Salami, Dariush; Sigg, Stephan

Zero-Shot Motion Pattern Recognition from 4D Point-Clouds

Published in:

DOI:
10.1109/MLSP52302.2021.9596375

Published: 15/11/2021

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
ABSTRACT
We address a timely and relevant problem in signal processing: The recognition of patterns from spatial data in motion through a zero-shot learning scenario. We introduce a neural network architecture based on Siamese networks to recognize unseen classes of motion patterns. The approach uses a graph-based technique to achieve permutation invariance and also encodes moving point clouds into a representation space in a computationally efficient way. We evaluated the model on an open dataset with twenty-one gestures. The model outperforms state-of-the-art architectures with a considerable margin in four different settings in terms of accuracy while reducing the computational complexity up to 60 times.

Index Terms— zero-shot learning, Siamese network, triplet, contrastive, 4D point cloud, gesture recognition

1. INTRODUCTION
Zero-shot learning refers to being able to distinguish new classes that have not been present in the training data [1]. Essentially, zero-shot learning distinguishes samples of known classes from those belonging to non-known classes. In most cases, the cross-modal mapping between an input and a class embedding space with discriminative losses is learned, while few generative models exist too [2]. Zero-shot learning is an important academic challenge with significant practical relevance since the generation of training data is expensive and interaction may change over time for various reasons. This is especially true in human activity recognition [3].

We address zero-shot learning in human activity recognition utilizing 4D point-cloud data. Such temporal point-cloud data is found, for instance, in gesture-based interfaces [4] for robots [5], in-cabin vehicular scenarios, gaming [6], and Virtual/Augmented Reality (AR/VR) [7]. Point-clouds can be acquired, for example, from mmWave Radars, RGB-D cameras, or LiDARs. They constitute unstructured sets of permutation-invariant points, often in a three-dimensional space. In particular, the evolution of point patterns is interpreted, which requires temporal data processing.

We develop a neural network architecture to learn features of temporal point-clouds, and further recognize unseen gestures in the inference phase through zero-shot learning. Under the zero-shot assumption, the network estimates for pairs of possibly unseen gesture instances, whether they belong to the same class. We further propose a computationally light-weight graph-based point-cloud encoder, P-GEN (Point-cloud Gesture ENcoder), which can be implemented on constrained devices in real-world scenarios. While most existing models extract the spatial features from each frame and then fuse them through time to recognize gestures, we reflect the temporal dependency in a graph structure and process it using graph convolutional neural networks to reduce the computational load. Our contributions are:
1. We propose a light-weight graph-based temporal point-cloud encoder, P-GEN, for gesture recognition
2. We introduce a zero-shot gesture processor based on P-GEN to recognize unseen classes of gestures
3. We publish the trained models and source code to allow others to reproduce and build upon our results

2. RELATED WORK
A temporal point-cloud is a sequence of frames through time each of which consists of an unordered set of points in space. Gesture recognition using temporal point-clouds falls into three categories of techniques: projection, voxelization, and direct point-cloud processing. In projection-based approaches, usually a Convolutional Neural Network (CNN) based architecture is applied on different projected views of a gesture [8], to extract a fine-grained set of features and classify the gesture. But these approaches suffer from information loss due to the projection as well as high computational complexity to process different views.

In the second group of approaches, the point-cloud representation is transformed into a voxel-based one (voxels in 3D space are equivalent to pixels in 2D space). Since mid-air gesture recognition requires temporal feature extraction, voxel-based approaches either use 3D CNNs [8, 9] or Recurrent CNNs (R-CNNs) [10] to process spatio-temporal features of gestures. Information loss due to the voxelization process

---

1https://version.aalto.fi/gitlab/ambientintelligence/one-shot-gesture-recognition
as well as the cubic increase in the computational complexity are two major drawbacks of this group.

