Classification of Sign Language using Image Segmentation and Hand Pose Estimation

Steffen Epp, Michael Mohr

January 2021

1 Introduction

An essential part of communication with the deaf community is sign language. Despite its wide use, no accurate count of ASL users has been taken, but it is estimated, that only a fraction of the American population is capable of using sign language. To enable the communication between ASL users and non-user, an interface between those two parties is essential. For this purpose, we propose an automated tool for interpreting sign language. This work focuses on localizing the hand and interpreting their different hand poses to robustly interpret it to a word of the ASL. There are several different sign languages, e. g. American Sign Language (ASL), German Sign Language (GSL), or Polish Sign Language (PSL). Given a suitable dataset, a model trained for a different language should yield similar results. To capture signs on a word-level basis, a time-dependent series of frames have to be analyzed and classified. For this, the hand pose and the location of every single finger is fundamental for classifying sign language. Subtle differences in hand poses need to be discernible to prevent ambiguity and guarantee an explicit allocation of hand pose to the different meanings. A robust detection and classification of the exact hand poses is therefore crucial for the classification task of signs.

In Kang, Tripathi, and Nguyen [1], a combination of image segmentation and classification is used to detect simple signs for the characters of the alphabet and the numbers $1-9$ in ASL. Through image segmentation a bounding box is created, which is used to cut out the relevant parts containing hands from the image. This is handed over as input to the classifier, which learned the different representations of the signs. We think that, as an intermediate step, an estimation of the hand pose is crucial. With hand pose estimation, we can opt with self occlusion and variances of different hand shapes. This is especially useful, since our training data for actual ASL signs is limited. Our approach can be split into two steps. In the first step, hands are detected and cut out from the original image. Therefore, a combination of classification on a pixel level and bounding boxes are applied. Hereby, the bounding boxes indicate the location of the respective hands. In the second step, based on the results from the first step, we will detect the respective hand poses and classify them to a specific sign.

The paper is organized as follows: In Section 2, several work related to our approach will be discussed. Section 3 outlines our approach and the structure of our proposed pipeline. Then, the experimental setup and the used datasets will be introduced in Section 4. In Section 5 and Section 6, we present our results with an afterwards discussion. In the last Section 7, we finish our work with a conclusion.

2 Related Work

When comparing different researches on the topic of hand segmentation and hand posture estimation, the first question is which type of images to process. There are several approaches that estimate 3D hand poses from RGB images [2] [3] [4]. But, the estimation of hand postures or even the segmentation of hands has a lot of ambiguities regarding RGB images. Therefore, additional information provided by depth images help to disambiguate the information.

Hand Segmentation Along with a model for image segmentation based on hands, Bojja, Mueller, Malireddi, *et al.* [5] also introduced a new dataset containing 158,315 images and their respective annotation masks. The images are depth-images and the annotation images contain numeric values for three classes. They divide the parts of the image in left hand, right hand and into background. Different hand poses from various users have been captured with an RGB-D camera. The users were wearing skin-tight brightly colored gloves. The annotation images were automatically generated by color segmentation with a small false positive rate. Their proposed architecture is a hybrid encoder-decoder. For up-sampling in the decoding part of the model, a hierarchy of transposed convolution layers to improve sharpness and detail have been used. Additional, skip-connections between the encoder and the decoder part have been utilized. Instead of pooling and unpooling layers, strided convolutions for encoding and strided transposed convolutions for decoding have been used. Pooling layers are helpful in a classification task, as they provide invariance to local deformations, which is not helpful in semantic segmentation. HandSeg has been evaluated on their own dataset as well as on various other datasets and is on par with state-of-the-art methods in terms of accuracy.

In Li and Kitani [6] the task of pixel-level hand detection in the context of ego-centric cameras has been analyzed. The ego-centric view offers a foundation to analyze hand-object manipulation and hand-eye coordination. For this, a different experimental setup has been used. Thereby, the user is wearing the camera on top of his head instead of standing in front of a camera. This leads to different body motion and transitioning in lightening. In their work, Li and Kitani [6] provide two dataset, including videos of ego-centric views of hands. These videos have been recorded indoor as well as outdoors. Li and Kitani [6] also proposed an own model, which they compared to several baseline models. Their model is a mixture of local features and global appearance. The local features are calculated by applying a bank of 48 Gabor filters and spatially varying local gradient histograms. For the global appearance model, a collection of regressors indexed by a global color histogram has been trained.

In Bambach, Lee, Crandall, *et al.* [7] a model for hand detection based on an ego-centric view has been introduced. In their work, Bambach, Lee, Crandall, *et al.* [7] introduce the new dataset EgoHands. The dataset consists of videos of two people interacting with each other. The videos have been captured via a head mounted camera. Overall, 48 unique combinations of videos has been captured with a Google Glass camera, which recorded 720×1280 video at 30 Hz. The ground truth labels have been manually annotated. For this, random 100 frames from each video have been annotated with pixel-level hand masks for overall 4 different classes. The classes have been separated into both users left and right hand. For their hand detection model, Bambach, Lee, Crandall, *et al.* [7] propose an approach to candidate window sampling using Convolutional Neural Networks (CNN) and appearance models.

