Machine Perception Project 1: Dynamic Gesture Recognition

Nicolas Küchler
kunicola@student.ethz.ch

Yoel Zweig
zweigy@student.ethz.ch

ABSTRACT

Deep neural networks have shown impressive results in the area of image classification, sometimes even outperforming humans. At the moment, the performance in video classification is not on the same level. This is due to the additional temporal dimension which makes the problem much harder. In this work, we tackle the task of classifying short video clips of 20 different Italian sign gestures using a combination of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). The predictions of three networks with identical base architectures but slightly different configurations are combined in a majority voting ensemble to further improve accuracy. Moreover, a variety of data augmentation and regularization techniques are used to prevent overfitting.

1 INTRODUCTION

The goal of this project is to create a model capable of distinguishing 20 Italian sign gestures. The network is trained on a customized version of the ChaLearn dataset[1], which provides RGB, depth, segmentation mask and skeletal information. The dataset contains 9961 video clips, with 5722 in the training set, 1765 in the validation set and 2174 in the test set. Each clip has a length of 50-150 frames with a resolution of 80x80 pixels.

Since the dataset is relatively small for a task of this complexity, there is a danger of overfitting the network to the training data. To counteract this problem, we make heavy use of data augmentation and regularization techniques.

The high-level idea of our approach is to use a CNN to project each video frame to a low dimensional feature space and then use a bidirectional RNN with LSTM cells[2] to capture temporal dependencies of the gestures. Finally, three such models are combined in an ensemble to improve classification accuracy.

2 DATA PRE-PROCESSING AND AUGMENTATION

The RGB input video is normalized to zero-mean and unit variance. Since the background does not contain any relevant information, we apply the segmentation mask to every RGB frame of the video. This results in a black background for all RGB frames. Instead of using the whole clip, we extract a random 64-frame subsequence every epoch. This is inspired by [3] because a gesture is generally only about 20-50 frames long. Additionally, this generates more diverse training data.

At the start of every epoch, the data augmentation techniques listed below are applied to the training set using parameters chosen uniformly at random from an interval. As a result, the network always receives slightly different samples which helps generalization.

We apply temporal augmentation by sampling a factor in the interval [0.8, 1.2] and slow down respectively speedup the video by this factor. Additionally, we perform color augmentation by randomly perturbing the RGB channels through adjusting brightness [-0.2, 0.2], changing saturation [0.6, 1.6] and modifying contrast [0.7, 1] using the functions from tf.image. Moreover, each sample is flipped horizontally with a probability of 0.5. Finally, every video is randomly cropped and rescaled such that at least of 90% of both height and width remain in the video. This results in both a translation and a zoom.

3 MODEL

After online pre-processing and data augmentation, the videos are fed into a CNN to extract spatial features from each frame. Afterwards, the spatial features are fed into a RNN to extract temporal information. Finally, a dense layer with a softmax activation function is used to calculate the class probabilities from the output of the RNN. Figure 1 gives an overview over the complete architecture.

3.1 CNN

The CNN architecture consists of five blocks, each composed of a convolutional layer with a leaky ReLU (alpha=0.3) activation function followed by a max pooling layer. All blocks use 3x3 convolution kernels and 2x2 max pooling with a stride of 2. Starting with 16 filters, the amount of filters doubles in every successive block. A single dense layer of size 512 concludes the CNN model. In the dense layer, we use dropout in the range of [0.4, 0.5] to reduce overfitting.

The overall architecture can be summarized in the following way: C(16)-P-C(32)-P-C(64)-P-C(128)-P-C(256)-P-D(512)

3.2 RNN

We use a bidirectional LSTM architecture with 512 units to extract temporal information from the CNN output. The forward and backward directions of the RNN are combined by adding the output for each timestep together.

A dense layer with dropout in the range of [0.4, 0.5] projects the RNN output of every time step to the 20 classes. The same architecture was successfully used in [3].

4 TRAINING

The loss is calculated by using the same sequence label for each time-step of a video clip but excluding a warm-up of 16 frames in both directions. The exclusion of the warm-up is motivated by the fact that a few frames are insufficient to recognize the gestures and therefore it is better to ignore the RNN output of the first frames in both directions [3].

In order to combat the exploding gradient problem, we clip gradients with a norm larger than a certain value.

