Salami, Dariush; Palipana, Sameera; Kodali, Manila; Sigg, Stephan

Motion pattern recognition in 4D point clouds

Published in:
Proceedings of the 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing, MLSP 2020

DOI:
10.1109/MLSP49062.2020.9231569

Published: 01/09/2020

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

Please cite the original version:
Salami, D., Palipana, S., Kodali, M., & Sigg, S. (2020). Motion pattern recognition in 4D point clouds. In *Proceedings of the 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing, MLSP 2020* Article 9231569 (IEEE International Workshop on Machine Learning for Signal Processing). IEEE. https://doi.org/10.1109/MLSP49062.2020.9231569

2020 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2020, ESPOO, FINLAND

MOTION PATTERN RECOGNITION IN 4D POINT CLOUDS

Dariush Salami, Sameera Palipana, Manila Kodali, Stephan Sigg

Department of Communications and Networking, Aalto University, 02150 Espoo, Finland

ABSTRACT

We address an actively discussed problem in signal processing, recognizing patterns from spatial data in motion. In particular, we suggest a neural network architecture to recognize motion patterns from 4D point clouds. We demonstrate the feasibility of our approach with point cloud datasets of hand gestures. The architecture, PointGest, directly feeds on unprocessed timelines of point cloud data without any need for voxelization or projection. The model is resilient to noise in the input point cloud through abstraction to lower-density representations, especially for regions of high density. We evaluate the architecture on a benchmark dataset with ten gestures. PointGest achieves an accuracy of 98.8%, outperforming five state-of-the-art point cloud classification models.

Index Terms— 4D point clouds, gesture recognition, deep learning

1. INTRODUCTION

4D point cloud data, comprising three spatial dimensions and time, is at the heart of numerous timely applications in signal processing, such as perception in robotics or autonomous driving, or recognition of motion and activity patterns. While several approaches for the processing of stationary 3D point clouds exist, 4D point clouds feature unique properties not addressed by these traditional approaches. In particular, the density of individual regions in the point cloud is not constant over time, and the amount of change might contain information in itself. Furthermore, in the temporal domain, new points appear while others disappear. Traditional approaches do not consider the temporal relation in shape, density, and points of 4D point clouds.

Over the past few decades, sensors for spatial information have evolved from simple range sensors based on sonar or IR, providing a few bytes of information about the world, to ubiquitous cameras to laser scanners. Recently, LIDAR sensors are able to provide high-quality 3D representations of the world: point clouds. Point clouds are richer in detail than 2D images due to the provision of a third dimension, and are invariant to illuminations. Lately, further 3D sensors have become available that provide the same type of data, such as the Kinect sensor for the Microsoft Xbox 360 game system, or also millimeter-wave radar sensors.

A point cloud recorded from a human gesture carries unique properties. It comprises a sequence of frames, and each frame consists of a set of unordered points in a 3D space. The points in a frame are not isolated, and together with the neighboring points, they form a meaningful structure. Additionally, rotating or translating nearby points all together does not change the point cloud's underlying structure. This structure is used by gesture recognition models to classify different motion patterns.

Hand gesture recognition has been an active research topic in recent years. It has branched out into different categories, such as hand motion recognition, hand pose estimation, or hand tracking. We focus on hand motion gesture recognition, which has applications in human-computer interaction, virtual reality, gaming, human-robot collaboration, and sign language recognition.

In this work, we develop a neural network to learn the features of 4D point cloud data. The point cloud operates in the Euclidean space over time. To capture temporal dependencies, we employ multiple parallel blocks of a known 3D point cloud processing model, PointNet++, supplemented by LSTM layers to extract temporal features of a 4D point cloud. In particular, we propose a hybrid architecture that combines PointNet++ for framewise spatial feature extraction and Long Short-Term Memory (LSTM) [1] modules to optimize spatio-temporal feature extraction.

