Point Attention Network for Gesture Recognition Using Point Cloud Data

Cherdsak Kingkan*
kingkan.cherdsak.r3@dc.tohoku.ac.jp
Joshua Owoyemi*
tjosh@dc.tohoku.ac.jp
Koichi Hashimoto
koichi@m.tohoku.ac.jp

Department of System Information Sciences, Graduate School of Information Sciences
Tohoku University
Sendai, Japan

Abstract

In this work, we propose a neural network with novelty attention modules to perform human gesture recognition from point cloud data. Our network directly takes point clouds as input, and the attention modules learn to pay attention to only points that contribute to more accurate gesture recognition. The input point cloud consists of both spatial and temporal dimensions since the point cloud frames are concatenated as one input sample. This enables the network to learn the spatio-temporal features of the data without explicit modeling of gesture dynamics. The gather and scatter operations are proposed for points downsampling and upsampling in the feature space. We evaluate the performance of our network using a dataset of common Japanese gestures. The proposed network achieves state-of-the-art performance on this dataset. The analysis of architecture design and parameter choices are also discussed.

1 Introduction

Human gesture recognition plays an important role in several application domains ranging from human-robot collaborative [18] to video surveillance for security [10], and sign language recognition [20]. The problem has been extensively studied in the computer vision community by using images and videos [21, 27, 34]. However, 2D images give only limited information of the physical size and shape of an object in a scene. Recently, 3D sensors became more affordable, this allows 3D data such as depth images and 3D point clouds for more uses. These 3D data representations have many advantages over its 2D counterpart such as explicit geometry, and illumination invariance. For example, point clouds express the geometry in terms of 3D coordinates so that the size and shape of an object in a scene can be straightforwardly computed from its 3D coordinates. A depth image has been popularly used in many gesture recognition works [3, 28]. For example, Xia *et al*. [33] proposed a filter method to extract local spatio-temporal interest points from depth videos for action recognition. However, many of the works relied on hand-engineered features extracted from 3D data. Thus automatic feature extraction methods are needed.

* indicates equal contributions

In this paper, we are interested in using a neural network to learn features of point cloud data for gesture recognition. The point cloud is an unorganized set of points in 3D space. Point clouds can be easily acquired using 3D sensors like Kinect camera or intel RealSense camera. It has been used in several applications including gesture recognition. The gesture recognition is mainly focusing on the upper body movements of the subject which convey a meaning within a particular context. Therefore, designing a network with attention capability to focus on certain parts of the body for gesture recognition is an interesting area of research. However, there is a difficulty using point clouds with typical convolutional architectures which require regular input data formats, like those of image grids or 3D voxels. Since point clouds are not in a regular format, it is normally transformed to 3D voxels or multi-views of images before feeding them to deep learning networks. For example, Liang *et al*. [15] proposed a method to perform a hand gesture recognition which projects point cloud into view images before feeding into a convolutional neural network for feature extraction.

The main challenge in our work is to design the attention mechanism which can be used with point cloud data. In this work, the point cloud frames are directly fed into the network. The main contributions of this paper are as follows: (1) We propose a neural network with attention modules for predicting dynamic gestures using point cloud data, (2) The gather and scatter operations are proposed in order to sample the most informative points in the feature space for attention mechanism. The attention modules make use of these operations in order to downsample and upsample the number of points in input point clouds. (3) We demonstrate our network's performance on the common Japanese gestures dataset. As a result, the network outperforms the state-of-the-art model on this dataset. We also show the effectiveness of the attention modules.

The remaining of the paper are organized as follows. Related works are discussed in Section 2. The proposed attention module and network architecture are explained in detail in Section 3. In Section 4, the experiments are demonstrated, the results and analysis are also discussed. We then conclude our work in Section 5.