Hand Pose Estimation For the task of solving hand pose estimation problem, there is currently no state-of the art method. There are several different approaches. In this section, we are not focusing on RGB image based methods but on methods based on depth map images, since our concept idea is based on them. Such approaches can be categorized into estimating 2D or 3D skeletons, detection- and regression-based algorithms.

In Tang, Jin Chang, Tejani, *et al.* [8] research, an architecture is presented for real-time 3D hand pose estimation. Hereby a latent regression forest (LRF) structure was used to identify in a single depth image 16 joints in the hand. The method to identify the different fingers and their position can be viewed as a coarse-to fine search process for skeletal joints, guided by the LRF-based, unsupervised learned model of the hand to enable a structured search. Beginning at the root, the offset to the children gets minimized at each level. This process starts with the whole hand as input, recursively dividing the input region which later on, when the region of interest is reduced, leads to a node, that matches a skeletal joint position. This approach is not only capable of enforcing learn global kinematic constraints, it is also possible to generalize to hands of various shapes, sizes and point of views. Their results on two different datasets show, that LRF outperforms baselines and prior arts in both accuracy and efficiency

In the research paper of Sinha, Choi, and Ramani [9], a regression-based method was used to find 21 joints in the hand, based on a depth map. The core network of this approach is a convolutional

neural network architecture to reduce the dimensionality of the depth map. To identify the hand posture, a particular focus was set to identify the location of joints in each finger independently, by also training a separate network for each. Hereby, a pose estimation matrix is inputted with a combination of the deep activation vectors of nearest neighbors, including their joint angles and the activation vector of the input image. Their focus was not only to identify spatial features, but also the inclusion of temporal activation features from previous images. Also, normally the task of hand segmentation and hand posture estimation is pipelined by two stand-alone approaches. In this research, the same deep convolutional neural network was used for both problem-solving tasks. According to their results, an architecture based on convolutional networks outperforms PCAs and random forests at the present time.

The approach of Moon, Chang, and Lee [10] presents a method for real-time continuous pose recovery of markerless complex objects also from a single depth image. This approach is detection-based using heat maps. Normally, fully connected networks are used to maintain spatial information. In a detection-based approach, instead of joint coordinates an encoded spatial-form representation (SFR) was designed as the learning target including a decoder for backtransformation to joint coordinates. The most common decoder also used in this approach are heat maps. In this approach, the input depth image gets converted into a three-dimensional voxel map. Additionally, a three-dimensional heatmap was established in this voxel map to extend the two-dimensional problem to a three-dimensional one. But the detection-based method is not easy to implement in the hand pose estimation problem. This is due to the fact, that it is not only necessary to identify the 2D coordinates in the image, but also the depth coordinates. This is problematic,, since a representation is difficult when using a heatmap. Furthermore, this approach relies on an artifically defined preprocessing to allow a transformation into a 3D voxel map.

In the research of Zhang and Zhang [11], the 3D hand pose estimation problem from single depth images can be expressed as a mapping function from depth images to 3D joint coordinates. This activation function can be approximated with a single neural network. But instead of directly applying this, a so called Pixel-wise Regression was applied. Hereby, an encoded spatial-form representation (SFR) gets designed, which divides the 3D joint coordinates into two parts, plane coordinates and depth coordinates. This has the target to bypass the need to flatten the input data, which would result in the loss of all spatial information in the original image. To recover the joint coordinates, the representation needs to be encoded again. By putting the decoder after the convolutional layer, still inside the model, the SFR and the direct supervision from 3D coordinates benefit the result of the neural network model. This De- and Encoding process is applied by the two modules named Plane Regression and Depth Regression.

A model for simple fingerspelling classification has been introduced in Kang, Tripathi, and Nguyen [1]. The model consists of two steps. First, image segmentation is applied to the images. Bounding boxes are then created from the resulting segmented images. In the second step, classification is applied on the images in the predicted bounding boxes. The architecture of the classification network is similar to AlexNet Krizhevsky, Sutskever, and Hinton [12]. The introduced pipeline is interesting and similar to our approach. We decided to train the hand segmentation and hand pose estimation on different datasets. There are not many datasets suitable to train segmentation and classification on. This is because the collecting and annotating of the data is cumbersome and labor intensive. Since the proposed work is closely related to our work, we will use the dataset to evaluate our pipeline.

3 Method

In this section, the pipeline of the proposed model will be introduced. Each step of the pipeline will be explained in more detail below.

