Data Augmentation of Semantic Embeddings for Skeleton based Zero-Shot Gesture Recognition

David Heiming
Karlsruhe Institute of Technology
uween@student.kit.edu

Hannes Uhl
Karlsruhe Institute of Technology
ujjmv@student.kit.edu

Jonas Linkerhägner
Karlsruhe Institute of Technology
uoega@student.kit.edu

Abstract

Interaction with computer systems is one of the most important topics of the digital age. Interfacing with a system through body movements rather than tactile controls can provide significant advantages. To make that possible, the system needs to reliably detect the performed gestures. Systems using conventional deep learning methods are therefore trained on all possible gestures beforehand. Zero-Shot learning models, on the other hand, aim to also recognize gestures not seen during training when given their labels. The model thus needs to extract information about an unseen gesture's visual appearance from its label. Using typical text embedding modules like BERT, that information will be focused on the semantics of the label rather than its visual characteristics. In this work, we present several forms of data augmentation that can be applied to the semantic embeddings of the class labels in order to increase their visual information content. This approach achieves a significant performance increase for a Zero-Shot gesture recognition model.

1. Introduction

Gesture recognition of videos is a rapidly growing field of research and is becoming an important component of input-device-less control of consumer products such as drones or televisions. While various past works have focused on the classification of gestures known in advance [9, 5], this work deals with gesture recognition using Zero-Shot learning. This approach makes it possible to use unseen gestures (meaning gestures that the model has not seen during training). The user of the product is thus offered the opportunity to expand the command set for controlling the device.

In order to be able to classify samples of an unseen class, a network needs to form an expectation of what the gesture looks like based on its label. This is usually done through the use of text embeddings [2]: Trained on unannotated text data, language embedding models extract meaning from words or sentences by converting them into a semantic embedding vector. After creating a semantic embedding for each class label, it is possible to compare the embeddings of an unseen class with those of the seen classes to find similarities between them. Based on those similarities, the network can construct an expectation of what a sample of that unseen class might look like.

It is quite common to apply data augmentation techniques such as cropping, scaling or flipping to the video input of a network in order to increase the amount of available training samples [11]. However, in Zero-Shot learning there are two different, equally important kinds of training information for each class: visual and semantic. The common data augmentation strategies only make it possible to multiply the amount of visual training data. But the semantic information remains minimal, usually restricted to the simple label of the class. We aim to provide the network more relevant semantic information about the different classes by applying several forms of data augmentation to the semantic embeddings of the class labels.

2. Method

First we need to build a network capable of Zero-Shot learning for gesture recognition. Then we define different forms of data augmentation for the semantic embeddings of the class labels and specify the experimental setting.

2.1. Architecture

The architecture chosen for our experiments largely corresponds to the model presented in [4]. We rebuild its modular architecture using the information published in the paper.

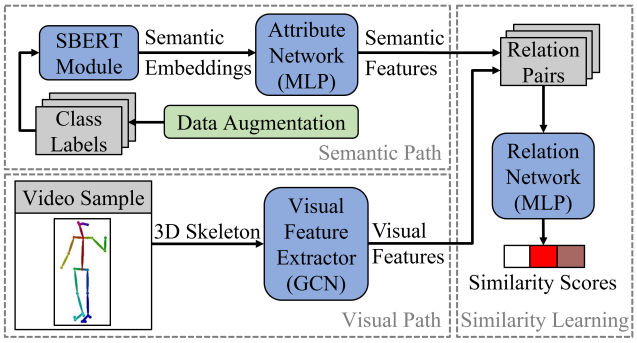

Figure 1. Overview of the network modules.

Certain modules are replaced or slightly modified to fit our specific task. Here, only a brief overview of the functionality is given, explaining how the model tries to solve the Zero-Shot task, and which changes are made. For detailed information on the network modules, which are illustrated in Figure 1, refer to [4, 14, 12, 13].

As visual input we use a temporal series of skeletons, instead of RGB videos to remove unnecessary details such as the background or a person's clothing. Each skeleton is a graph whose nodes represent the person's joints. A full input sample consists of a series of one skeleton graph per frame. Such skeleton data can be obtained from RGB video using a framework like *Openpose* [1]. To perform a visual feature extraction of these input samples, a Graph Convolutional Network (GCN) [14] is used. It consists of 9 spatial temporal graph convolution layers with residual connections. The resulting ouput is a 256-dimensional vector containing the visual features.