The cross-entropy loss is minimized using the Adam optimizer with a learning rate of 0.0005 and a batch size of 16. The learning

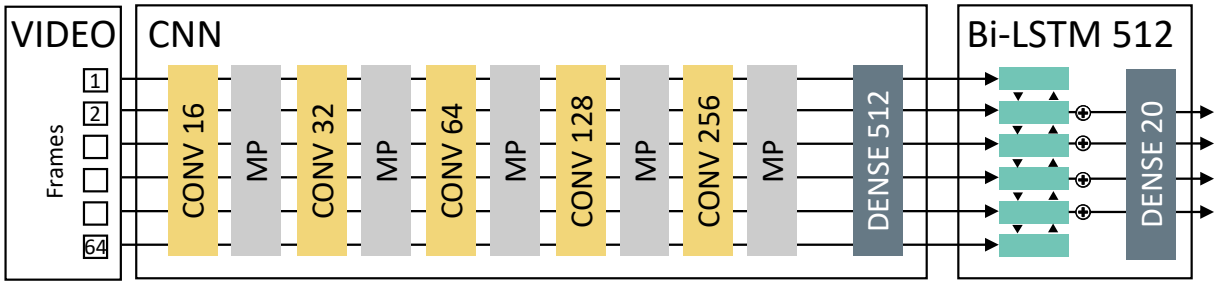

Figure 1: Model Architecture: Frames of a video are fed separately through a CNN to extract spatial features. Afterwards, the bidirectional LSTM is used to infer temporal dependencies. Only the middle 32 frames are used for prediction.

rate has an exponential decay rate of 0.97 every 500 steps. We trained for a minimum of 28 epochs and implemented early stopping based on validation accuracy. The models were trained on the ETH Leonhard cluster using a single GeForce GTX 1080 graphics card for approximately 4 hours.

As described in section 2, we only train the network on subclips with 64 frames. In evaluation mode, we ensure that the whole video length is considered for the prediction using a sliding window approach. For every test sample, 16 evenly spaced subsequences of 64 frames are extracted and fed separately to the model. The 16 separate predictions are combined using an ensemble approach by adding all logits.

5 RESULTS

The final and best submission is a majority vote ensemble of the three best individual submissions. In case of a tie, the prediction of the model with the best individual performance (part 2) is used as a tie-breaker.

Table 1: Experiment Configuration: Differences

Configuration	Part 1	Part 2	Part 3
Gradient Clipping Norm	1	10	10
Training on Validation Set	true	false	false
CNN Dropout	0.5	0.5	0.4
LSTM Dropout	0.4	0.5	0.4
LSTM Layers	1	2	1

The model architecture and hyperparameters of the three different parts of the ensemble are as described in section 3 but with a few configuration differences highlighted in table 1.

Table 2 summarizes the experimental results of the final submission and compares them to the provided baselines.

All individual models are better than the easy baseline but only the model with two LSTM layers manages to marginally beat the hard baseline on the public test set. Combining the three models in an ensemble approach boosts the performance well above the hard baseline. We only performed a small ensemble of three similar models. However, our results suggest that already with these small configuration changes and the randomness in the training process there is enough variety such that the classifier can benefit from the ensemble approach.

Table 2: Experimental Results

Experiment	Valid Accuracy	Test Accuracy
Sample Baseline	-	0.4765
Easy Baseline	-	0.7553
Hard Baseline	-	0.8326
Part 1: Train on Valid	-	0.8132
Part 2: Stack LSTM	0.6992	0.8399
Part 3: Less Dropout	0.6980	0.8142
Vote Ensemble	-	**0.8546**

Another observation which was consistent across all experiments is that the validation set poses a much harder challenge than the public part of the test set. The accuracy scores are on average a bit more than 10% lower on the validation set.

It should be noted that the final submission has been trained using only RGB data and the segmentation mask. We experimented with different approaches to make use of depth and skeleton data but failed to improve accuracy. More information on the other experiments can be found in appendix A.

6 CONCLUSIONS

We have successfully combined a CNN and a RNN for spatial- and temporal feature extraction. Using this approach, we have built a classifier for Italian sign gestures from videos and demonstrated its effectiveness on the customized ChaLearn dataset[1].

The problem of overfitting due to the small size of the training set compared to the large dimensionality of the data remains. Nevertheless, we confirm the results of [3] that data augmentation is integral in combating this.