3.1 Pipeline Overview

The pipeline consists of four main steps. The processing of the pipeline mainly focuses on detecting the hands and estimating their poses. The basic procedure is structured as following:

- **Step 0: Preprocessing** As an initial step, the input images are reshaped and normalized to fit the model.
- **Step 1: Hand Segmentation** In this step, the input depth images are segmented between hands and background. The output is a segmented image, in which every pixel has been classified in one of the respective classes.
- **Step 2: Bounding Box Prediction** This step is a transition to connect the output of step two with the expected input of step three. It reduces irrelevant information outside of the box and therefore improves the identification between the hand and noise on the image.
- **Step 3: Hand Pose Estimation** The hand pose estimation is a process of localizing the position of a hand in an image and modeling the human hand as a set of some parts, in this case as a set of different joints. Hereby, the image gets preprocessed to calculate the center of mass of the image and uses these information to predict 16 joint coordinates, three of each finger and the center of the palm.
- **Step 4: Classification of American Sign Language** The last step of the Classification Pipeline describes a mapping function for classifying the results of the model of hand posture estimation. Hereby, the main focus was set to recognize the finger position.

3.2 Step 0: Preprocessing

The input of the pipeline are depth images, usually recorded with a camera, e. g. Kinect. These images are normalized so that the measured signals have a unit average. If the image size differs from the defined size for the model, the images will be resized accordingly.

3.3 Step 1: Hand Segmentation

Step one of the pipeline can be described as semantic image segmentation. The goal is to classify each pixel in the image. The input for this step are preprocessed depth images. The model then will create a segmentation for the hands on these images. We did not differentiate between the hands, since we focus on the ASL signs for the first nine numbers. These signs are static, so they can be represented with only one frame and are waved with only one hand.

The model for this step is taken from Bojja, Mueller, Malireddi, *et al.* [5]. At the time of this work, the code of their model was not publicly available yet, so we recreated the proposed model as described in their paper. This model is build with an Encoder-Decoder architecture using convolutional layers. Skip connections between the respective encoder and decoder parts are used to maintain global information in deep layers and enable a deep network architecture, as depicted in Figure 1. This is done by simply adding the outputs of the activation layers from the encoder parts to the outputs of the convolutional layers in the decoder part. The model consists of eight blocks, four in the encoder and four in the decoder. Each block is build up with a convolutional layer, followed up by a batch normalization and a Leaky ReLU activation layer.

Instead of pooling and unpooling layers, convolutions with stride are used in the encoder part and transposed convolutions with stride are used in the decoder part. Pooling layers are useful in classification tasks as they provide invariance to local deformations, which is exactly the opposite of what this model tries to archive, that is, a segmentation output that is pixel-level accurate [5]. The loss is computed via categorical cross entropy.

$$L(y, \hat{y}) = -\sum_{i=1}^{n} y_i \cdot log(\hat{y}_i)$$

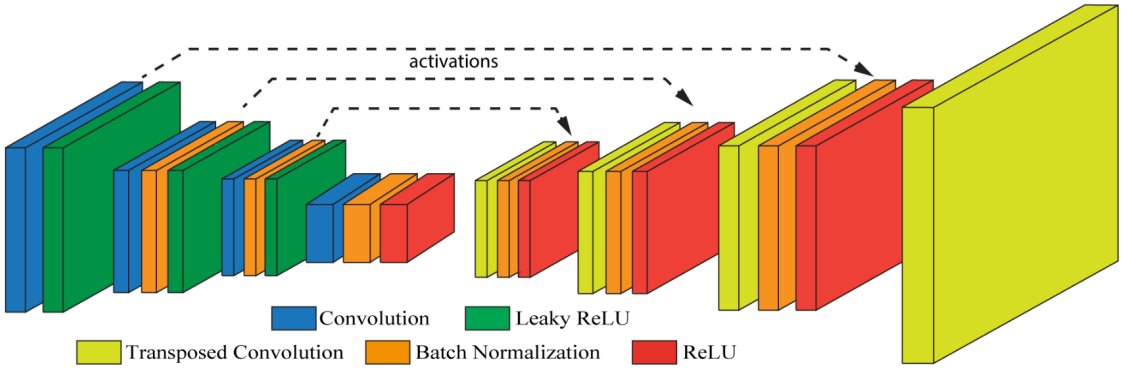

Figure 1: Architecture of the model used for semantic image segmentation in step one.

The last layer of the model is a softmax layer, which is used for classification. In our case, there are two possible classes: hands and background.

3.4 Step 2: Bounding Box Prediction

Since the output of the previous step is a pixel-level segmented image and the input for the next step are depth images centralized on hands, this works as a transition of these two major parts of the pipeline. From the predicted segmented image, a bounding box is created for the hand class. This bounding box contains all pixels that have been classified as part of a hand. The inside of this box is then cut out from the original image and padded. For padding, we take the first pixel values in the upper right corner and add them around the insides of the predicted bounding box. Such a value can almost always be classified as background. If necessary the size will be rescaled accordingly. The padded image is then directly used as input for step 4.