Finally, direct point-cloud processing gesture recognition models are another group of approaches which attempt to classify gestures directly from point-clouds [11, 4, 12]. These models have addressed the need for data conversion and its computational-complexity and information loss problems. However, majority of the mentioned models extract spatial features and then fuse them to capture temporal evolution causing a high computational-complexity. Furthermore, the complexity of novel applications in gesture recognition requires flexible models to generalize well on unseen classes of gestures, trained on a limited number of classes. Although there are few works in zero-shot learning on shape classification from static point-clouds [13, 14], gesture recognition from temporal point-clouds still needs to be investigated more. In this work we propose a novel Siamese network architecture for zero-shot learning from temporal point-clouds to recognize unseen classes of gestures.

3. ZERO-SHOT GESTURE RECOGNITION

This section explains our model to perform zero-shot learning for 4D point-cloud gesture data, covering siamese networks and their training paradigms, as well as our encoder for gestures to a latent space.

3.1. Siamese Network

A Siamese neural network [15] is an architecture with multiple identical encoder components $g_i : X \rightarrow Y, i \in \{1, 2, \ldots \}$, each of which is responsible for transforming an input sample $x_i$ to a latent space representation $r_i$. The encoders have the exact same architecture and weights and are updated simultaneously during the back-propagation phase. This network architecture aims to describe the similarity between a pair of instances. An overall structure of a Siamese network is shown in Fig. 1. Because a siamese network is trained to separate dissimilar pairs in the latent space, it learns discriminative features which are capable to realize zero-shot learning [15]. The output is a similarity metric, which is also computed for pairs of samples from a distribution that is unknown to the model. The loss functions that are often used to train Siamese networks are contrastive loss and triplet loss.

3.1.1. Contrastive Loss

Contrastive loss $L_c$ is a Euclidean-based similarity metric.

$$L_c = y \cdot D(r_i, r_j) + (1 - y) \cdot \max(0, m - D(r_i, r_j))$$

Here, $y$ determines whether the gestures $i$ and $j$ belong to the same class ($y = 1$) or not ($y = 0$); $D(r_i, r_j)$ is the Euclidean distance between the encoded representations of gestures $i$ and $j$. $m$ is a margin to prevent the trivial solution with $L_c = 0$, by configuring the hidden representations for all gestures onto a single point.

3.1.2. Triplet Loss

Triplet loss minimizes the distance between an anchor gesture and a positive (same class) while increasing the distance to a negative (different class) gesture.

$$L_t = \max(0, D(r_a, r_p) - D(r_a, r_n) + m)$$

$r_a$ is the encoded anchor gesture, $r_p$ is the encoded positive gesture, $r_n$ is the encoded negative gesture, and $m$ is the margin to prevent a trivial solution.

3.2. Point-cloud Encoder

To perform zero-shot learning using a siamese network, we transform the input sample to a latent representation with considerably lower dimension compared to that of the input. Via a graph-based temporal point-cloud encoder module, P-GEN, we convert the 4D temporal point-cloud into a 2D representation. The two point-cloud processing steps of P-GEN are graph creation and graph processing.

3.2.1. Graph Creation

A temporal point-cloud is an unordered set of points in space, including the temporal dimension. For an $F$-dimensional point-cloud $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^F$ with $n$ points, a directed graph $G = (V, E)$ with $V = 1, \ldots, n$ and $E \subseteq V \times V$, is derived. In particular, we clustered groups of $N$ points each along the temporal axis into frames and add for each frame $H_j$, $k$ directed edges between to the $k$ nearest neighbours of every point $x_i \in H_j$ and all points $x_j \in H_{j+1}$. Note that the authors of [16] had proposed to apply a k-NN on unprocessed static point clouds. As shown in the example in Fig. 3a, this approach yields low connectivity and no major direction of movement across the points. Instead, our construction (Fig. 3b) yields direction of arrows which reflects the temporal evolution of the gesture.
3.2.2. Graph Processing