2 Related Work

Deep Learning on Point cloud: The problem of utilizing point clouds as inputs of deep learning models has been addressed for several tasks in the past few years [22, 23, 26, 32]. Qi *et al*. [22] is a pioneer introducing a network, named PointNet, that worked directly on point cloud data for 3D object classification and segmentation. Klokov *et al*. [13] proposed a Kd-network which takes a Kd-tree structure of point clouds as an input. This network does not require input to be in the grid-like structure. However, all of these works focused only on rigid object classification and segmentation.

Attention Mechanism: Attention mechanism is inspired by how the human eye works, i.e. our eye tends to pay attention to only a particular region of the view we are looking at. Attention mechanism has been used mostly for image and video captioning [35] and machine translation [29]. There are two types of attention (1) soft attention and (2) hard attention. Soft attention is a differentiable deterministic process so that it can be trained with a backpropagation algorithm. Examples of soft attention are [8, 30, 35]. Sharma *et al*. [25] proposed a network for action recognition using visual attention which is a kind of soft attention. Hard attention, on the other hand, is a non-differentiable stochastic process and relies on a sampling-based method for training. Gregor *et al*. [7] proposed a network which classifies and generates images by focusing on arbitrary regions of the input and output

respectively. However, there are not many works that utilize attention mechanism with 3D data, especially point cloud data. Thus, we propose to leverage the soft attention mechanism for gesture recognition using point cloud data in this work.

Deep Learning for Gesture Recognition: Learning-based methods outperform traditional approaches that relied on handcrafted features [16, 31]. Asadi-Aghbolaghi *et al*. [2] provided a survey on deep learning approaches which utilize image sequence for gesture and action recognition. Many works have focused only on an image for gesture recognition. For 3D data, Kang *et al*. [11] used the depth images as input of the convolutional neural network for a real-time sign language fingerspelling recognition. Owoyemi and Hashimoto [19] converted point cloud frames into 3D occupancy grids before feeding them into 3DCNN for gesture prediction. Using such a data representation has a few drawbacks. First, 3D voxels consume a lot of memory since there is a need for storing 3D grids. Second, the models require much computational cost because the conversion from point clouds to 3D voxels is required before a prediction.

3 Method

In this section we introduce the main components of our framework. We first explain the proposed gather and scatter operations which are used in the attention modules for downsampling and upsampling the input features. The structure of our attention modules and how the gather and scatter are used are then discussed. Finally, the network integrated with attention modules is explained in details.

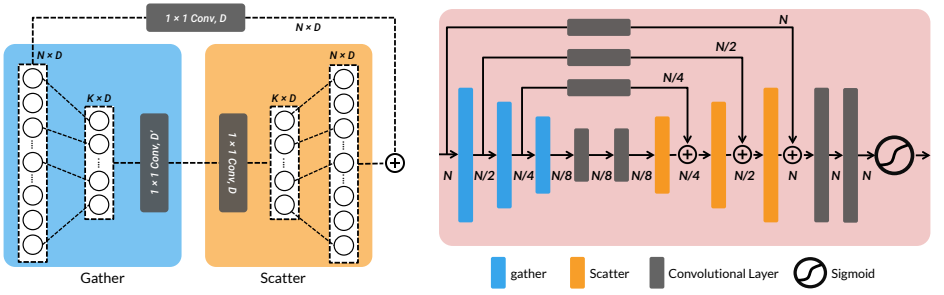
3.1 Gather and Scatter Operations

The proposed attention module makes use of bottom-up top-down structure to learn attention masks. This structure requires the network to downsample and upsample the feature maps. Since a point cloud is an unorganized set of points, the spatial relationship among points is not available, i.e., there is an $N!$ permutations of points in data feeding order. So that the traditional max-pooling for downsampling and the deconvolution operation [36] for upsampling are not applicable. Therefore we propose the gather and scatter operations for downsampling and upsampling points in the point cloud. These operations are inspired by [24], but our operations differ from [24] in that there is no need of mask to guide the convolutional kernels.