3.5 Step 3: Hand Pose Estimation

The approach applies 3D Hand pose estimation from a single depth image as input. The most striking problems for hand pose estimation are the complex structure of the human hands and the variety of view-points causing self-occlusion [13] [14]. Existing methods with deep learning so far are mostly regression-based methods, which treat hand pose estimation as an end-to-end problem [11]. Hereby, the neural network consists of one neural network to regress the 3D coordinates. This simplifies the task of finding and learning features for the neural network. The downside with these networks is the necessity to use fully-connected convolutional neural networks. To get the joint coordinates, the image needs to be flattened resulting in a loss of spatial information of hand structures in the original image and lack a direct supervision of joint coordinates. But spatial information are necessary to recognize a complex structure like hands and furthermore, researches [15] showed, that detection-based approaches show higher accuracy then regression-based methods. Detection-based approaches use fully-connected neural networks, but instead of joint coordinates, a different learning target is used. Hereby, an encoded spatial-form representation (SFR) is designed, which allows to maintain all spatial information by transforming the representation and providing a decoder for backtransformation to joint coordinates after prediction. But since depth images are only a projection of real 3D data, the transformation is not easy to achieve. To overcome those problems, a method called Pixel-wise regression was used here [11]. This method, further depicted in Figure 2, uses SFRs in combination with a differentiable decoder (DD). The main idea lies in dividing the 3D joint coordinates in two separate parts, namely processing the plane coordinates in the module Plane Regression and depth coordinates in the module Depth Regression. Since we use the ICVL Hand Posture Dataset, the input for the model will be depth images, which only contain one single complete hand, after applying general filter methods and resizing of the image. Since there

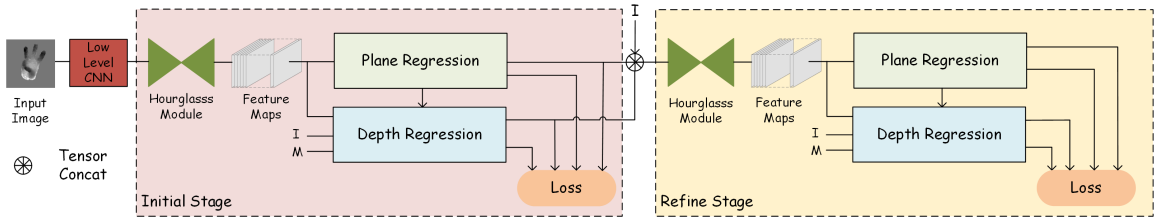

Figure 2: Overview of the architecture of the designed model used for hand pose estimation, proposed by Zhang and Zhang [11]

is always the possibility of background in images, we first use boundary boxes to roughly remove the background. Exploiting the fact, that the hand will always be the closest obejct to the camera, the center of mass (com) will be calculated to locate the hand in the image and precisely calculate bounding boxes. Afterwards, an empirical threshold is applied to further improve the quality of the images. The feature extraction is processed by the Hourglass Module. The hourglass module is also related to encoder-decoder architectures. It is widely used in segmentation and pose estimation problems and can provide features extracted in multi-scales. The design of the hourglass is motivated by the need to capture information at every scale. Since the dataset provides 3D coordinates in a 2D plane, two SFRs are used. The first SFR provides heat maps to encode plane coordinates (Plane Regression). The second is used to provide a local offset depth map to encode depth coordinates (Depth Regression). For the heatmap it is assumed that the closer the pixels are to the predicted joints, the larger the value will be in the end. Both, the Plane Regression and the Depth Regression, take feature maps as input and further put them through a CNN network, as already mentioned, one predicting the heat map and the other predicting the depth map for each joint. In the plane Regression module, the heatmap goes through a center of mass based convolution. This converts the heatmap, resulting in normalized coordinates representing plane coordinates. After the initial stage, the heatmap, depth map and representation scale image get concatenated and used as input for the refinement stage. This architecture of a multistage network improves the accuracy by refining the results and dealing with occlusions. By using the same structure of the Initial Stage also for the Refine Stage, the overall training speed can be reduced when comparing with differing architectures. The loss is calculated by using the mean-square-error of predicted values to the annotations of the dataset.

Dataset To train our model, we chose the ICVL Hand posture dataset of Tang, Jin Chang, Tejani, *et al.* [8]. Currently, there is no best practice on what appropriate amount of joints are to model hands. The ICVL Hand posture dataset, though, decomposes the hand pose estimation problem into estimating the location of 16 skeletal parts on the hand model. Further details about this and all other datasets used in this paper are introduced in Section 4.1.

3.6 Step 4: Classification of ASL

In the last step of the pipeline, the ASL sign will be predicted. The input for this step are the predicted joint locations of the hand. We described each hand pose of an ASL sign with joint locations. From these joint locations, we are able to determine whether a finger is stretched or bent. With this simple method, we are able to differentiate uniquely between different signs. In this approach, we classify the numbers from one to nine but did not implement further signs. This is due to the fact, that there was ambiguity between different signs of the ASL alphabet as depicted in Figure 3. We check which hand pose matches with the defined joint locations and predict the respective ASL sign.