We compare our approach to five state-of-the-art algorithms for point cloud data, namely spatiotemporal learning [2], PointNet [3], PointNet++ [4], PAN [5], and FlickerNet [6] on a benchmark dataset for human gesture recognition from point cloud data [5]. PointGest outperforms all state-of-the-art algorithms in terms of gesture recognition accuracy on this 4D point cloud data. Our contributions are as follows:

1. We propose PointGest, a neural-network architecture for recognizing gestures by directly consuming 4D point cloud data without any voxelization or projection steps.
2. We demonstrate the performance of PointGest on benchmark 4D point cloud data for gesture recognition by outperforming the state-of-the-art models by up to 9.6%.
3. We analyze the robustness of PointGest to missing points by visualizing the critical points contributing to the last parallel max-pooling layers in the architecture.

978-1-7281-6662-9/20/$31.00

2. RELATED WORK

A popular modality for motion gesture recognition has been red, green, blue (RGB) images and video [7] mainly due to the success of 2D image classification using convolutional neural networks (CNNs). More recently, though, through depth-capable sensors like MS Kinect, 3D data like RGB-Depth information has increasingly been used for motion gesture recognition. An MS Kinect sensor's output is an RGB-depth image obtained by overlaying an RGB image from a camera and depth information from an infra-red sensor. A large body of research directly classifies RGB-depth information for gesture recognition [8].

Hand pose estimation and hand motion recognition using 3D point cloud data have been previously studied in [9] and [5, 2] respectively. Hand pose estimation applications are based on spatial features of the input point cloud. But, in hand motion recognition not only spatial features are required but also temporal features play an important role. Inspired by successful CNN architectures in image classification, most of the existing methods have used CNNs [10, 11]. To effectively process 3D point clouds, either multi-view CNNs have been proposed to use the projected 2D images for classification [12, 10] or dense occupancy grids have been generated using the voxelization process to make the data suitable for 3D CNNs [2]. Information loss due to the projection or voxelization and cubical growth in computational complexity and memory consumption due to the dense occupancy grid generation are important drawbacks of these methods, which make them unsuited for real-time scenarios.

Another line of work consumes the point cloud without prior transformations. These models are inspired by PointNet++, a pioneering work that classifies and segments stationary 3D shapes. Kingkan et al. [5] leveraged PointNet++ for gesture recognition with attention modules in which, the gather and the scatter operations sample the most informative points of the gesture. FlickerNet [6] extended the set abstraction level (SAL) of PointNet++ (SALs map a set of points to a sparser subset) to a spatio-temporal SAL and consumed the point cloud by aggregating all frames of a gesture.

In contrast, we propose a hybrid architecture that combines PointNet++ for framewise spatial feature extraction and Long Short-Term Memory (LSTM) [1] modules to optimize spatio-temporal feature extraction.

3. SYSTEM ARCHITECTURE

We discuss the architecture in three sections: spatial feature extraction, temporal feature extraction, and classification.

3.1. Spatial feature extraction

PointNet++ [4] is a deep neural network architecture mimicking the functionality of classic (2D) CNNs to extract hierarchical spatial features from (3D) point clouds. Local features are first extracted to capture important structures in small regions, and the local features are further grouped into larger units to extract higher-level features. The same steps are repeated until the features of the whole point set are computed.

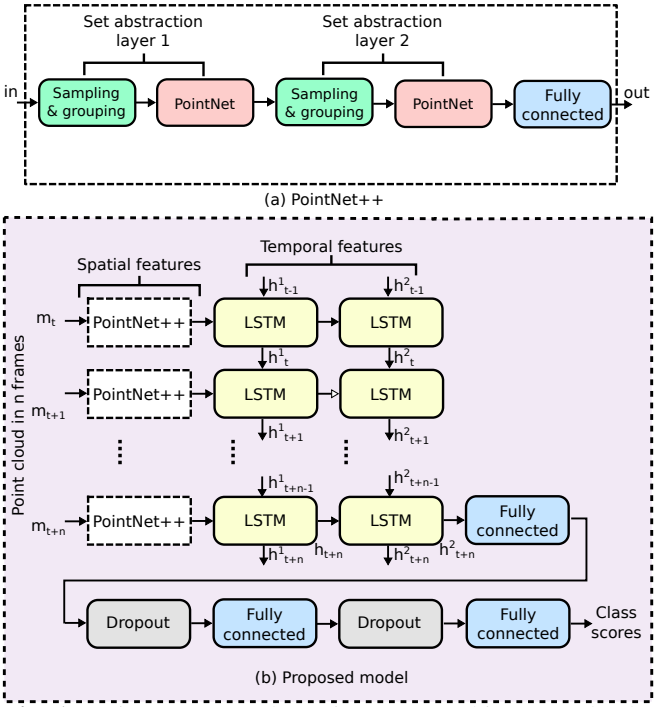

Fig. 1: PointGest: (a) The PointNet++ architecture, (b) the proposed architecture.

