Data Engineering – Background Removal

Explainable Machine Learning - Deep Learning Life Cycle

Jonas Amling Baptiste Patrice Francis Bony Benedikt Markus Marsiske

January 29, 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
- $\implies$ 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

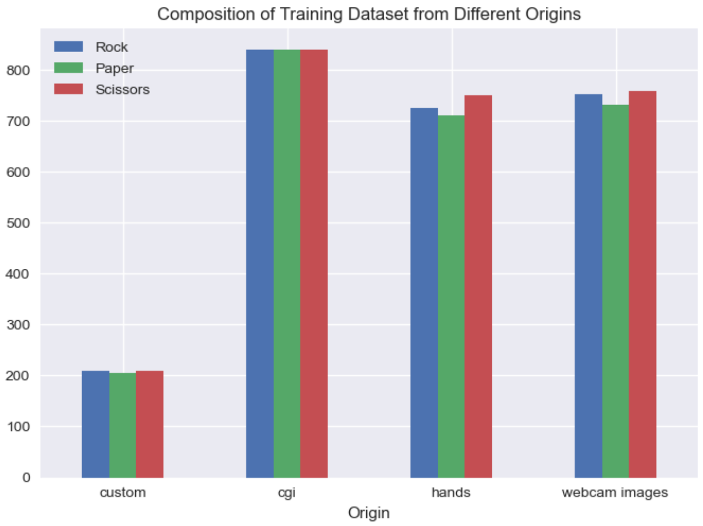


Figure 1: Distribution of individual Datasets

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

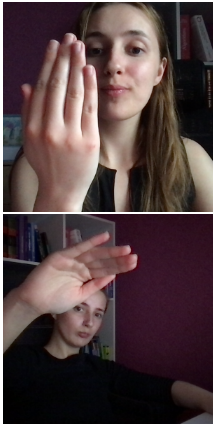


Figure 2: original

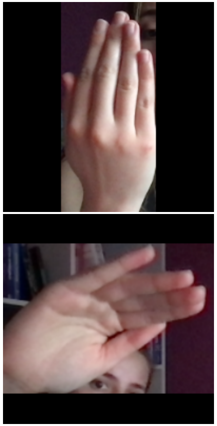


Figure 3: cropped

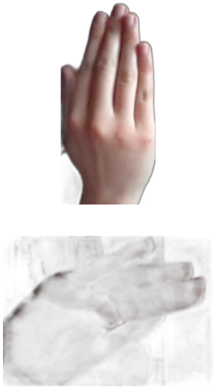


Figure 4: background removal

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

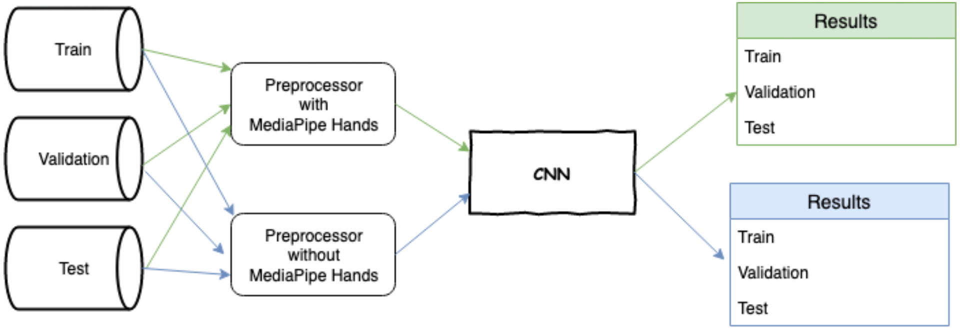


Figure 5: Experiment Setup

Results Training Data

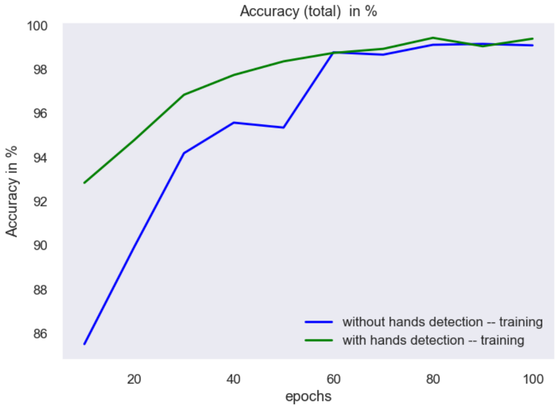
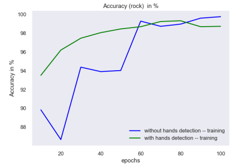
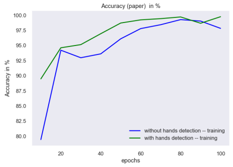
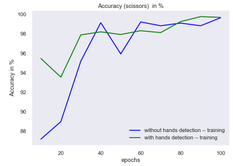


Figure 6: Total Accuracy on Training Data







Results Validation Data

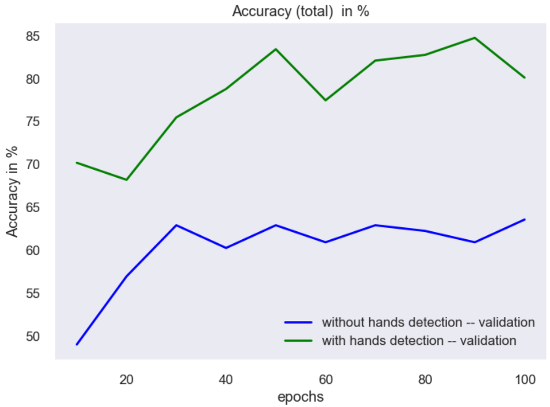
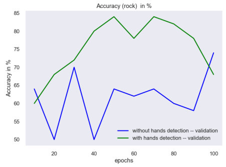
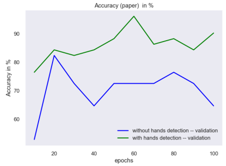
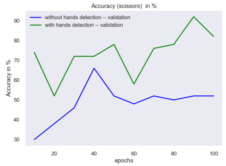


Figure 7: Total Accuracy on Validation Data







Results Test Data

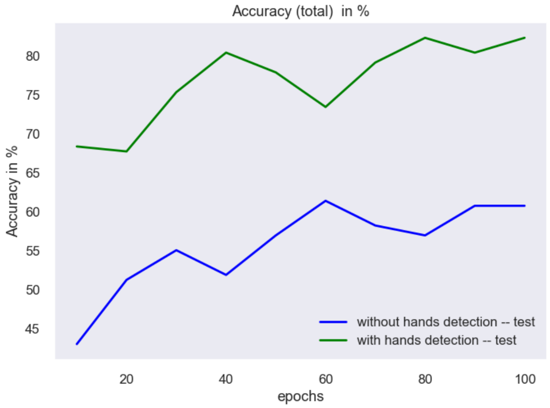
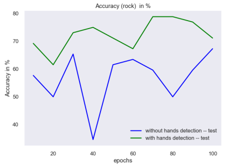
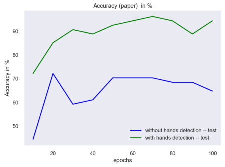
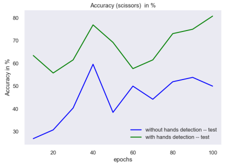


Figure 8: Total Accuracy on Test Data







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.