Parallel to this visual path, a semantic feature extraction of the vocabulary, i.e. all possible class labels, is performed in two steps. First a *Sentence BERT* (SBERT) module [12] transforms the class labels into semantic embeddings. This is different from the original architecture in [4], where an older *Sent2Vec* module [10] is used. The SBERT module takes a sentence as input, analyzes it and yields two kinds of outputs: a cls-token vector, which is a representation of the entire sentence, and a series of embedding vectors, that each represent one word of the input sentence with its context. A 768-dimensional mean token vector can be created out of this secondary output by applying an attention mask to the series of tokens to combine them into a single one. We use the mean-token output of the SBERT module over the cls-token as our semantic embedding because it resulted in a better performance. In the second step, the attribute network (AN) transforms the semantic embeddings into semantic features by mapping them into the 256-dimensional visual feature space. Compared to its original form in [13], we apply dropout with a factor of 0.5 to the first layer of this multi layer perceptron (MLP).

Finally, the visual and semantic feature outputs are combined by forming relation pairs. Each pair is a concatenation of the visual features of our input sample with the semantic features of one class. These relation pairs are then fed into the relation network (RN) introduced in [13]. The RN applies a similarity metric in order to assess the resemblance of the semantic and visual features within each relation pair. This way, it computes a similarity score for each pair, which symbolizes the input sample's similarity to the corresponding class. Then, the similarity scores are compared to a one-hot vector representing the ground truth class using mean squared error (MSE) loss. In contrast to previous works, this architecture does not use a fixed similarity metric. Instead, the RN is a MLP that learns a deep similarity metric during training, which was introduced and shown to improve performance in [13]. We add an additional linear layer to the RN and apply dropout to the first and second layer with a factor of 0.5.

2.2. Data augmentation

In this section, we present three data augmentation methods for the semantic embeddings of the default class labels provided by the dataset. We apply these methods directly to the labels, because they are more tangible than the abstract embeddings. This still results in an augmentation of the semantic embeddings, since they are created from the labels. The goal of these methods is to increase the visual information content of the semantic embeddings of our gesture classes in order to improve the classification performance. To demonstrate them, we apply each augmentation to the class with the default label "squat down" as an example.

2.2.1 Descriptive labels

In a first step, we provide more visual information by substituting the class labels, which in their original form mostly consist of one or two words, with a complete sentence. We use sentences that give a more precise description of the movements required to perform a particular gesture. This way, the default label "squat down" is manually augmented to create the new descriptive label: "A human crouches down by bending their knees". During training and testing, every default label is replaced by its manually written descriptive counterpart.

2.2.2 Multiple labels per class

We now increase the information content of the semantic embeddings even further by labeling each gesture with several different descriptions. Thus, we manually create additional descriptions that use different wording for each gesture. An example of using three descriptive labels per class is shown in Table 1. Since the network computes a similarity score for each possible label, there are now three times

1:	A human crouches down by bending their knees.
2:	A person is bending their legs to squat down.
3:	Someone crouches down from a standing position.

Table 1. Three descriptive labels for the class "squat down".

Description:	A human crouches down by bending their knees.
Augmentation 1:	A *small* human crouches *duck* down by bending their knees.
Augmentation 2:	A human crouches *fall* down *somewhat* by bending their knees.

Table 2. Descriptive label and two automatic augmentations for "squat down".

as many similarity scores due to the expanded vocabulary. In each iteration of the training process, the ground truth of a sample is randomly selected from one of the three possible labels. During inference, all three possibilities are considered correct if the network predicts one of them for the corresponding sample.

2.2.3 Automatic augmentation

To reduce the manual annotation effort, we now generate additional labels automatically for the multiple labels approach. For this purpose, we use an augmenter from *nlpaug* [8] with the *RoBERTa* language model [7] to insert words into a manually created descriptive label. We do not use word substitutions, since it is often impossible for the automatic text augmentation to find multiple suitable synonyms for specific words. Word deletions are also suboptimal, because removing key words leads to a sentence that does not describe the given action appropriately. An example label set is shown in table 2. One can see, that the reduced manual annotation effort sometimes comes at the cost of generating grammatically incorrect sentences.