We have shown the usefulness of building an ensemble of multiple models to boost performance significantly. This suggests that in future work it might be beneficial to build an ensemble with more than three classifiers. Further exploration is needed on what kind of model differences are helpful in such an ensemble. A wide range of possibilities exists such as training on different training-validation splits, using different architectures or using different input data.

Even though the approaches described in appendix A did not yield improved performances in our experiments individually, some of them could be further tuned and prove beneficial in an ensemble.

A FURTHER EXPERIMENTS

During the work on the project, we explored a few directions which did not lead to better results, but we list them here for completeness.

A.1 Larger CNN

The use of a larger CNN similar to the one used in [3], [4] did not result in a better performance. The network architecture we evaluated can be described in the following way: C(16)-C(16)-P-C(32)-C(32)-P-C(64)-C(64)-P-C(128)-C(128)-P-C(256)-C(256)-P-D(512).

A.2 Temporal and Spatial Convolutions

We could not reproduce the results from [3] with the spatial- and temporal (2+1)D-convolution architecture due to convergence problems.

The model converged using a smaller version of the architecture with only one instead of two (2+1)D-convolutional layers per block. However, the temporal convolutions did not appear to benefit the model in our experiments compared to the CNN architecture described in section 3.

A.3 Depth

Using RGB-Depth input as in [3] resulted in covergence issues. These issues were resolved by introducing batch normalization after every convolutional layer. However, the overall performance did not benefit from the additional channel.

A.4 Hands CNN

Studying the confusion matrix during the training process has shown that the model struggles sometimes to discriminate between gestures with similar arm movement but different hand shapes.

To support the network we use the skeleton information to extract two 16x16 patches centered around the two hands and feed them to a small CNN. The resulting hand features are concatenated with the features from the RGB image and fed to the bidirectional LSTM.

Even though training the model only on the hands showed promising results, the anticipated increase in accuracy did not occur when fusing the features of the hand model with the RGB features.

A.5 Additional Augmentation

Inspired by [3] we implemented random translation, rotation and shearing as additional data augmentations. These additional augmentations did not boost performance but slowed down the training process, since they are computationally expensive.

A.6 Concat Bidirectional Output

Instead of adding the outputs from the forward and backward LSTM frame wise, we tried to concatenate them. In theory, this should make the model more powerful. However, in our experiments, this setup did not converge.

A.7 Joint Trajectory Maps

Joint Trajectory Maps is a method to encode spatio-temporal information carried in sequences of skeleton data into 2D-images [5]. Spatial information of the joint is captured by the dot position in the image and the temporal information is captured by the dot color.

We create a separate joint trajectory map for hand, elbow, shoulder, spine and head of every video clip. The resulting maps are concatenated and fed into a standard CNN for classification.

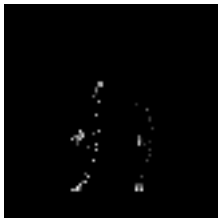

Figure 2: Example of a joint trajectory map of two hands

A problem of the approach is that the shape of some body parts cannot be fully represented by the given skeletal data (e.g. hand shape). Still, the general approach seems to hold potential in particular when considering that the problem of video classification can be reduced to a problem of image classification.

ACKNOWLEDGMENTS

The authors would like to thank Emre Aksan for the well-structured template code and the Machine Perception team for organizing the project.

REFERENCES

[1] Segio Escalera, Xavier Baró, Hugo Jair Escalante, and Isabelle Guyon. 2017. ChaLearn Looking at People: A Review of Events and Resources. (2017). https://arxiv.org/abs/1701.02664

[2] Sepp Hochreiter and Jürgen Schmidhuber. 1997. LONG-SHORT TERM MEMORY. (1997). https://www.bioinf.jku.at/publications/older/2604.pdf

[3] Lionel Pigou, Aäron van den Oord, Sander Dieleman, Mieke Van Herreweghe, and Joni Dambre. 2016. Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video. (2016). https://arxiv.org/abs/1506.01911

[4] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

[5] Pichao Wang, Zhaoyang Li, Yonghong Hou, and Wanqing Li. 2016. Action Recognition Based on Joint Trajectory Maps UsingConvolutional Neural Networks. (2016). https://dl.acm.org/citation.cfm?id=2967191