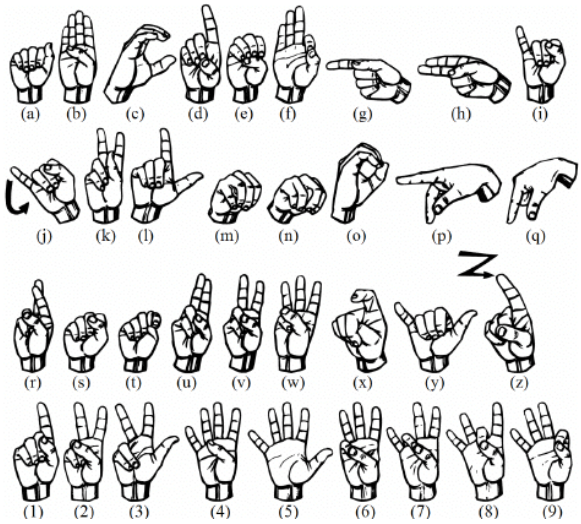

Figure 3: ASL alphabet and the first nine numbers.

4 Experimental Apparatus

In this section, we describe how we applied our models and set up the evaluation. We also give an overview over the datasets we used to train our models. We evaluated each method separately. The evaluation methods are described in the following subsections.

4.1 Datasets

Overall two datasets have been used for training and evaluation. We give an introduction of the datasets and describe the way we used them in our training and evaluation phase.

- **HandSeg** This dataset has been automatically labeled for hand segmentation using depth images [1]. It is introduced in Bojja, Mueller, Malireddi, *et al.* [5]. The dataset consists of $158,315$ depth images and their respective mask images. The images and masks are of size 480×640. An RGBD camera has been used to record the images. To automatically annotate the images, the subjects wear colored gloves for each hand respectively. Since the colored hands are not relating to most real-life scenarios, only the depth images are used in the dataset. The mask images contain the class value for each pixel. There are three classes namely background, left, and right hand respectively. An example of the depth images and the corresponding masks can be depicted in Figure 4.

- **ICVL Hand Posture** The ICVL Hand Posture dataset [8] collected fully 3D annotated depth images, with an image size of 480×640. The images were collected from sequences, which were recorded from 10 different subjects with varying hand sizes. Thereby, each subject was asked to make various hand poses. The sequences were recorded with an Intel Creative depth sensor and were sampled with three fps. An illustration of 26 different postures were shown as aid. Afterwards, in-plane rotations were applied, extending the amount of so far 20000 images to an overall of 180000 annotated training images. For testing purposes, two different sequences were collected. Each of which with 1000 frames, showing several different hand poses, scales and viewpoint changes. The annotations are arranged as followed: Every image is described by $16x3$ numbers, indicating the x, y and z coordinates of 16 joint locations. The variables x and y describe the position of the pixel in the picture, and z describes the depth of the pixel. The 16 different joint locations are Palm, Thumb root, Thumb mid, Thumb tip, Index root,

[1] `https://vision.uvic.ca/pubs/2019/bojja2019handseg/page.md`

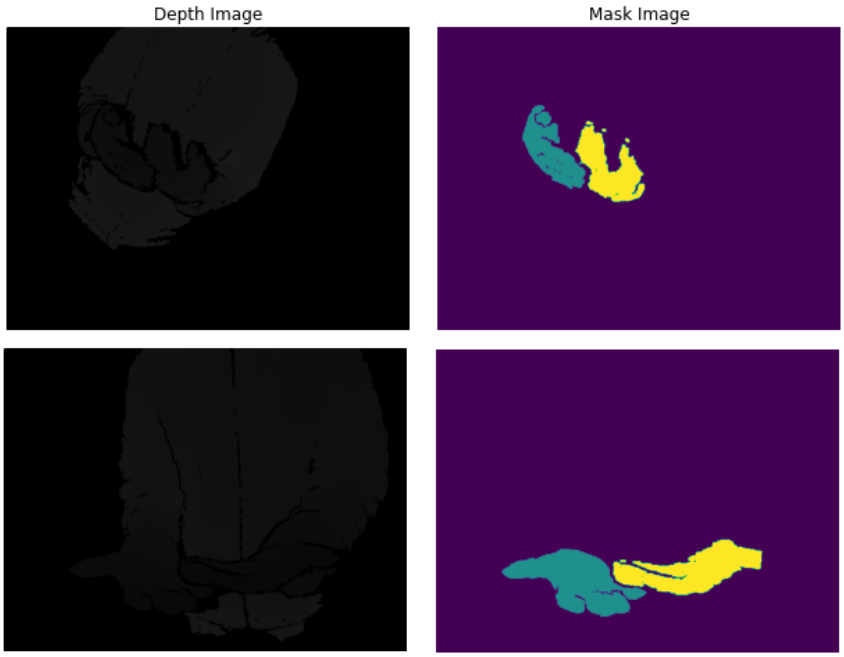

Figure 4: Two depth images from the HandSeg dataset with the respective annotation images.

