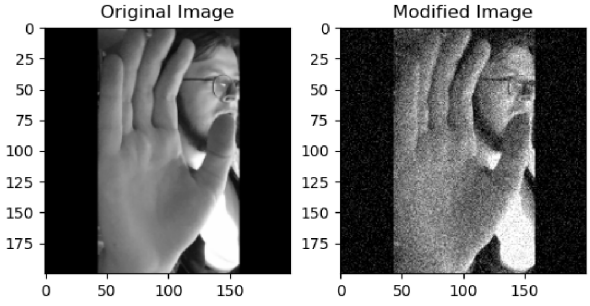


Figure 11: distortion using a Gaussian filter with SD = 25

Model Performance on distorted data

Random Distortion:

```
testing final_model_D_True_B_False against the testing dataset with random noise:
Accuracy of the network on the test set: 44.30379746835443%
Accuracy of rock : 67.3076923076923%
Accuracy of paper : 53.7037037037037%
Accuracy of scissors : 11.538461538461538%
```

Figure 12: Accuracy on randomly distorted testset

Gaussian Distortion:

```
testing final_model_D_True_B_False against the testing dataset with gaussian noise:
Accuracy of the network on the test set: 47.46835443037975%
Accuracy of rock : 28.846153846153847%
Accuracy of paper : 87.03703703703704%
Accuracy of scissors : 25.0%
```

Figure 13: Accuracy on testset distorted with a Gaussian filter

Random Distortion - Confusion Matrix

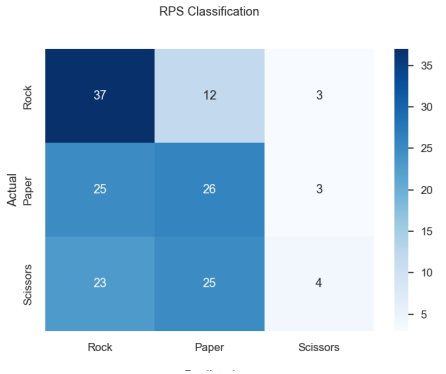


Figure 14: Numeric CM of distorted testset

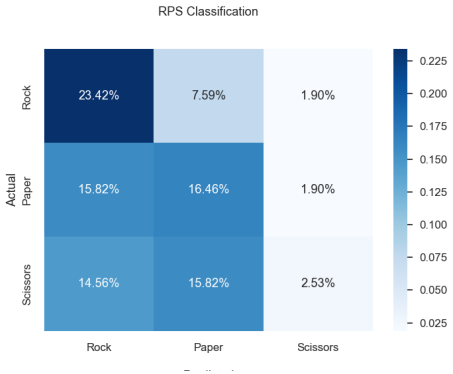


Figure 15: CM of distorted testset (in %)

Gaussian Distortion - Confusion Matrix

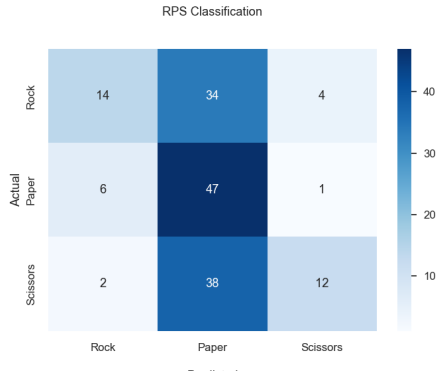


Figure 16: Numeric CM of distorted testset

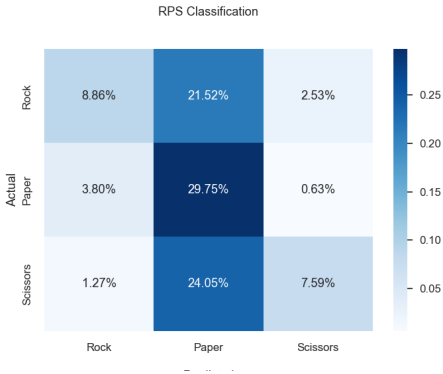


Figure 17: CM of distorted testset (in %)

Performances on Different Test Data

	Old	Current	No Regularization	With Noisy Data
Undistorted Testset	67.8%	83.5%	80.4%	61.4%
Random Distortion	42.4%	44.3%	38.0%	64.4%
Gaussian Distortion	55.9%	47.5%	58.2%	63.3%

Table 1: Performance comparison of various models on different test data

Conclusion

Conclusion

Things we learned evaluating our model:

- better performing model $\neq$ more robust model
- big impact of noisy data
- robustness can be improved by training with noisy data

Conclusion

Things we learned evaluating our model:

- better performing model $\neq$ more robust model
- big impact of noisy data
- robustness can be improved by training with noisy data

Answering our research question:

The usage of distorted test data does lead to a worse model performance compared to the same test data without distortion.

Thank you!