The fundamental idea behind our gather and scatter operations is that instead of convolving the features of every point, only the most informative K points in the feature space are selected for further feature learning. The informative points are identified by the average values along the feature dimension. These can be seen as sparse operations. Let assume that the input tensor has a shape of $N \times D$, where N represents the number of points, and D represents the number of feature dimensions.

Gather: The gather operation is depicted in Figure 1(a). The average of feature values of each point, $p_i \in \mathbb{P}$, for $i = 1,\ldots,N$ and $\mathbb{P}$ is a set of points, is computed. The top K points with highest average values are gathered as the most informative points in the feature space where $K < N$. These K points are then sliced out, giving the output tensor of shape $K \times D$. Mathematically,

$$T\{p_1,\ldots,p_K\} = Top\_K(\overline{f}_{n=1,\ldots,N}), \tag{1}$$



(a) (b)

Figure 1: (a) The gather and scatter operations for downsamping and upsampling point clouds. Note that D and D' represent the channel dimensions of conv. layers. (b) The attention module in the bottom-up and top-down structure with skip connection.

where T denotes the output tensor of the gather operation, $\overline{f}$ represents the average feature value of each point in the feature space. Top_K is an operation that picks out K points with largest $\overline{f}$. The order of the selected points in T is in descending order. This tensor can then be used for feature learning. Note that a list of indices of the top K points is also retrieved for later use in the scatter operation. This operation can be seen as global max-pooling for dimensional reduction.

Scatter: After those most informative K points are processed by some number of convolutional layers, the results are then added back to the same location where they are gathered from in the input tensor. This can be achieved by referring to the list of indices retrieved by the gather operation. Thus the output tensor has the same shape as the input tensor, i.e., $N \times D$. So that this operation can be used for upsampling. Figure 1(a) illustrates how the scatter operation works.

3.2 Attention Module

In our work, we make use of the soft attention mechanism for point attention. The goal of an attention module is to identify the most informative points and exploit their features for the network to make prediction decisions. This attention module is designed on the assumption that only a fraction of points in an input point cloud is required for the network to classify the gestures. This is because only particular parts of body movement are needed for recognizing the gestures. For example, there are only the arm and hand movements involved in the gesture "Come here". The structure of our attention module is inspired by the recent development of visual attention in tasks like segmentation [4], image classification [30], and human pose estimation [6]. The attention module is constructed in a bottom-up top-down feedforward network structure [14, 30]. Our structure differs from [14, 17, 30] in its intention of guiding feature learning. For images, the bottom-up feed-forward process collects global information of an image, and the top-down process combines global and local information with skip connections. In our network with point clouds as input, the features of informative points are learned in the bottom-up step. This is done by the gather operation selects points based on its averaged features' value. Then these learned features are added back to those points in the top-down step. The output feature maps from the attention module are utilized as attention masks.

Figure 1(b) shows the structure of the attention module. In the bottom-up process, the

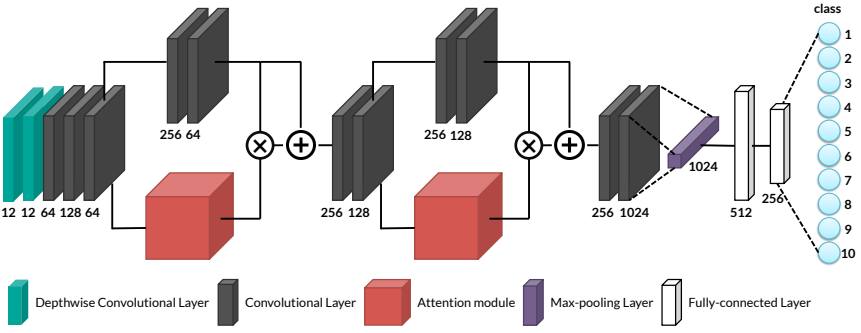


Figure 2: Our network architecture consists of (1) two depthwise conv. layers for learning each point cloud frames independently (2) Convolutional layers, (3) Two attention modules which can be trained in an end-to-end manner, (4) Max-pooling layer for information aggregation, and (5) fully-connected layers.