Index mid, Index tip, Middle root, Middle mid, Middle tip, Ring root, Ring mid, Ring tip, Pinky root, Pinky mid, Pinky tip.

4.2 Image Segmentation Setup

The model for the image segmentation has been trained and evaluated on the HandSeg dataset. Around 15% of the overall images in the dataset have been used for the evaluation. From the remaining training data, 20% have been used as validation data. The dataset has been shuffled before splitting it into training and validation sets, since the images were sorted by users performing their different poses. The model has been trained with a learning rate of 10^{-4} and the categorical crossentropy loss function. The kernel size in the (transposed) convolutional layers was set to 3. For optimization, adaptive momentum optimization (Adam) has been used. To measure the performance of this model we chose two different metrics, accuracy and mean Intersection over Union (IoU). The accuracy is calculated by counting all true positive and true negative classified pixels, divided by the overall amount of pixels.

$$\text{Accuracy} = \frac{TN + TP}{N}$$

Where N is the amount of all pixels in the image and TN are true negatives and TP are true positives. The accuracy metric for image segmentation alone is not very expressive, especially when class imbalance is occurring. This is the case when one class is predominantly present in the image, e. g. background. For example if there is a lot of background in the image, the model could predict the background class for every pixel and still achieve a high accuracy. A more expressive metric for image segmentation is IoU, see Figure 5. This value is calculated by dividing the area of overlap from the predicted and annotated pixels by the area of union for each respective class.

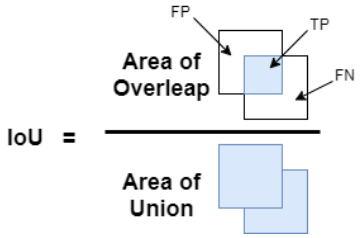

Figure 5: Intersection of Union (IoU) illustration.

The more these areas overlap, the better the performance of the model. This can be observed, when the IoU value is approximating 1. The area of overlap is represented by TP and the area of union by the sum of TP, FP and FN. Hereby, FP stands for false positive and FN for false negative. Thus, the formula for calculating the IoU score is:

$$IoU = \frac{TP}{TP + FP + FN}$$

The mean value can be calculated for each IoU score to measure the performance on one prediction. We used Google Colab [2] with a Tesla K80 GPU with 12 GPU-RAM to train this model.

4.3 Hand Pose Estimation

The model for the Hand Pose Estimation has been trained and evaluated on the ICVL Hand Posture dataset. For evaluation, 2000 test images have been used.

The input depth image goes through a low-level CNN to extract low-level features and for down sampling the image to the representation scale. We used a bounding box size of $125x125$. For the padding of the first convolutional layer, we used a kernel size of 7. The model was trained on an NVIDIA Quadro P1000 with a CUDA Version of 10.2. We used the Tensorflow framework, version 2.0.0 and used the Adam optimizer with a learning rate of $2e-4$. Furthermore, we calculated the center of mass to again set the center of the bounding box and reduce the image size to 64×64. We set the bottleneck vector size of the hourglass module for feature reduction to a size of 128. The model was trained in 50 epochs. Hereby, we highly differ from the initial paper which suggests to train the model in 10 epochs. For the calculation of the heatmap, softmax was used.

To then evaluate the model, we calculated the Mean Standard Deviation between the position of every predicted joint to the ground truth, provided by the annotations of the test data. Hereby, the loss of the Plain Regression and the Depth Regression was summed up for each stage. This step is again repeated to receive the mean value for each stage (Initial and Refine).

5 Results

The following section presents the results of our experiments. The first subsection covers the hand segmentation model used in the first step of the pipeline. We present the results of our bounding box prediction in subsection two. The third subsection presents the results of the hand pose estimation model used in the last step of the pipeline.

5.1 Hand Segmentation

The proposed model has been applied to $28,440$ images from the HandSeg dataset, as shown in Table 5.1 The pixel-wise accuracy on these images is 0.97 and the mean IoU value is 0.66. This means that on average, 66% pixels of a class are overlapping with the respective labels. Overall, around 81% of pixels are classified correctly. The original implementation of HandSeg yielded a mean IoU values of 0.87, while still performing on a three-class setup. The best mean IoU value reached SegNet. However, SegNet did not yield any useful results with a three-class setup [5].

[2] `https://colab.research.google.com/notebooks/intro.ipynb`

Dataset	Mean IoU
Own implementation	0.66
HandSeg	0.87
SegNet	**0.89**

Table 1: The results of different models used for hand segmentation on the HandSeg dataset. Evaluation for our model and Segnet is performed on a two-class setup, where left and right hands are not distinguished. The mean IoU values for HandSeg and SegNet are taken from [5]

	Amount of boxes
Scenario 1	166
Scenario 2	30
Scenario 3	4

Table 2: The results of bounding box prediction. Scenario 1 describes bounding boxes, which fully covered the hand. In scenario 2, the box did only partially cover the hand and in scenario 3 the bounding box yielded unusable results for further processing.