In the graph processing phase, first, a set of edge features as $e_{ij} = h_\Theta(x_i, x_j)$ with $h_\Theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$ are calculated using a message passing scheme. $h_\Theta$ is implemented using a non-linear function with a set of learnable parameters $\Theta$:

$$h_\Theta(x_i, x_j) = h_{\Theta_0}(x_i, x_j - x_i),$$  \hspace{1cm} (3)

where $h_{\Theta_0}$ is a concatenation function with a set of learnable parameters $\Theta_0$ implemented as a shared Multi-Layer Perceptron (MLP). Finally, a max aggregation function is applied feature-wise to update the representation of each point:

$$x'_i = \max_{j \in (i,j) \in E} h_\Theta(x_i, x_j)$$  \hspace{1cm} (4)

Summarizing, an $F$-dimensional point-cloud with $n$ points yields an $F'$-dimensional one with equal number of points. The architecture of P-GEN is shown in Fig. 2. The initial point-cloud of a gesture (a) is unstructured. By applying 4D k-NN (b), a graph is formed which reflects the temporal evolution of the gesture. In (c), edge features are calculated through Eq. (3) using concatenation and a MLP. Through max aggregation (Eq. (4)) the updated representation of each point $x_i$ is computed and a fully connected layer, followed by a max pooling layer are applied to extract the representation for the whole graph. Finally, we obtain a fine-grained 2D representation for each gesture using an MLP.

4. IMPLEMENTATION

We discuss the implementation of preprocessing and data augmentation to increase the generalizability of the model as well as the framework and optimization approach to train the model. Finally, we demonstrate how an online scheme for choosing gesture pairs may improve the model performance.

4.1. Preprocessing and Data Augmentation

To increase the stability of the model, we normalize the frames and number of points per frame for all gestures (cf. [11]). In particular, we assign $n/f$ points to each frame, yielding fixed number of frames, $f$ for gestures with $n$ points. We empirically set $f$ to 32 to in all our experiments.

Then, to fix the number of points in each frame without loosing the spatial structure of the gesture, we use the density-based approach introduced in [17]. To up-sample the frames, Agglomerative Hierarchical Clustering (AHC) is applied and the centroids of the clusters are added to the frame to hold the desired number of points. For down-sampling, the $k$-means algorithm with $k$ as desired number of points is applied and
centroids are added as points to the frame. To improve the generalizability of the model, we apply an on-the-fly data augmentation in the training phase for each batch. In particular, we apply random translation of gestures up to 10cm, random scaling between with a factor between 0.8 and 1.2, random point-wise translation based on a Gaussian distribution with \( \mu = 0 \) and \( \sigma = 0.01 \), random clipping of 0.03m, and random shuffling of points in each frame.

### 4.2. Model Training Process

We implemented the model with PyTorch Geometric [18], a graph-based neural network framework and trained the model using a server with 64GB of RAM and a Tesla V100 16GB GPU. Furthermore, we utilized an early stopping mechanism to save the best performing model if no improvement in the loss value of the validation set is observed after 100 epochs. Finally, we optimized the model with Adam Optimizer, using a step-decay mechanism to decrease the learning rate for faster convergence. We set the drop rate after every \( b \) epochs, \( r \) as 5, and \( e \):

\[
\alpha = \alpha_0 \cdot d_t^\left(\frac{\lfloor e \rfloor}{e}\right)
\]

where, \( \alpha(0) \) is the (initial) learning rate, \( d_t \) the drop rate after every \( e \) epochs, \( e \) the current epoch and \( \lfloor \cdot \rfloor \) the floor operator. We vary \( \alpha_0 \) as 0.001, \( d_r \) as 0.5, and \( e_r \) as 20.

### 4.3. Pair Selection Strategies

We now discuss strategies to select pairs for the Siamese network. The effect of each strategy is analyzed in Section 5.

#### 4.3.1. Random Pair

The simplest strategy to select a pair for contrastive loss and negative/positive pairs for a given anchor gesture in the triplet loss is random. We randomly choose pairs for each gesture in the training set. On the downside, after a few epochs, it becomes less probable to select pairs for which we have a positive loss value in Eq. 1 and Eq. 2. Consequently, it will take a long time for the model to converge.

#### 4.3.2. Hardest Pair For Contrastive Loss

To solve this, we can assign a pair which maximizes the loss function in contrastive loss. However, it is infeasible to calculate the loss for every possible training pair. Instead, we select the hardest pair in a mini-batch instead of finding it from the whole training data [19]. Consequently, in a mini-batch, we sample 25 random gestures from each class, then find the hardest pair from these to pair up with each gesture.