gather operation is used to select $N/2$ points with highest average feature values where N is the number of input points. These points are then projected onto a new feauture space by a convolutional layer. This process is continued iteratively until the number of points gets down to $N/8$ points. In the top-down process, the gathered points are processed by another convolutional layer before adding back to the positions where they are gathered from in the input tensor using scatter operation. Similar to [17, 30], the skip-connections are also added for feature fusion. The output from the last scatter module feed into two convolutional layers and then the sigmoid function is used to normalize the attention weights into a range [0,1]. These weights are element-wise multiplied and added with the feature maps from the upper branch as shown in Figure 2 to amplified the attended points features. We can define the weighted points features W as,

$$W_{i,j}(p) = A_{i,j}(p) * G_{i,j}(p), \tag{2}$$

where i ranges over all points and j is the index of the channel. The attention mask is denoted by $A(p)$, and $G(p)$ is the point features.

3.3 Network Architecture

Figure 2 shows an architecture of our network. A sequence of F frames of point clouds is concatenated as an input to the network with the shape of $N \times H \times F$, where N is the number of points in a point cloud, H represents the number of coordinates, and F represents the number of frames. Each point p in the point cloud is represented by its coordinate (x,y,z) in a 3D-coordinates. For the input layer, each frame can be considered as a temporal dimension of the input. If each frame is processed independently, the network can learn both spatial and temporal features of the input. To do so, the depthwise convolution operation [5] is utilized to learn features of each point cloud frames independently. The depthwise convolution operation performs spatial convolution on every channel separately. There are 3 filters for each point cloud frames in the first layer and results in $3 \times F$ output channels. Another depthwise convolutional layer with one filter for each channel is applied. Then the convolution layers with $\{64, 128, 64\}$ filters are leveraged to perform a cross-channel combination in order to project the computed channels from the previous step onto a new

space. This output are further fed into two branches. One goes through two convolutional layers. The other goes into the attention module that will learn which points to pay attention to. The output of the first branch and the attention masks are then multiplied and added. There are two attention modules integrated into our network.

As mentioned earlier, the network needs to be invariant to the permutation of points. To solve this problem, the symmetric function in PointNet [22] is adopted in our network. The symmetric function aggregates information from all points, thus the global feature of point cloud can be obtained. This function is defined as follows,

$$h(\{p_1,\ldots,p_n\}) \approx s\{f(p_1),\ldots,f(p_n)\}, \tag{3}$$

where $h:\ 2^{\mathbb{R}^N} \rightarrow \mathbb{R}$, $f:\ \mathbb{R}^N \rightarrow \mathbb{R}^D$, $s:\ \mathbb{R}^D\times,\ldots,\times\mathbb{R}^D \rightarrow \mathbb{R}$. A convolutional layer can be used for approximating f. The symmetric function, s, can be implemented as max-pooling or average pooling operation. In our network, the max-pooling operation achieves the best performance because the aggregated information are mostly from those attended points.

The output of our network is classification score of C classes of gestures. The cross entropy is used as a loss function which can be defined as,

$$\mathcal{L}(\hat{c},c) = -\sum_{j=1}^{C} c_j \log(\hat{c}_j), \tag{4}$$

where c is the ground truth label, $\hat{c}$ is the predicted label, and C denotes the number of gesture classes.

4 Experiments and Evaluations

4.1 Dataset

The proposed network is evaluated using a common Japanese gestures dataset collected by [19]. The dataset consists of 10 classes of gestures which are (1) No gesture, (2) Come here, (3) Me, (4) No Thank You, (5) Money, (6) Peace, (7) Not allowed, (8) OK, (9) I am sorry, and (10) I got it. Figure 3 illustrates sample point cloud frames of the dataset. There are 26,712 point cloud frames for training, and 9,802 point cloud frames for testing. The dataset contains only the upper part of the body. Point clouds are also normalized into [-0.5, 0.5]. The maximum number of points in a point cloud is 2,048 points.