PointNet++ captures point cloud features iteratively, via multiple SALs to emulate multiple convolution levels in CNNs. A single SAL consists of *Sampling*, *Grouping* and *PointNet* layers. The Sampling layer outputs a set of points from the input points that define the centroids of local regions. The Grouping layer builds sets of local regions by selecting "neighboring" points of the centroids. Finally, the PointNet layer encodes the local region patterns into feature vectors.

Mathematically, the last set abstraction level maps a given unordered point set $\{z_1, z_2 \dots z_N\}$ with $z_i \in \mathbb{R}^n$ to a sparser subset with higher dimension $\{y_1, y_2 \dots y_M\}$ with $y_i \in \mathbb{R}^m, m > n$ and $M < N$. The mapping function $q(\cdot)$ is:

$$y_i = q(z_i) = \gamma\left(\max_{z_j \in \mathcal{N}(p_i); j=1\dots N} h(g(z_{p_i}, z_j))\right), \qquad (1)$$

where $\gamma(\cdot)$ and $h(\cdot)$ are multi-layer perceptron (MLP) networks, and $g(\cdot)$ is a grouping layer to construct local regions by finding all points in the spatial neighborhoods $\mathcal{N}(p_i)$ of the centroids z_{p_i} in the point set $\{z_1 \dots z_N\}$ [4]. The furthest-point sampling algorithm [13] is used to select centroids.

In PointGest, we utilize n parallel PointNet++ modules to extract spatial features from each frame. The higher the number of the trainable parameters, the easier the model can

memorize the target class for each training sample, however, this can lead to overfitting. To reduce the model's complexity in terms of trainable parameters and to prevent overfitting, we share all weights among the parallel PointNet++ blocks. As each PointNet++ block has 1.47M trainable parameters, with eight parallel PointNet++ blocks, we would have 11.76M parameters. By sharing all these weights among the parallel blocks, we decrease the trainable parameters to 1.47M.

3.2. Temporal feature extraction

In 4D point cloud classification, temporal features play a crucial role. Therefore, PointGest uses parallel PointNet++ blocks to extract spatial features from each frame, and LSTM layers to extract temporal features from the 4D point cloud. The n preprocessed frames are fed into n parallel PointNet++ layers (Figure 1) to extract spatial features. These are then transferred to two stacked LSTM layers to extract temporal features. After the LSTM layers, we add two fully-connected layers with a dropout layer (for regularization) in between to reduce the size of the temporal feature vector and to obtain fine-grained features.

The following equations define an LSTM unit:

$$f_t = \sigma_g(W_f x_t + U_f h^p_{t-1} + b_f) \quad (2)$$

$$i_t = \sigma_g(W_i x_t + U_i h^p_{t-1} + b_i) \quad (3)$$

$$o_t = \sigma_g(W_o x_t + U_o h^p_{t-1} + b_o) \quad (4)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h^p_{t-1} + b_c) \quad (5)$$

$$h^p_t = o_t \circ \sigma_h(c_t) \quad (6)$$

where $h^p \in \{h^1, h^2\}$, x_t is the input at time step t, h^p_{t-1} is the hidden state from the previous time step, W and U are weight vectors, b is the bias term, σ is the sigmoid activation function, and f, i, o, c are forget, input/update, output, and control gates respectively. For ease of presentation, we define $h^p_t = \text{LSTM}(x_t, h^p_{t-1})$ as alias for these equations. The spatio-temporal latent features are calculated by

$$L_k = \text{LSTM}(\text{LSTM}(\alpha(q(q(z_i))), h^1_{t-1}), h^2_{t-1}) \quad (7)$$

where α is an MLP and $k \in \{1, 2, \ldots, K\}$ is the frame index with last frame K.