2.3. Experiments

In this work, we use the *NTU RGB+D 120* dataset [6], which contains 3D skeleton data for 114,480 samples of 120 different human action classes. To evaluate our model we pick a subset of 40 gestures classes to execute four performance tests: one with the default labels as a baseline, and one per augmentation method. A performance test consists of eight training runs on 35/5 (seen/unseen) splits, which are randomized in such a way that every single class is unseen in exactly one training run.

During a training run, only the weights of the AN and RN modules are adjusted. The visual feature extractor is trained beforehand on the 80 unused classes of the *NTU* dataset to ensure that the unseen gestures have not appeared in the training process at some early stage. The SBERT module has already been trained on large text corpora by *Sentence-Transformers* [12].

We test the performance in two scenarios for each augmentation method: In the ZSL scenario, the model only predicts on the unseen classes, while it predicts on all classes (seen and unseen) in the GZSL scenario. In the latter we measure the accuracy for seen and unseen samples, as well as the harmonic mean, following recent works [3]. In each scenario the results are averaged over the eight individual training runs of a performance test. For default and descriptive labels, we train our network with a batch size of 32, as it was done in the original paper [13]. When using multiple labels, we increase the batch size to 128 and add batch normalization at the input of the RN.

3. Results

All our results are generated following the procedure described in the experiments section. For the multiple labels approach three manually created labels per class are used. The automatic augmentation approach utilizes five labels: one manually created label and four augmented versions. In table 3 one can see the ZSL, seen and unseen accuracies, as well as the harmonic mean. Table 4 displays a more detailed view of the achieved unseen accuracies. It shows the top-1 and top-5 accuracies for our approaches with their standard deviations (std) over the eight splits.

Improvements on the ZSL accuracy, the unseen accuracy and the harmonic mean are achieved using the descriptive labels. The accuracies increase even further with the multiple labels approach. Using automatic augmentation performs worse compared to multiple manually created labels, but it still constitutes a relative 23% increase over using only one descriptive label.

The seen accuracy stays within the same range, only experiencing a marginal increase for the two cases that use multiple labels. This behaviour along with a decrease in unseen accuracy is observed whenever batch normalization is applied to any of our approaches. Therefore it is only applied in the cases where multiple labels are used because they require batch normalization in order for the training to converge.

Table 4 shows that the top-5 accuracies behave similarly to their top-1 counterparts, with the exception of a less severe performance decrease when using automatic augmentations. The standard deviations of the top-1 accuracies are in the same range for all approaches based on the descriptive labels. The standard deviation belonging to the top-5 accuracies decreases for the multiple label aproaches, which indicates a higher prediction consistency.

Augmentation	ZSL	Seen	Unseen	h
Baseline	0.4739	0.8116	0.1067	0.1877
Descriptive	0.5186	0.8104	0.1503	0.2495
Multiple	**0.6558**	0.8283	**0.2182**	**0.3417**
Automatic	0.5865	**0.8290**	0.1856	0.3003

Table 3. ZSL and GZSL results for different approaches.

Augmentation	top-1± std	top-5 ± std
Baseline	0.1067 ± 0.0246	0.5428 ± 0.0840
Descriptive	0.1503 ± 0.0553	0.6460 ± 0.1250
Multiple	$\mathbf{0.2182} \pm 0.0580$	$\mathbf{0.8580} \pm 0.0657$
Automatic	0.1856 ± 0.0499	0.8272 ± 0.0476

Table 4. Unseen top-1 and top-5 accuracies (GZSL).

3.1. Discussion

3.1.1 From default to descriptive labels

The improvement from the use of descriptive labels shows that incorporating more visual information into the semantic embeddings helps the network to find a general relation between the semantic and the visual space. Plainly speaking the network can find more similarities between the class labels. This is important since the expected visual features of an unseen class are determined based on the similarities between its label and the seen labels. One might expect these similarities to also be present in the embeddings of the default labels because SBERT should be able to generate representative embeddings that share characteristics with similar classes. While such similarities are present in the SBERT embeddings, they are not focused on the visual appearance of the gestures. For example, the embeddings of the class labels "sit down" and "drink water" might be somewhat similar, because those words appear together frequently in the large text corpora that SBERT was trained on. Visually however, those classes look vastly different from each other. The embeddings falsely suggest, that a similarity between the classes is there, which is less likely to happen if the embeddings are created from visual descriptions of the actions.

