# Quaternion Product Units for Deep Learning on 3D Rotation Groups Supplementary File

Xuan Zhang[1], Shaofei Qin[1], Yi Xu*[1], and Hongteng Xu[2]

[1]MoE Key Lab of Artificial Intelligence, AI Institute [2]Infinia ML, Inc.
[1]Shanghai Jiao Tong University, Shanghai, China [2]Duke University, Durham, NC, USA
{floatlazer, 1105042987, xuyi}@sjtu.edu.cn, hongteng.xu@duke.edu

## 1. Effect of Angle-Axis Map on Accuracy

Table 1 shows the testing accuracy of all models using QPU-based layers with and without our angle-axis map. We perform the experiments on the FPHA dataset. Using our angle-axis map is beneficial in most situations.

Table 1: The impact of Angle-Axis map on accuracy (%).

| Model | w/ angle-axis | w/o angle-axis |
|---|---|---|
| QMLP-LSTM | **76.17** | 73.04 |
| QMLP-LSTM-RInv | **68.00** | 64.17 |
| QAGC-LSTM | **79.13** | 75.65 |
| QAGC-LSTM-RInv | 78.44 | **78.96** |
| QDGNN | **82.26** | 80.87 |
| QDGNN-RInv | **76.35** | 74.26 |

## 2. Effect of Rotation Data Augmentation

As we mentioned in our main paper, data augmentation can also be used to enhance rotation robustness. To test the effect of rotation data augmentation, we train AGC-LSTM and QAGC-LSTM-RInv on an augmented NTU dataset, whose augmented samples are rotated randomly around $y$-axis[1] and test them on the testing set with arbitrary rotations ($i.e.$, DA/AR). Table 2 summarizes the results. We can find that applying QPU is more effective than applying data augmentation on enhancing rotation robustness — the gains of testing accuracy caused by using QPU is much higher than those caused by training with augmented data.

*The corresponding author of this paper is Yi Xu (xuyi@sjtu.edu.cn). This work was supported in part by NSFC 61671298, STCSM 18DZ2270700, and 111 Plan B07022.

[1]Here $y$-axis is the axis parallel with actor's spine.

Table 2: Comparisons on the gain of testing accuracy (%).

| Model | AR | DA/AR | Gain from DA |
|---|---|---|---|
| AGC-LSTM | 55.17 | 66.64 | +11.47 |
| QAGC-LSTM-RInv | **89.92** | **90.07** | +0.15 |
| Gain from QPU | **+34.75** | **+23.43** | |

## 3. Training Procedures of Models

We show in this section the training and evaluation loss of all tested models in Figure 1, which verifies the convergence of our training method. The loss is derived from the experiments on the NTU dataset under the NR setting in the main paper. In all cases, evaluation losses of our QPU-based models, especially those with suffix "RInv", are comparable with those of real-valued models under the NR setting. Although their convergence rate is slightly slower than that of real-valued models, their robustness on rotations are much better, as we shown in our experiments.

## 4. Other Potential Applications

We further test our QPU for rotation-invariant point cloud classification on the ModelNet40 dataset [4], which contains normalized point clouds sampled from 40 types of CAD model. Each point cloud contains 1024 points. Following the Pointnet++ in [2, 3], we first sample 256 centroids for each point cloud. Then, for each centroid, we search 32 neighbor points in a fixed radius of 0.4. Accordingly, the relative 3D coordinates of each neighbor point with respect to its centroid, $i.e.$, $\boldsymbol{p}^{nbr} = [x, y, z]_{\text{neighbor}} - [x, y, z]_{\text{centroid}}$, is represented as a quaternion-based 3D rotation, $i.e.$, $q = [\cos(\theta), \sin(\theta)\boldsymbol{u}]$ by letting $\theta = \frac{\pi}{2} \cdot \frac{\|\boldsymbol{p}^{nbr}\|_2}{R}$ and $\boldsymbol{u} = \frac{\boldsymbol{p}^{nbr}}{\|\boldsymbol{p}^{nbr}\|_2}$, where $R$ is the searching radius. Inspire by [1], we cyclically sort each set of the neighbor points according to their rotation order around the vector from the