3.3. Classification

After extracting spatio-temporal features of the input moving point cloud, we empirically choose two consecutive fully connected (FC) layers to further extract fine-grained latent 256-dimensional feature vectors. Finally, we feed the global feature vector to another fully connected layer with no activation function to obtain the class scores. The final layer has C units, which is the number of gesture classes in the dataset. The output of the network is the class scores of size C. PointGest is a neural network with more than 1.5M parameters. To decrease the over-fitting probability, we use a dropout layer [14] between two consecutive fully connected layers.

Table 1: Experimental settings.**I**: Single user, **II** 70:30 split, 5 users, **III** leave 2 subjects out from 5 users, **II(6)** 70:30 split, 6 users, **III(6)** leave 4 subjects out from 6 users.

		I	**II**	**III**	**II(6)**	**III(6)**
Total users		1	5	5	6	6
Training	*# users*	1	5	2	6	2
	User ID	1	1-5	1-2	1-6	1-2
	Frames	24,861	65,916	56,468	83,795	56,468
	%	43	56	49	59	40
Validation	*# users*	1	1	1	1	1
	User ID	1	1	1	1	1
	Frames	22,748	22,748	22,748	22,748	22,748
	%	40	20	20	16	16
Testing	*# users*	1	5	3	6	4
	User ID	1	1-5	3-5	1-6	3-6
	Frames	9,776	28,250	35,912	35,912	63,239
	%	17	24	31	25	44

The classifier of the dense model is defined as

$$\text{Output} = \beta(L_{k=K}), \quad (8)$$

where β is another MLP and K is the last frame.

4. THE UBPG DATASET

We evaluate our model's performance using the UBPG dataset [5], a point cloud-based dataset comprising ten classes of gestures. Figure 2 illustrates the progress of the 10 gestures over time instants t_1, t_2, t_3 and t_4. The dataset contains 142,455 frames from 3882 video samples. The RGB-depth video samples collected using MS Kinect are segmented to gestures and transformed into a point cloud. Human subjects performed gestures, and each person repeats each gesture at least 30 times. The point clouds capture the upper part of the body.

The classification performance using the unprocessed point clouds was tested in five settings (cf. Table 1). The table describes the number of users, their IDs, number of frames, and how data was split into training, validation, and testing sets. Table 2 shows the class distributions in the training set for each setting. Setting **I** evaluates the model's performance on a single trained user, **II** tests the model's performance on the whole dataset, and **III** splits the training and testing data according to leave-k-subjects-out cross-validation. The dataset was initially performed and evaluated in the literature comprising five subjects. Recently, it has been extended to six subjects. For completeness, we also conduct experiments based on data from all subjects (**II(6)** and **III(6)**).

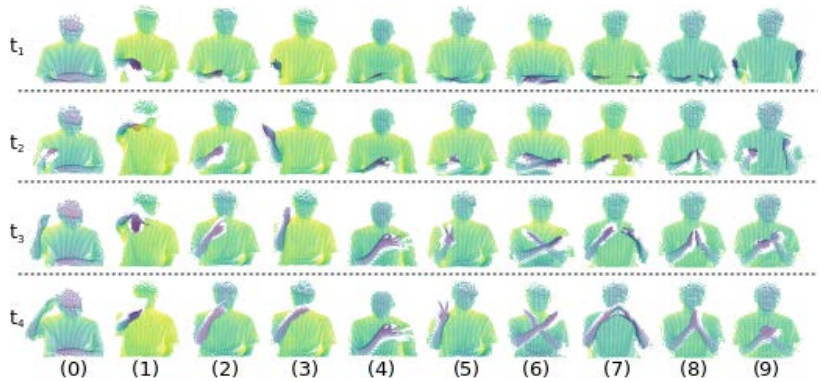

Fig. 2: The visualization of the point cloud of the gestures (0) no gesture, (1) come here, (2) me, (3) no thanks, (4) money, (5) peace, (6) not allowed, (7) ok, (8) I am sorry, (9) I got it over four different time frames t_1, t_2, t_3, t_4.

Table 2: Class distributions in the training set for each setting.