4.2 Implementation details

The default number of points is 2,048 points. We evaluate our model with both single and multiple frames of point clouds. For multiple frames, F consecutive point cloud frames are concatenated together, where $F = \{2,4,8\}$ in our experiments. This can be considered as adding a temporal dimension to the input. The network is trained with the default batch size of 32 for 25 epoch. The initial learning rate is 0.001 with the exponential decay to decrease the learning rate over time. To avoid overfitting and improve the generalizability of the network, point jittering is performed to each point in point cloud frames. Adam optimizer [12] with $\beta = 0.9$ is utilized for optimization. Batch normalization [9] is also used to improve training. The network is implemented using Tensorflow framework [1].

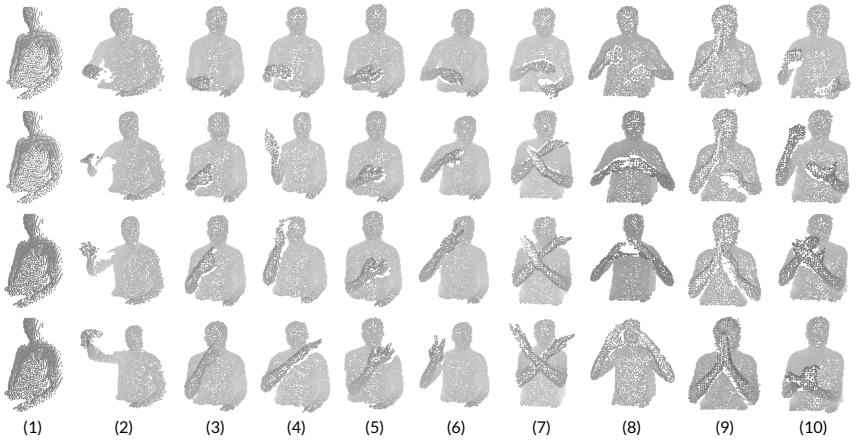


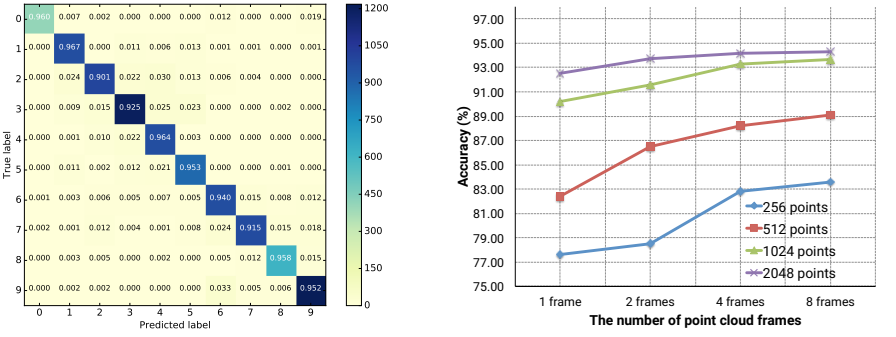
Figure 3: Examples showing the point cloud frames from the common Japanese gestures dataset [19]. The dataset consists of 1 class of no gesture and 9 classes of gestures.The class labels are as follows: (1) No gesture, (2) Come here, (3) Me, (4) No thank you, (5) Money, (6) Peace, (7) Not allowed, (8) OK, (9) I am sorry, and (10) I got it.

Model	Input	Accuracy
Random forest [19]	voxels	67.6
3DCNN [19]	voxels	84.4
PointNet (3.4M param) [22]	points	91.8
Ours (1M param) - single frame	points	**92.5**
Ours (1.3M param) - 4 frames	points	**94.2**

