Recognizing Italian Sign Gestures with Bidirectional Neural Networks

Stephan Artmann* and Christoph Glanzer*

ABSTRACT

During a project at ETH Zürich, we have implemented a neural network in TensorFlow which recognizes Italian sign gestures from short video clips. Using a bidirectional neural network architecture we were able to beat the *hard*-baseline with our model, reaching a correct classification rate of 82.8% on test data.

This report will give a detailed description of our model architecture. *Unfortunately, for reasons of copyright, we are unable to publish most of our code in this repository.*

1 DESCRIPTION OF THE TASK

The data is a modified version of the *ChaLearn Multimodal Gesture Recognition dataset*. Each sample consists of a video clip, depth data, a segmentation mask, and skeletal information (cf. Figure 1). Data is provided in the TFRecord file format. The resolution of the video clip, depth information and the segmentation mask is 80×80 pixels per frame. The video clips are of different length. The goal is to correctly classify 20 different sign gestures.

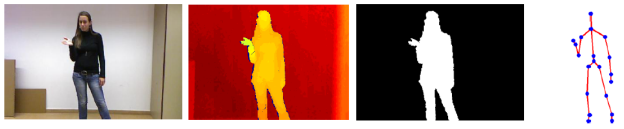

Figure 1: Sample frame from a video clip. The data includes depth data, a segmentation mask, and skeletal information (Image source: Sergio Escalera, Computer Vision Center and University of Barcelona).

2 OVERVIEW

The architecture of our model is based on the work of Pigou et al. [2]. The model we took as a baseline is the bidirectional CNN-RNN depicted in Figure 2.

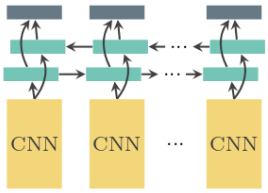

Figure 2: The bidirectional CNN-RNN which forms the basis of our model, cf. [2, Figure 1c].

*Authors are listed in alphabetical order as they contributed equally.

Due to the fact that Pigou et al. have a different, higher-resolution dataset and as a consequence of our observations, we have modified this model quite significantly. The three main ideas which we borrow from this architecture are the bidirectional setup, the design of the CNN in which two convolutional layers are stacked before performing a max-pooling step and the regularization techniques employed. However, we did not implement the peephole connections visible in Figure 2 as we observed that one bidirectional layer was sufficient. Furthermore, our model has a significantly lower number of variables, especially due to smaller dense layers.
One of the main challenges of this project is the prevention of overfitting. We have achieved this by implementing various regularization techniques which we will describe in detail below. Furthermore, we have identified some invalid (wrongly clipped) videos in the training data, cf. Section 5.[1]

3 DETAILED MODEL ARCHITECTURE

The data is fed into the model in batches of 8 videos each.[2] Every frame of each video is fed into a convolutional neural network. This CNN admits shared weights, i.e., every frame is fed into the same CNN. The architecture of the CNN is depicted in Figure 4. The filter size of each convolutional layer is $3x3$, where we use same padding. We use leaky ReLUs after each convolutional layer with a value of $\alpha = 0.3$. The max-pooling layers use a pooling size of $2x2$ and a striding of 2 in each direction. The output is fed into a dropout layer ($p = 0.5$) and subsequently into a dense layer, which again uses leaky ReLUs with $\alpha = 0.3$. The latter has an output size of 128 units.[3]

Next, we feed the output of the CNN into a bidirectional neural network with one layer. We employ long short-term memory units [1] with 256 hidden, resp. output units. The output is fed into a dropout layer ($p = 0.5$) and a dense layer (leaky ReLUs, $\alpha = 0.3$) with an output size of 20 units generating the logit values. The loss is calculated as an average across all video frames. The model's prediction of the gesture label is the argmax of the average across all frames.

4 MODALITIES EMPLOYED

The data consists of RGB, depth, segmentation and skeletal data. The skeletal data is stored as coordinates of the joints. In a preprocessing step of the data, we convert the skeletal data into RGB images, where the joint locations are stored as red dots and the connections between those joints are stored as blue lines. Furthermore, to make use of the segmentation data, we multiply it with the RGB, resp. depth modality to 'cut out' the person visible in the video. This ensures that no unnecessary data is fed to the model. We then

[1] We wish to emphasize that our data is a modified version of the *original ChaLearn dataset*.
[2] The small batch-size is due to memory limitations on the GPU.
[3] When using two dense layers of 2048 units each as suggested in [2], we obtained significantly worse results. We assume that this stems from the fact that the dataset used in their paper has a higher resolution.