	Setting			
Gesture (%)	**I**	**II**	**II(6)**	**III, III(6)**
No gesture	9.74	9.24	9.26	9.79
Come here	10.48	10.31	10.12	10.57
Me	10.72	10.20	10.38	10.43
No thanks	10.74	10.70	10.62	10.93
Money	11.07	10.67	10.47	11.02
Peace	10.81	10.18	9.97	10.52
Not allowed	10.72	10.51	10.15	10.66
OK	8.02	8.63	9.05	7.75
I am sorry	8.58	10.03	10.27	9.49
I got it	9.12	9.52	9.70	8.84

5. IMPLEMENTATION DETAILS

We use Tensorflow and Keras frameworks for all experiments. The batch size is 16, and the optimizer is Adam [15]. We use an exponential decay strategy with

$$d_{lr} = lr * dr^{(gs/d\_s)}, \tag{9}$$

with decayed learning rate d_{lr}, current learning rate lr, decay rate dr, global steps gs, and d_s decay steps. Then, we clip the decayed learning rate for every step using

$$f_{lr} = max(d_{lr}, 0.00001), \tag{10}$$

where f_{lr} is the final learning rate used in each step.

To prevent the models from overfitting, we use an early stopping mechanism with at most 800 epochs and patience of 40 epochs. We use the winner model before stopping for validation and test purposes. Experiments were run on a Tesla V100 GPU with 32GB memory.

6. RESULTS

PointGest achieves 97.4%, 98.8%, and 85.4% accuracy on settings **I**, **II**, and **III** respectively. As shown in table 3, gesture (1) 'Come here' is conflicted with the first class which is (0) 'No gesture'. Class 0 includes several types of recordings, some of which are a person without any movements, while others are gestures that do not belong to any of the other classes. Table 1 shows that the number of training frames for setting **I** is less than one-half of the rest of the settings, so the reason could be the lack of the training data in this setting.

According to the confusion matrix in table 4, 9 out of 10 gestures have an accuracy of more than 98% in setting **II**, and there is no notable conflict among gestures.

The confusion matrix for setting **III**, which is shown in table 5 indicates that a few gestures have conflicts with each other. The gesture (6) 'Not allowed' has conflicts with (8) 'I am sorry' and (9) 'I got it'. As shown in Fig. 2, all these gestures are both-handed. Moreover, there are conflicts between gestures (5) 'peace', (4) 'Money', (3) 'No thanks', and (2) 'Me', which are one-handed. We do not have these conflicts in setting **II**. These observations show that the style of making these conflicting gestures is different among users in the test set and the users in the train set since we have a different set of users for those two sets in Setting **III**.

6.1. Comparison with the state of the art

We evaluate PointGest in three different settings against five state-of-the-art neural network architectures for point cloud classifications. PointNet [3] and PointNet++ [4] treat the gesture recognition task as a shape classification problem by aggregating the temporal features of the 4D point cloud into 3D point clouds, while O&H [2], PAN [5], and FlickerNet [6] work on 4D point clouds. Due to the unavailability of source code for PAN and FlickerNet, we compared to their performance reported in [5] and [6] on the same data set.

PointNet is a neural network architecture that consumes point clouds. It uses symmetric functions to ensure the invariant to permutations of points and a specific type of transformer network to cope with geometric transformations of both input and latent features. PointNet++ utilizes a simplified version of PointNet in two consecutive set abstraction

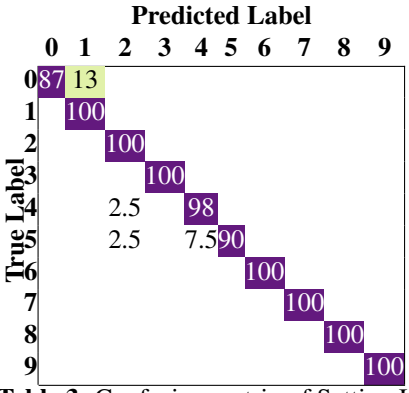

Table 3: Confusion matrix of Setting I.