5.2 Bounding Box

The evaluation of the bounding boxes was performed as follows. We selected 200 random images from the ICVL Hand Posture Dataset. 100 images from test sequence 1 of the dataset and 100 images from test sequence 2. The evaluation was done by two reviewers, whereby every image was iterated through manually, evaluating if the localized hand is usable for the following model as well as counting the numbers of joints inside the bounding box.

The results can be seen in Table 5.2. Overall 83% (166 occasions) of the evaluated bounding boxes did cover the whole hand and 15% (30 occasions) did not cover parts of the hand. Of these 30 cases, ten times one joint was missing, ten times two joints were missing, nine times three joints were missing and one time five joints were missing. Thus, in average 2.07 joints are missing when the box did not cover the hand as a whole. From these 200 bounding boxes, 2% could not be used for further processing. This was the case, when the annotators were not able to recognize the hand by solely evaluating the part of the image inside the bounding box. In these cases, more than five joints were missing.

5.3 Hand Pose Estimation

Our approach for the Hand Pose Estimation achieved a mean 3D Error of 3.46. The results of the initial paper reached a mean 3D Error of 6.177 in mm on the ICVL dataset. But since they did not provide a specific formula of how they converted the pixel values to mm we cannot be sure if we outperformed the presented approach Zhang and Zhang [11]. We calculated the Mean 3D error as follows:

$$d = \sqrt{(x_2 + x_1)^2 + (y_2 + y_1)^2 + (z_2 + z_1)^2}$$

$$\text{mean 3D error} = \frac{1}{2000 \cdot 16} \cdot \sum_{i=1}^{2000 \cdot 16} d_i$$

Hereby, d describes the distance between the two joint coordinates of the predicted values and the labels. The variables x and y describe the plane coordinates and z equals the depth. Afterwards, the mean 3D error of the distances gets calculated. These are 16 joints per hand and 2000 test cases. An example of the predicted joints is depicted in Figure 6, displayed as skeleton.

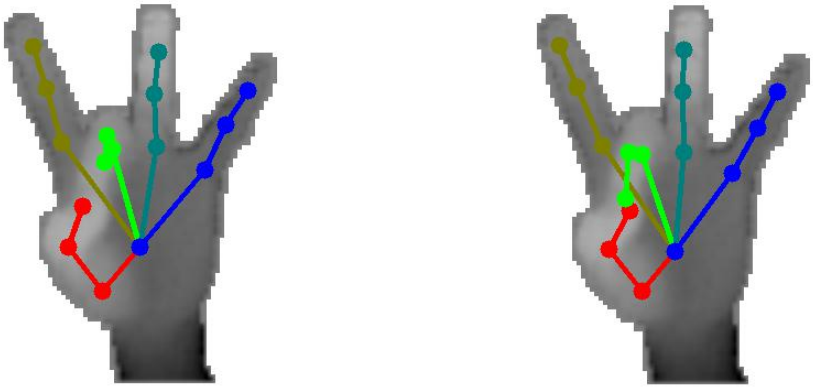

Figure 6: This figure shows a depth image from testing the model. The left image shows the predictions of the model and the right image displays the ground truth. The drawn lines describe the connections between the calculated joints to visualize the prediction results compared to the actual label of the image.

6 Discussion

The performance of the hand segmentation model in step 1 is not as good as the one from the original work. The original model was trained on three different classes, while our implementation was trained with only two classes. We expect that this should make training easier, since the model does not have to differentiate between the left and right hand. One possible reason for this difference could be training time. We trained our hand segmentation model approximately 16 hours, while the original implementation has been trained 28 hours. We regarded the performance as sufficient since the bounding boxes framed the hand as a whole in most of the cases, see Table 5.2. For the next steps of the pipeline this is adequate, since it clears most of the irrelevant features around the hand of the image. Since this denoising reduces the scattering of the calculated center of mass for the second model, this results in a reduction of the 3D error in joint calculation.

The images from the HandSeg dataset mostly contain a person performing different poses and stances in front of the camera. On some depth images it is hard to detect the hands, even for an annotator. This mostly is the case, when the hands are on the same plane with other objects and thus the structure of the hands are hard to detect in the depth field. This is usually not the case when classifying ASL signs, since the user will direct the signs towards the camera. On most of the images from the HandSeg dataset two hands are present. Regarding the classification of ASL signs, it is a valid assumption to expect two hands, since a lot of signs need two hands to be waved. However, in our scenario we only regarded signs waved with a single hand. This leaves room for further implementations to train with a three-class setup on the HandSeg dataset and apply classification on two hands.