3.1.2 Using multiple labels

When using multiple labels, the idea is somewhat different. The main motivation is that using larger amounts of data is generally a good idea. Here, the descriptions and therefore the embeddings of each sample are chosen randomly among the three possibilities during training. This forces the network to assign a high similarity to all three labels corresponding to a sample, which leads to a more general mapping between the semantic and the visual feature space. The model has to adapt to the greater diversity of the used semantic embeddings. This improved generalization on seen training data then helps the network understand and therefore classify the unseen samples better.

For the methods using multiple labels per class, the batch size during training is increased from 32 to 128. Since the network needs to learn a mapping for a greater amount of classes, increasing the batch size is necessary to find relations between more classes at once. Increasing the batch size does not benefit the single label approaches.

3.1.3 Automatic augmentation

The individual labels for a class are very similar when using automatic augmentation compared to multiple manually created labels, since only a few additional words are inserted for each version. The diversity in the semantic embeddings is therefore less pronounced, which leads to a worse performance. However, compared to the single labels, where the semantic embeddings contain no diversity, the performance is significantly better.

If diversifying the semantic embeddings is the key to improving the performance, one might expect, that generating the additional embeddings by adding random noise to a single embedding could also work. This would obviate the need for a text augmentation module. However, this method does not improve the performance compared to the single label approach when tested on our model. This shows that a specific kind of diversity is needed to get an improvement. Using word insertions clearly provides a suitable diversity, since there is an improvement despite the resulting grammatical errors described in section 2.

4. Conclusion

In this work, we demonstrate the potential of applying data augmentation to the semantic embeddings of a Zero-Shot gesture recognition model. By including more visual information in the class labels and combining multiple descriptions per class we are able to improve the performance of a model based on [4] by a significant margin. The use of automatic text augmentation still leads to a sizable performance gain, while keeping the manual annotation effort low.

Future works might further investigate the following topics: Firstly, generating descriptive sentences from the default labels, e.g. by using methods from Natural Language Processing (NLP), would further reduce the manual annotation effort. Secondly, our methods could be tested on different Zero-Shot architectures to verify our improvements. Finally, different kinds or combinations of automatic text augmentation methods could be evaluated.

With these advances, data augmentation of the semantic embeddings in Zero-Shot learning can prove useful in optimizing the performance of any Zero-Shot approach in the future.

References

[1] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *arXiv:1812.08008*, 2019.

[2] Valter Estevam, Helio Pedrini, and David Menotti. Zero-shot action recognition in videos: A survey. *arXiv:1909.06423*, 2020.

[3] Pranay Gupta, Divyanshu Sharma, and Ravi Kiran Sarvadevabhatla. Syntactically guided generative embeddings for zero-shot skeleton action recognition. *arXiv:2101.11530*, 2021.

[4] Bhavan Jasani and Afshaan Mazagonwalla. Skeleton based zero shot action recognition in joint pose-language semantic space. *arXiv:1911.11344*, 2019.

[5] Okan Köpüklü, Ahmet Gunduz, Neslihan Kose, and Gerhard Rigoll. Real-time hand gesture detection and classification using convolutional neural networks. *arXiv:1901.10323*, 2019.

[6] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C. Kot. Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2684–2701, Oct 2020.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692*, 2019.

[8] Edward Ma. Nlp augmentation. https://github.com/makced ward/nlpaug, 2019.

[9] Zdravko Marinov, Stanka Vasileva, Qing Wang, Constantin Seibold, Jiaming Zhang, and Rainer Stiefelhagen. Pose2drone: A skeleton-pose-based framework for human-drone interaction. *arXiv:2105.13204*, 2021.

[10] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

[11] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv:1712.04621*, 2017.

[12] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv:1908.10084*, 2019.

[13] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. *arXiv:1711.06025*, 2018.

[14] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *arXiv:1801.07455*, 2018.