Table 1: Models Performance Comparison

4.3 Models Comparison

We compare our model with the state-of-the-art model on the common Japanese dataset. We also compare our model with a random forest model. Since these models worked only on 3D voxels, thus we compare our model with PointNet [22] which takes point clouds as inputs as well. PointNet is modified by keeping only feature transformer but not input transformer because no rotation is needed. Our model outperforms the state-of-the-art model by a large margin of $\sim 10\%$. Even with a single point cloud frame as an input, our model is still better than the state-of-the-art model. Comparing with PointNet, our model also achieved a better performance with just one-third of the number of training parameters. The confusion matrix of our model on the test set is shown in Figure 4 (a). There are advantages of our model over those that use 3D voxels as input. First, our model directly takes point clouds as input in which no data preprocessing is required. Data preprocessing in this sense is that a conversion from point clouds to 3D voxels. Second, models that use a 3D voxel representation as an input require much more memory than those that use point cloud representation.



(a) (b)

Figure 4: (a) Confusion Matrix of the recognition on the test set, showing the performance of the model in each class. (b) Effect of the number of points and point cloud frames. The metric is overall recognition accuracy on the test set.

4.4 Parameter Choices and Architecture Analysis

In this section, we validate our model with different hyperparameters setting and shows the effects of these parameters on our network's performance. The analysis of the network design is also discussed.

4.4.1 Effect of the number of points and point cloud frames

Here we show the change of our model's performance with regard to the number of points as well as the number of point cloud frames in the training samples. In Figure 4 (b), we can see that as the number of points increases, the model's performance increases. However, there is not much improvement after 1,024 points.

We also experiment with the different number of point cloud frames. Giving the concatenated point cloud frames as input, the model can learn spatiotemporal features in the data. To explore the effect of the concatenated point cloud frames, we first prepare the training data with the different number of frames. We experiment with a single frame, and $\{2,4,8\}$ concatenated point cloud frames. In Figure 4 (b) the result shows that the number of frames affects the performance of our model to some extent. However, our model does not achieve much performance gain using more than 4 concatenated frames of the point cloud. It is worth noticing that even with a single frame of point cloud as an input, our model still outperforms the state-of-the-art model. From the experiments, we can conclude that the number of points affects the model's performance more than the number of concatenated frames.

4.4.2 Attention module

We explore the effectiveness of the attention module with the following questions in mind: (1) How many attention modules are needed for our model ?, and (2) What is the minimum number of points we should downsample to ? Note that, in these experiments, each training sample consists of 4 concatenated point cloud frames with 2,048 points in each frame. To answer the first question, we conduct experiments by adding a different number of attention modules to our network which ranges from 1 to 4 modules. Table 2 (top) shows that using more than 2 attention modules cannot significantly improve the performance but could

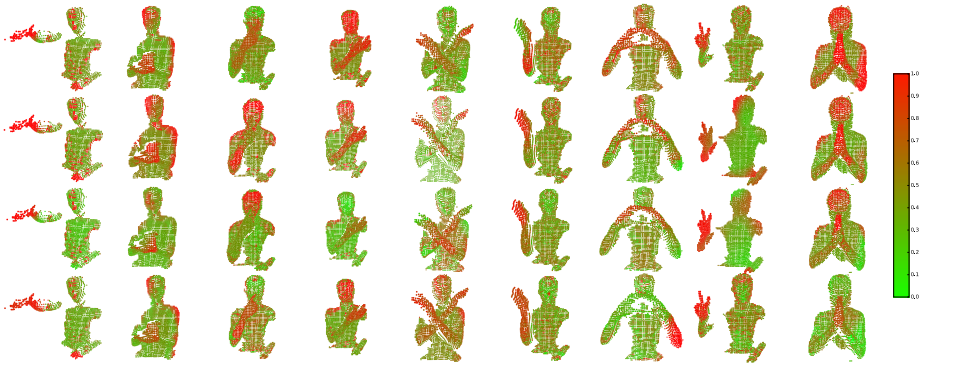


Figure 5: Examples of point clouds with weighted attention masks. The attention modules learn to pay attention to the points on the parts of body that convey the meaning of the gestures.