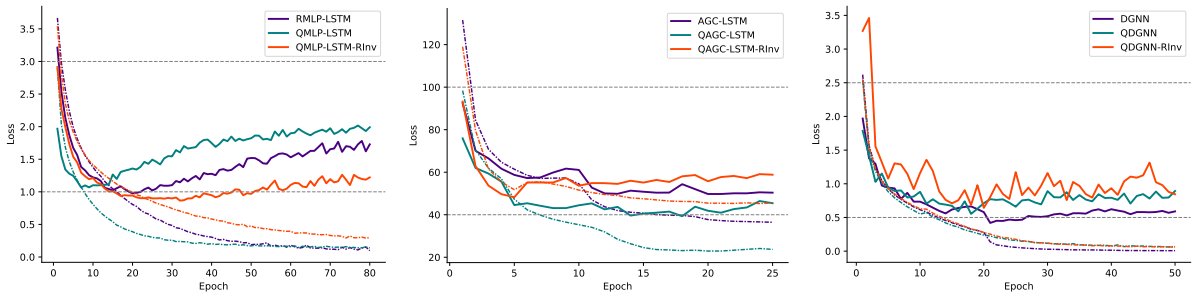


Figure 1: Comparisons on learning process. From left to right: MLP-LSTM, AGC-LSTM, DGNN.

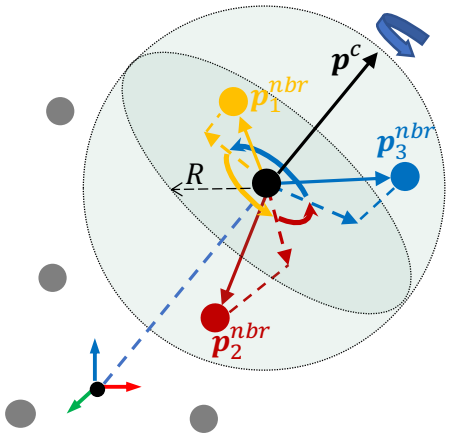


Figure 2: Illustration of neighbor points construction. For each centroid we group neighbor points within the radius $R$. $\boldsymbol{p}^c$ is the vector from the origin to a centroid. $\boldsymbol{p}_{1,2,3}^{nbr}$ are vectors from the centroid to its neighbor points. We sort them cyclically by the clockwise rotation order of their projections on the plane orthogonal to $\boldsymbol{p}^c$. In this case, $\boldsymbol{p}_1^{nbr}$'s next point is $\boldsymbol{p}_2^{nbr}$, $\boldsymbol{p}_2^{nbr}$'s next point is $\boldsymbol{p}_3^{nbr}$ and $\boldsymbol{p}_3^{nbr}$'s next point is $\boldsymbol{p}_1^{nbr}$.

origin to the centroid. Then for each point we concatenate its quaternion with the next 7 consecutive quaternions in the sorting so that each point contains information of its local structure. Figure 2 illustrates the data preprocessing steps mentioned above.

For each point cloud, we take its quaternions as input and replace the first Set Abstration layer in the Pointnet++ [2, 3] with a QMLP module. Specifically, the QMLP contains a QPU-based FC layer with $8\times4$ input channels and 64 output channels and a real-valued FC layer with 64 input channels and 128 output channels. For each set of neighbor points, we pass its quaternion through this QMLP. Two variants are tested, the first one is the model without rotation-invariance where we keep both the real parts and the imaginary parts of the outputs of QPU-based FC layer, the second one is the model with rotation-invariance where we multiply the output channels of QPU-based FC layer by 4 and only keep the real parts of the outputs of QPU-bsed FC layers. We use a max-pooling layer to aggregate the real-valued outputs as the feature of the corresponding centroid. Finally, we obtain the representation of the point cloud by passing these centroids' features through two other Set Abstraction layers, each of which is composed of a MLP and a max-pooling layer.

We train our models without rotation augmentation and test them under two settings: $i$) without arbitary rotation (NR), $ii$) with arbitary rotation (AR). Using these features, we achieve 80.1% test accuracy for the rotation-invariant model under both settings. For the model without rotation-invariance we achieve 90.0% test accuracy under NR setting and 21.4% under AR setting. These results prove that our QPU together with our proposed point cloud representation can learn efficiently from point cloud data and that rotation-invariant classification can be achieved by keeping the real part of QPU's output. We also tested our rotation-invariant model without the rotation ordering mentioned above and only achieve 75.9% accuracy under both NR and AR settings. This shows that the proposed rotation sorting is important for QPU to capture the structural information in neighbor point clouds. We believe the accuracy can be further improved with more dedicated network architecture. We also see QPU's potential in point cloud pose estimation.

## References

[1] Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *CVPR*, 2019. 1

[2] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 1, 2

[3] Erik Wijmans. Pointnet++ pytorch. `https://github.com/erikwijmans/Pointnet2_PyTorch`, 2018. 1, 2

[4] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 1