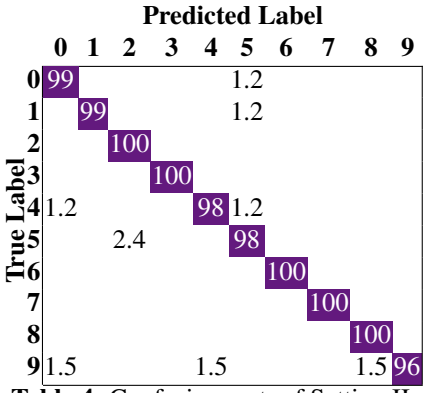

Table 4: Confusion mat. of Setting II.

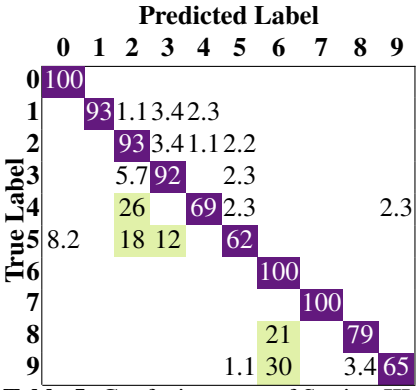

Table 5: Confusion mat. of Setting III.

Table 6: Comparison of our results with 5 SoA architectures in 5 different settings. PC: point cloud, fr.: frames

Model	Input	I	II	III	II(6)	III(6)
O&H [2]	voxels	-	84.4	-	-	-
PointNet [3]	PC, 1 fr.	91.8	86.2	56.3	84.9	54.6
PointNet++ [4]	PC, 1 fr.	90.2	89.8	61.9	91.5	58.9
PAN [5]	PC, 4 fr.	94.2	-	-	-	-
FlickerNet [6]	PC, 16 fr.	96.4	95.3	77.9	-	-
Ours	PC, 8 fr.	97.4	98.8	85.4	96.9	67.5

layers as it is shown in Fig. 1 to capture local structures.

O&H models the temporal dependency as an extra dimension on the input tensor. Consequently, it consumes a 4D tensor in which the fourth dimension is time. Apart from the temporal mapping, O&H requires the input to be a dense occupancy grid, which is generated by a voxelization process. O&H uses four 3D convolution layers to extract spatio-temporal features and two fully connected layers to extract fine-grained features and to classify the gestures.

PAN expects 4D input, which are the XYZ coordinates and the temporal frames. This model uses gather and scatter operations to sample points in the feature space for an attention mechanism that is developed to emphasize points that are more important for classification. FlickerNet extends PointNet++ towards 4D point clouds. It has a specific type of SAL capable of extracting spatio-temporal features from a time window in the input 4D point cloud.

In comparison to these state-of-the-art architectures, PointGest features an increased classification accuracy by 1%, 3.5%, and 7.6% in settings **I**, **II**, and **III** respectively.

6.2. Experiments with 6 subjects

The UBPG dataset has been extended to 6 subjects recently. We evaluate our model with the other models, for which source code is available, on two scenarios with six subjects. As shown in table 6, PointGest outperforms PointNet++ with a margin of 5.4% and 8.6% on **II(6)** and **III(6)**.

6.3. Effect of different parameters on the accuracy

We also investigated the effect of frame count and points in each frame on the accuracy of PointGest. As shown in Fig. 3, both the number of frames and points in each frame impact the model performance. For a gesture with 32 points by increasing the number of frames from 2 to 8, we obtain a 29% gain in the accuracy. For a gesture with eight frames, we can improve the accuracy by 16% if we increase the number of points in each frame from 32 to 1024.

Moreover, by decreasing the number of points from 1024 to 128 we only lose 1%, 3%, and 1% accuracy with 8, 4, and 2 frames respectively. Consider a very strict scenario in which we decrease the number of points from 1024 to 32. By eliminating almost 97% of points, we only lose 16% accuracy, which shows that PointGest is resilient to missing points.

6.4. Visualization of critical points

We visualize critical points that contribute to the last max-pooling layer of PointNet++ blocks in the architecture to analyze the model's robustness to missing points. In particular, we repeat one arbitrary point of the input point cloud 1024 times and remove the remaining points in each frame. Then, we bring the removed points back one by one so that we replace it with one of those repeated arbitrary points and measure the last max-pooling layer's output in each frame. Points that improve the last max-pooling layer's output are considered critical points.