# attention modules	1	2	3	4	
Accuracy	92.8	94.2	94.35	94.39	
Min. # sampled points	128	256	512	1,024	2,048
Accuracy	87.7	94.2	92.3	91.7	90.1
Batch size	1	4	16	32	64
Accuracy	20.5	91.3	92.6	94.2	94.3

Table 2: Comparison of our model with different configurations.

greatly slow down the training and the deployment of the network. For the second question, we experiment by varying the minimum number of points from N to $N/16$. As shown in Table 2 (middle), the model with downsampled points of $N/8$ achieves the best performance. We observed that the model without downsampling process in the attention module does not perform as good as those with downsampling in the attention module. Figure 5 shows examples of the attention masks of our network after training. We can see that the network pays more attention to the points on the parts of the body that convey the gestures meaning. The attention modules make use of both spatial and temporal context of the input data in order to infer the gesture class.

It is worth noting that points are represented by its coordinates, so that there are gestures which are represented by similar point's coordinates in a 3D space. In such situation, adding the temporal context of the input data can improve the robustness of the network. Figure 6 shows the examples of misrecognized gestures. For example, the gesture "Peace" is predicted as "Come", this misrecognized is caused by the similarity of the gesture in terms of points' coordinates.

4.4.3 Batch size matters

From our experiments, we observe that the model's performance suffers from changing the batch size. Therefore, we further conduct experiments to evaluate the effect of the different batch size. Table 2 (bottom) shows that the model with the batch size of 1 achieves just

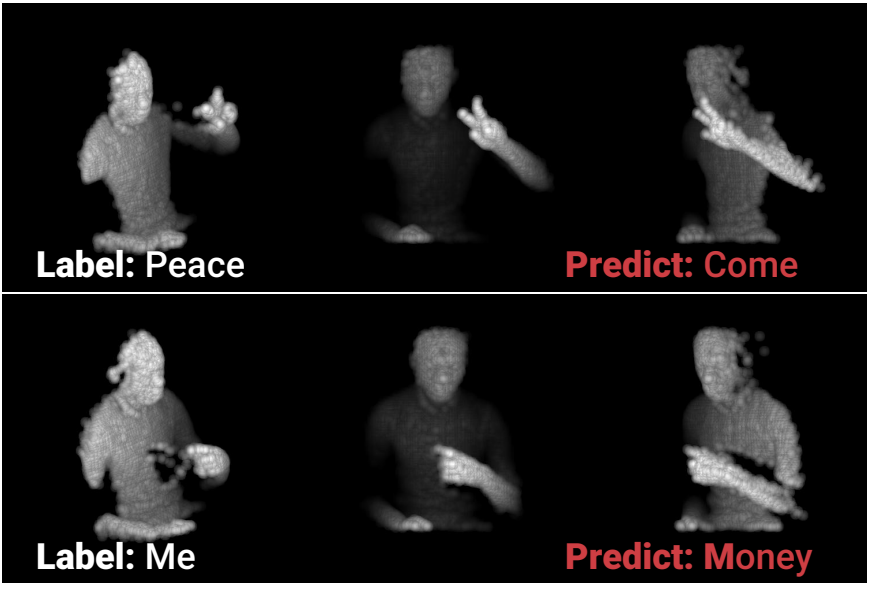


Figure 6: Examples showing the misrecognized gestures by our network. Each column shows the different viewpoint of misrecognized gestures. The network misrecognizes gestures that are similar in terms of hands' location. For example, the gesture "Me" and "Money" where the location of hand is approximately the same, the network is unable to distinguish between the two.

$\sim$ 20 % accuracy. The performance grows as we increase the batch size. However, there is no performance gain of the models trained with the batch size larger than 32.

5 Conclusions

We proposed the network with attention modules that takes point clouds as input for gesture recognition. The network learned to pay attention to points that convey gesture meaning. The gather and scatter operations are also proposed in order to downsample and upsample point clouds in the feature space. Consecutive point cloud frames are concatenated in order to add the temporal dimension to the input. Using both spatial and temporal information can improve the robustness of the network. We demonstrated the effectiveness of our network with a common Japanese gestures dataset. The network achieved state-of-the-art performance on this dataset. We have shown that the attention modules guided the network to focus on particular points in the input. The analysis on architecture design and parameter choices of our network are discussed.