#### 4.3.3. Semi-Hard Triplets

Following a similar approach, we can find the hardest negative/positive pair for a given anchor encoded gesture from a mini-batch. As discussed in [19], this can lead to local minima or even a collapsed model. An alternative approach (semi-hard triplets) is to select a negative gesture that satisfies the following inequality for a given anchor:

\[
D(r_a, r_p) < D(r_a, r_n),
\]

(5)

Likewise, the distance between \( r_a \) and \( r_n \) is less than \( m \). Although the distance between encoded anchor-positive is less than encoded anchor-negative, the loss function exceeds zero.

### 5. EVALUATION

#### 5.1. Dataset

We use an open dataset [11], collected using a mmWave radar. It has over 10,000 samples of gestures from 21 classes. We adapt the train/test split provided in the dataset and use 10% of the training data for validation.

#### 5.2. Evaluation Metric

To detect if a pair of gestures are from the same class, we define a distance threshold. If the distance between the encoded representation of the gestures is less than the threshold, they are considered as similar gestures. The threshold value is determined via the Equal Error Rate (EER) using validation data. Finally, to evaluate the model, we determine an average threshold distance by running the model on validation set for 10 times. Then, we evaluate the average accuracy and standard deviation of the model by running it on the test set for 100 times with random pairs.

#### 5.3. Contrastive loss vs. Triplet loss

The effect of training the proposed network using contrastive and triplet losses with online and random pair/triplet selection mechanisms is shown in Fig. 4. In both loss functions, using an online pair/triplet selection increases the accuracy. Triplet loss achieves better results than contrastive loss. Hence, we use the triplet loss with an online semi-hard triplet selection approach for rest of the experiments.
5.4. Effect of Margin

The effect of margin value on average accuracy for values from 1 to 10,000 is shown in Fig. 5. The optimal margin value for the data is 10. As we further increase the margin value, the accuracy decreases. In addition, we show the effect of margin on the encoded representation of the validation set in Fig. 8. The higher the margin value, the larger the representation space. Intuitively, we observe that the classes are separated slightly better in Fig. 8.a compared to the other two.

5.5. Comparison to State-of-The-Art

We compare with two different models: (I) zero-shot proposed by [11] who encoded a gesture, a k-Means clustering algorithm applied on the latent space to cluster the gestures. (II) Using the baseline encoder [16] in the architecture in Fig. 1 instead of using P-GEN. The results are shown in Fig. 6. We outperform both models in all cases with a considerable margin. Additionally, the Floating Point Operations (FLOPS) required to extract the encoded representation for a gesture is 15.16, 0.87, and 0.27 millions for Pantomime, baseline, and the proposed model respectively. It shows our graph-based model is 60 and 3 times computationally efficient compared to Pantomime and the baseline.

We further study the effect of training class selection on the accuracy (training with three sets of randomly chosen classes) in each setting (Fig. 7). With increasing unseen classes, accuracy decreases. As a result, the model would not learn sufficient discriminative features to encode the challenging gestures during inference. Conversely, classes in training might be extremely hard for the model to separate. The model will then overfit the challenging classes and fail to encode easier classes. A possible solution is to increase the number of training classes to reduce variance (Fig. 7).

6. CONCLUSION

We have introduced a novel neural network architecture for zero-shot motion recognition from 4D point clouds. We have shown that P-GEN is up to 60 times computationally efficient compared to state-of-the-art while outperforming them with a considerable margin in terms of average accuracy of unseen classes of gestures. We also studied different loss functions, training schemes, unseen scenarios, and margin values through a set of extensive experiments.

7. ACKNOWLEDGMENT

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813999 and ERANET-COFUND (H2020) CHIST-ERA III, RadioSense. The calculations presented above were performed using computer resources within the Aalto University School of Science “Science-IT” project.

8. REFERENCES

Fig. 8: 2D latent space representation on validation set for different margin values of online triplet loss


