Machine Perception 2018 Report - Project 1

Frédéric Debraine
Robotics, Systems and Control Master Student
fdebrain@student.ethz.ch

ABSTRACT

Gesture recognition is an active field of computer vision with a wide range of applications such as virtual reality, consumer electronics and automation. It has been a particularly dynamic research area for the past few years with the rise of deep learning and the progress of hardware capabilities. Furthermore, new powerful algorithms using neural networks have shown tremendous improvements in accuracy compared to traditional machine learning methods. In this project report, we propose a re-implementation of an end-to-end neural network architecture inspired from multiple recent research papers. Furthermore, we evaluate our model on the ChaLearn dataset and achieve results close to state-of-the-art.

1 INTRODUCTION

Dynamic gesture recognition is the task of classifying short sequences of images into one or multiple labels, using a limited vocabulary of gesture categories.

1.1 Challenges and motivation

The main challenge behind gesture recognition is that we need a model robust to the way each individual moves. Contrary to spoken language which is build on a rigorous phoneme structure, sign language shows more flexibility in the way "words" are performed. This is not only due to morphological differences between individuals but may also stem from the way the language was learned and the individual's personality. Another challenge in gesture recognition comes from the environment where the movement is perceived. Relevant visual information needs to be extracted despite the presence of noise, illumination changes, background or even cluttering. All these challenges participate in making gesture recognition a stimulating area of research.

1.2 Dataset

We will use the ChaLearn dataset to train, validate and test our model. The data consists in sequences of RGB, depth and segmentation mask images as well as skeletal information. Each sequence is labeled as one of the 20 Italian sign language gesture.

2 RELATED WORK

Early works on gesture recognition often rely on traditional computer vision and machine learning algorithms to classify a sequence of images into a gesture category. Hasanuzzaman et al. [1] proposed to use a skin-like region detector to get the relative positions of the hands and the face and compare it with template images. Sefat and Shahjahan [4] used a similar skin region detector to segment the hands and extracted HoG features before trained a SVM classifier. Despite the good accuracy of these methods, their robustness towards environment changes seems limited.

Most recent works leverage deep learning techniques to create adaptive models for gesture recognition. Neverova et al. [2], winners of ChaLearn 2014 competition, trained a tree-structured deep learning architecture combining different visual modalities inputs. This work paved the way for robust deep learning models and had a major impact in the development of new gesture recognition algorithms.

In our project, we will focus on a state-of-the-art end-to-end deep learning model similar to the one established by Pigou et al. [3]. This model can be seen as a two-stage network. On a first stage, an RGB-D sequence of images is fed to a convolutional neural network to extract visual meaningful features. On a second stage, a bi-directional recurrent neural network captures the temporal information of each sequence and outputs the predicted label. Our re-implementation will however present some fundamental changes as we do not include any temporal convolution and considered different input modalities.

3 METHOD

3.1 Inputs

We consider two modalities out of the four provided as inputs: segmentation and depth images. For each sequence, we normalize and then concatenate the segmentation and depth frames leading to 2-channel input images of resolution 80 by 80 pixels.

3.2 Architecture

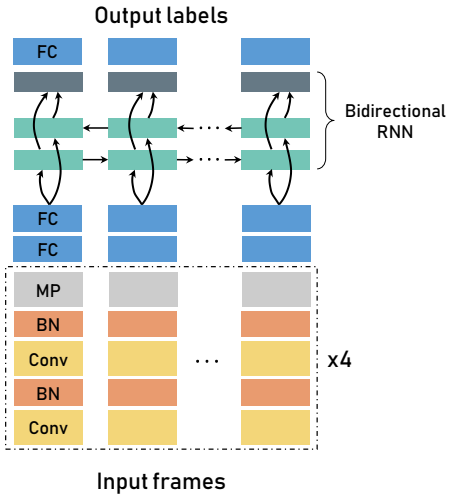

Figure 1: Overview of our our two-stage neural network. MP refers to max-pooling, BN to batch normalization and FC to fully-connected layer.

This project is part of the "Dynamic Gesture Recognition 2018" Kaggle competition

The core idea behind gesture recognition is to be able to extract the motion pattern, i.e. a combination of both spatial and temporal information that uniquely describes the gesture category. To that extent we construct our model architecture in two stages (see Fig. 2). First, a convolutional neural network (CNN) extracts meaningful spatial features from each input images. Second, a recurrent neural network (RNN) captures the temporal dependencies between the previously extracted features. For a given sequence, the full pipeline predicts a gesture category for each input frame, averages all the predictions and classify the whole sequence given the most likely category.

3.2.1 *CNN.*

As a first stage of our model, we consider a CNN composed of four stacks (see Fig. 2). Each stack consists in two convolutional layers using 2x2 filter size and performing max-pooling on non-overlapping 2x2 spatial regions. The CNN ends with two fully connected layers of 2048 units before entering the second stage. The full CNN architecture can be summarized as follows: C(16)-C(16)-P-C(32)-C(32)-P-C(64)-C(64)-P-C(128)-C(128)-P-D(2048)-D(2048) where C(n_c) denotes a convolutional layer with n_c filters, P is a max-pooling layer and D(n_d) a dense layer with n_d units. We use leaky ReLU activation functions in all layers with the parameter α=0.3 and add perform dropout before each fully connected layer with p=0.2. In contrast with [3], we apply batch normalization after each convolutional layer using a decay value of 0.9. This helps the network converge faster and add a slight regularization effect.