6 Acknowledgment

This work is partially supported by JSPS Grant-in-Aid 16H06536.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A System for Large-Scale Machine Learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Maryam Asadi-Aghbolaghi, Albert Clapés, Marco Bellantonio, Hugo Jair Escalante, Víctor Ponce-López, Xavier Baró, Isabelle Guyon, Shohreh Kasaei, and Sergio Escalera. Deep learning for action and gesture recognition in image sequences: A survey. In *Gesture Recognition*, pages 539–578. Springer, 2017.

[3] Kanad K Biswas and Saurav Kumar Basu. Gesture recognition using microsoft kinect®. In *International Conference on Automation, Robotics and Applications*, pages 100–103, 2011.

[4] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *IEEE conference on computer vision and pattern recognition*, pages 3640–3649, 2016.

[5] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

[6] Xiao Chu, Wei Yang, Wanli Ouyang, Cheng Ma, Alan L Yuille, and Xiaogang Wang. Multi-context attention for human pose estimation. *arXiv preprint arXiv:1702.07432*, 2017.

[7] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[8] Paul Hongsuck Seo, Zhe Lin, Scott Cohen, Xiaohui Shen, and Bohyung Han. Progressive attention networks for visual attribute prediction. *arXiv preprint arXiv:1606.02393*, 2016.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[11] Byeongkeun Kang, Subarna Tripathi, and Truong Q Nguyen. Real-time sign language fingerspelling recognition using convolutional neural networks from depth map. In *IAPR Asian Conference on Pattern Recognition*, pages 136–140, 2015.

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

[13] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3D point cloud models. In *IEEE International Conference on Computer Vision*, pages 863–872, 2017.

[14] Anat Levin and Yair Weiss. Learning to combine bottom-up and top-down segmentation. In *European conference on computer vision*, pages 581–594, 2006.

[15] Chaoyu Liang, Yonghong Song, and Yuanlin Zhang. Hand gesture recognition using view projection from point cloud. In *IEEE International Conference on Image Processing*, pages 4413–4417, 2016.

[16] Liwei Liu, Junliang Xing, Haizhou Ai, and Xiang Ruan. Hand posture recognition using finger geometric feature. In *International Conference on Pattern Recognition*, pages 565–568, 2012.

[17] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499, 2016.

[18] Kai Nickel and Rainer Stiefelhagen. Visual recognition of pointing gestures for human–robot interaction. *Image and Vision Computing*, 25(12):1875–1884, 2007.

[19] Joshua Owoyemi and Koichi Hashimoto. Spatiotemporal learning of dynamic gesture from 3D point cloud data. In *IEEE International Conference on Robotics and Automation*, pages 5929–5934, 2018.

[20] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen. Sign language recognition using convolutional neural networks. In *Workshop at the European Conference on Computer Vision*, pages 572–578, 2014.

[21] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2016.

[23] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017.

[24] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun. SBNet: Sparse Blocks Network for Fast Inference. *arXiv preprint arXiv:1801.02108*, 2016.

[25] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015.

[26] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Neighbors do help: Deeply exploiting local structures of point clouds. *arXiv preprint arXiv:1712.06760*, 2017.

[27] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[28] Jesus Suarez and Robin R Murphy. Hand gesture recognition with depth images: A review. In *Ro-man, 2012 IEEE*, pages 411–417, 2012.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.

[30] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2017.

[31] Hanjie Wang, Qi Wang, and Xilin Chen. Hand posture recognition from disparity cost map. In *Asian Conference on Computer Vision*, pages 722–733, 2012.

[32] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[33] Lu Xia and JK Aggarwal. Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2834–2841, 2013.

[34] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3D joints. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–27, 2012.

[35] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[36] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2528–2535, 2010.