As shown in figure 4, although we feed 1024 points in each frame to the model, only a small portion of them (200-250) defines the output of the last max-pooling layer on which we apply LSTM layers. Consequently, every point cloud between the original input and the critical point set would end up in the same spatial features. We conclude that the model can tolerate significant noise in each frame without a notable drop in the accuracy.

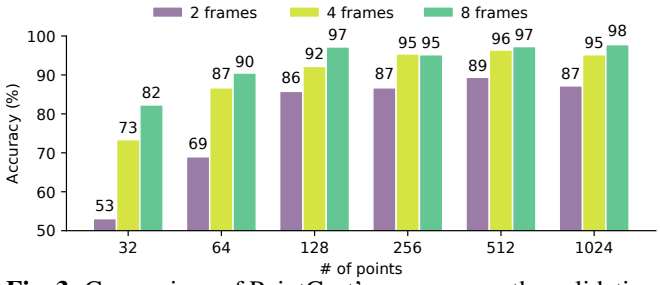

Fig. 3: Comparison of PointGest's accuracy on the validation set for different combinations of the number of frames and the number of points in each frame

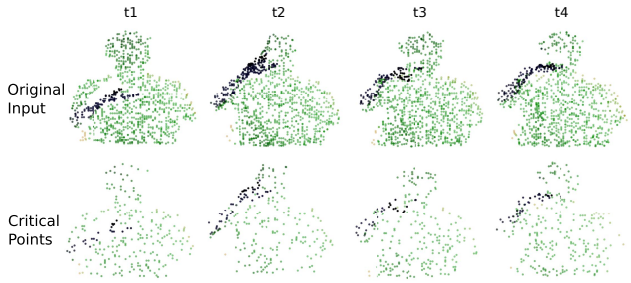

Fig. 4: Visualization of the original input frames of a gesture and the output of critical point set of four Pointnet++ units

7. CONCLUSION

We have introduced PointGest, a novel neural network architecture for motion recognition from 4D point clouds. We have shown that PointGest is robust to noisy point cloud inputs without a notable drop in the overall accuracy because of its ability to summarize the input point cloud to an intermediate point cloud. PointGest outperformed five state-of-the-art point cloud classifiers on a benchmark dataset comprising ten human gestures in five different settings. We plan to decrease the computational cost of PointGest to make it suitable for scenarios with a higher number of frames.

8. ACKNOWLEDGMENT

The calculations presented above were performed using computer resources within the Aalto University School of Science "Science-IT" project. We appreciate partial funding in the frame of the ChistEra project RadioSense and the MSCA-ITN-ETN Windmill.

9. REFERENCES

[1] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] Joshua Owoyemi and Koichi Hashimoto, "Spatiotemporal learning of dynamic gestures from 3d point cloud data," in *Proc. of ICRA*, 2018.

[3] Charles R Qi et al., "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. of CVPR*, 2017.

[4] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. of NIPS*, 2017.

[5] Cherdsak Kingkan, Joshua Owoyemi, and Koichi Hashimoto, "Point attention network for gesture recognition using point cloud data.," in *Proc. of BMVC*, 2018.

[6] Yuecong Min et al., "FlickerNet: Adaptive 3D Gesture Recognition from Sparse Point Clouds," in *Proc. of BMVC*, 2019.

[7] Siddharth S Rautaray and Anupam Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," *Artificial intelligence review*, vol. 43, no. 1, pp. 1–54, 2015.

[8] Pichao Wang et al., "RGB-D-based human motion recognition with deep learning: A survey," *Computer Vision and Image Understanding*, vol. 171, pp. 118–139, 2018.

[9] Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan, "Hand PointNet: 3D hand pose estimation using point sets," in *Proc. CVPR*, 2018.

[10] Liuhao Ge et al., "Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs," in *Proc. CVPR*, 2016, pp. 3593–3601.

[11] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann, "3D convolutional neural networks for efficient and robust hand pose estimation from single depth images," in *Proc. CVPR*, 2017, pp. 1991–2000.

[12] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. ICCV*, 2015, pp. 945–953.

[13] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Trans. on Image Process.*, vol. 6, no. 9, pp. 1305–1315, 1997.

[14] Nitish Srivastava et al., "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, 2014.

[15] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.