3.2.2 *RNN.*

As a second stage, we employ a bidirectional recurrent neural network. This makes the model able to predict a label for each time step of a sequence. The bidirectional RNN is composed of a forward and a backward network, each containing 512 LSTM cells. Each cell's output of the forward network is then averaged with the corresponding cell's output of the backward network leading to a prediction at each time step. Finally the RNN's ouputs are made interpretable using a fully connected layers with 20 units, i.e the total number of labels. The overall sequence label corresponds to the most likely category after averaging w.r.t the temporal axis of each sequence.

3.3 Training

We train our model in a end-to-end manner using a cross-entropy loss function and mini-batch gradient descent with Adam's update rule. All the weights are initialized using a Glorot normal initialization, as we found a noticeable decrease in performance with the random orthogonal initialization used in [3]. We stop the training when obtaining a good compromise between a minimum validation loss and a maximum validation accuracy and adopt the following hyperparameters:

- Learning rate: 10^{-4}
- Exponential decay rate: 0.97
- Mini-batch size: 8

Our final model is trained on 5722 training samples and 1765 validation samples for approximately 14 300 steps, i.e. 20 epochs with a batch size of 8.

4 RESULTS

4.1 Metrics

We use a performance metric based on the accuracy, i.e. the proportion of correct predictions among all the sequences. In [3], we can see that this metric is highly correlated with the Jaccard index, which is typically used for frame-wise gesture recognition. Considering the final task is a sequence-wise classification and not frame-wise, we found the accuracy metric to be more adequate.

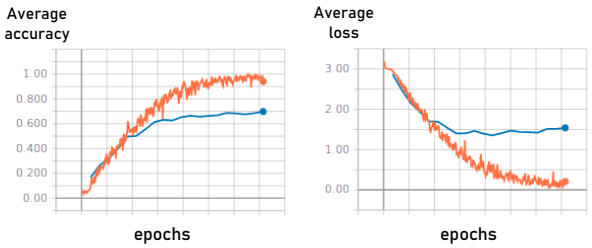

Figure 2: Overview of our trained model's accuracy and loss after 20 epochs. Orange curves are for the training dataset while blue curves are for the validation dataset.

As expected and thanks to batch normalization and dropout, there is not much overfitting during the training (See Fig. **??**). This allows us to train for more epochs and benefit from a higher validation accuracy 1.

Table 1: Summary of our model performances using validation accuracy and loss.

Architecture	Modalities	Epoch	Accuracy	Loss
Our model	Seg+Depth	14300	0.69	1.54

4.2 Number of training epochs

We select the epoch to stop the training based mostly on the validation loss, as it represents a good indicator whether the model is overfitting on the training set or not. Our final model is trained over 20 epochs, i.e the number of time the network sees the full training dataset.

4.3 Effects of choosing different input modalities

Exploiting the depth information gave a good boost to our model performance as is acts as a individual-background extractor and helps differentiate the body of the individual from his hands which are usually closer to the camera. Choosing the segmentation information over the RGB had a slight beneficial effect, telling the model where the individual is located. At the end, we used a combination of segmentation and depth information as input to our model.

4.4 Effects of data augmentation

We tried several combinations of preprocessing and augmentation steps with the current model: cropping, resizing, random temporal padding or adding random rotations, scales and brightness to

the input frames. Despite a clear reduction in overfitting, we were surprised to see these methods show a slight decrease in performance. Preprocessing and data augmentation are a delicate part in a pipeline and require a lot of trials and hyperparameters tuning. We unfortunately didn't manage to find a good combination of these preprocessing techniques and thus didn't include them in our final pipeline.

4.5 Effects of 3D convolutions

We also tried to extract spatio-temporal features directly in the CNN by using 3D convolutions and 3D maxpooling, following the method mentioned in [3]. This lead to a much bigger model, more prone to overfitting and achieving nearly the same performance than our current model. Therefore, we decided to stick with the simpler, yet best performing architecture presented in 3.2.

5 DISCUSSION

Despite our model didn't achieve a performance as high as presented in the reference paper [3], it still performed reasonably well considering the small volume of input data and the lack of an augmentation strategy. Furthermore, Chalearn can be consired a challenging dataset for gesture recognition as it exhibits non uniform backgrounds, difficult illumination settings and individuals performing gesture sometimes almost off camera.

We used several common methods to reduce overfitting on the training data such as dropout, which randomly drops neuron units in the network to avoid the model getting to complex, and batch normalization, which also makes the training faster.

6 CONCLUSION

In this project report, we proposed a slightly revisited implementation of a state-of-the-art two-stage neural network for gesture recognition using segmentation and depth images as inputs. We showed that adding batch normalization after each convolutional layer made the training faster and less prone to overfitting. Even though our model does not reach the performance of [3], it still achieves close to state-of-the-art results and could definitely benefit from an adequate augmentation method as well as using other modalities such as skeletal information that could useful to crop the images w.r.t the body position. Finally

REFERENCES

[1] M. Hasanuzzaman, V. Ampornaramveth, Tao Zhang, M. A. Bhuiyan, Y. Shirai, and H. Ueno. 2004. Real-time Vision-based Gesture Recognition for Human Robot Interaction. In *2004 IEEE International Conference on Robotics and Biomimetics*.

[2] Natalia Neverova, Christian Wolf, Graham W. Taylor, and Florian Nebout. 2015. ModDrop: adaptive multi-modal gesture recognition. *CoRR* (2015).

[3] Lionel Pigou, Aäron van den Oord, Sander Dieleman, Mieke Van Herreweghe, and Joni Dambre. 2018. Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video. *International Journal of Computer Vision* (2018).

[4] Md Syadus Sefat and Md. Shahjahan. 2015. A hand gesture recognition technique from real time video. *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)* (2015).