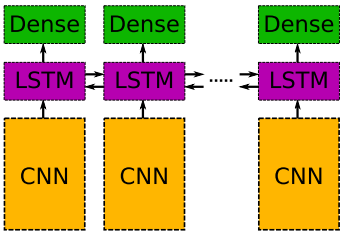

Figure 3: Graphical representation of our model.

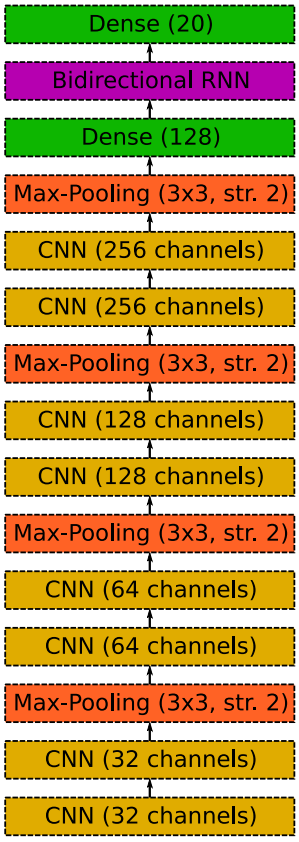

Figure 4: Graphical representation of the convolutional architecture of our model.

feed data consisting of 7 channels to the model: The first three channels are the cut-out RGB data. Channel four is the cut-out depth data. The last three channels contain the skeletal data as RGB image.

Figure 5 below was generated with a debugging script. The first row of images shows the normalized and regularized RGB, depth and segmentation data. The second row of images shows the corresponding cut-out RGB and depth data as well as the skeletal information.

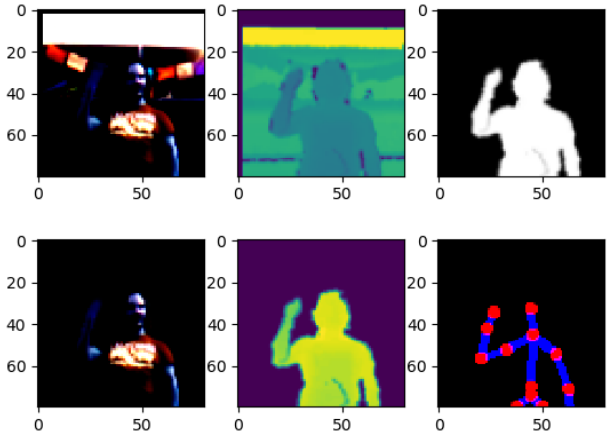

Figure 5: Modalities used in our model.

5 REGULARIZATION

The main challenge of this project is to lower the generalization error. To this end, we have implemented the following regularization techniques, some of which are also used by [2].

- Cropping: It turns out that many videos are badly cut and still contain gestures from the previous or next clip. Furthermore, the gestures are not always in the approximate middle of the video. To make up for this, we randomly crop up to 5 frames from the beginning and from the end of the video.[4]
- Random rotation of each video by $[-\pi/16, \pi/16]$ radians.
- Random translation of each video by $[-10, 10]$ pixels horizontally and $[-5, 5]$ pixels vertically.
- Randomly changing the brightness of each video by up to 30%.
- Resizing each video to a size of 82%-118%.
- Dropout with $p = 0.5$ before each Dense layer.
- Usage of leaky ReLUs with $\alpha = 0.3$ in the CNN and dense layers.

We observed that performing L_2-regularization does not improve the results. Furthermore, flipping does not improve the results. We believe that the latter is due to the fact that many gestures are already performed in a 'flipped' manner.

Early stopping: We have created a python script that converts the data to .gif files. It turns out that the quality of the validation data is significantly worse compared to the training or test data. Many clips contain multiple gestures or badly recognizable gestures. This matches our observations: When training on both the training and validation data, the results are surprisingly worse (by 1-2%)! Therefore, we have not implemented early stopping in our final submission.

REFERENCES

[1] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[2] Lionel Pigou, Aäron Oord, Sander Dieleman, Mieke Herreweghe, and Joni Dambre. 2018. Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video. *International Journal of Computer Vision* 126, 2-4 (2018), 430–439. https://doi.org/10.1007/s11263-016-0957-7

[4] Cutting off more than 5 frames from the beginning / end decreases the test accuracy.