For the first and the second model, it was possible to find datasets that would fit the requirements for our models. But for classifying the hand poses to actual items of American Sign Language, the task of finding a labeled dataset came with problems. We could not find a dataset containing an image or a sequence of images of a hand pose representing a word in ASL, including a label describing the meaning of this word in text. There are several datasets containing sequences of images, but either the amount per specific word was single-digit, or there was no label at all. Therefore, to

make an evaluation with a representative result, creating a custom dataset is a necessity. Since the generation of a dataset is a lot of work, it was not possible for us to realize it within the framework of this project. This problem not only existed for the task of evaluation. It also restricted us in designing the creation of step 4 of the pipeline, the Classification of American Sign Language. Since the specifications were to develop an approach based on supervised learning, it was not possible for us to realize the first plan of building a convolutional based approach for step 4, where the input consists of the joints of the hand. An alternative could be an unsupervised or semi-supervised approach. Our approach is sufficient for the task of classifying several words of the ASL as well as the numbers from one to nine. But when trying to increase the number of predictable labels, an approach like ours would be cumbersome and labor intensive. Then, a decision needs to be made what trade-off will be more valuable. Whether if a supervised approach would be beneficial enough to generate a dataset, or if an unsupervised approach would be of more value.

7 Conclusion

We have presented an automated tool for interpreting sign language to enable the communication between ASL users and non-user. As we have shown in our results, the combination of image segmentation and hand posture estimation seems promising by denoising the images, especially around the region of interest. As explained in the discussion, there were some issues regarding the availability of datasets and ambiguity between several hand poses. Some of those could be improved with more effort, but we think that using or creating a labeled dataset and applying a supervised approach could lead to more promising results.

References

[1] B. Kang, S. Tripathi, and T. Q. Nguyen, "Real-time sign language fingerspelling recognition using convolutional neural networks from depth map," in *3rd IAPR Asian Conference on Pattern Recognition, ACPR 2015, Kuala Lumpur, Malaysia, November 3-6, 2015*, IEEE, 2015, pp. 136–140. DOI: 10.1109/ACPR.2015.7486481. [Online]. Available: https://doi.org/10.1109/ACPR.2015.7486481.

[2] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4903–4911.

[3] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan, "3d hand shape and pose estimation from a single rgb image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 833–10 842.

[4] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM transactions on graphics (TOG)*, vol. 28, no. 3, pp. 1–8, 2009.

[5] A. K. Bojja, F. Mueller, S. R. Malireddi, M. Oberweger, V. Lepetit, C. Theobalt, K. M. Yi, and A. Tagliasacchi, "Handseg: An automatically labeled dataset for hand segmentation from depth images," in *16th Conference on Computer and Robot Vision, CRV 2019, Kingston, ON, Canada, May 29-31, 2019*, IEEE, 2019, pp. 151–158. DOI: 10.1109/CRV.2019.00028. [Online]. Available: https://doi.org/10.1109/CRV.2019.00028.

[6] C. Li and K. M. Kitani, "Pixel-level hand detection in ego-centric videos," in *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, IEEE Computer Society, 2013, pp. 3570–3577. DOI: 10.1109/CVPR.2013.458. [Online]. Available: https://doi.org/10.1109/CVPR.2013.458.

[7] S. Bambach, S. Lee, D. J. Crandall, and C. Yu, "Lending A hand: Detecting hands and recognizing activities in complex egocentric interactions," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, IEEE Computer Society, 2015, pp. 1949–1957. DOI: 10.1109/ICCV.2015.226. [Online]. Available: https://doi.org/10.1109/ICCV.2015.226.

[8] D. Tang, H. Jin Chang, A. Tejani, and T.-K. Kim, "Latent regression forest: Structured estimation of 3d articulated hand posture," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3786–3793.

[9] A. Sinha, C. Choi, and K. Ramani, "Deephand: Robust hand pose estimation by completing a matrix imputed with deep features," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4150–4158.

[10] G. Moon, J. Y. Chang, and K. M. Lee, "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map," in *Proceedings of the IEEE conference on computer vision and pattern Recognition*, 2018, pp. 5079–5088.

[11] X. Zhang and F. Zhang, "Pixel-wise regression: 3d hand pose estimation via spatial-form representation and differentiable decoder," *arXiv preprint arXiv:1905.02085*, 2019.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 1106–1114. [Online]. Available: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

[13] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4724–4732.

[14] L. Sigal and M. J. Black, "Measure locally, reason globally: Occlusion-sensitive articulated pose estimation," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, IEEE, vol. 2, 2006, pp. 2041–2048.

[15] S. Yuan, G. Garcia-Hernando, B. Stenger, G. Moon, J. Y. Chang, K. M. Lee, P. Molchanov, J. Kautz, S. Honari, L. Ge, *et al.*, "Depth-based 3d hand pose estimation: From current achievements to future goals," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2636–2645.