Interpretable Rotation-Equivariant Quaternion Neural Networks for 3D Point Cloud Processing

Presenter: Angxiao Yue

September 18, 2024

Outline

Background

- 3D data structures: Point cloud, Molecular, Protein...
  - $g(t \circ \mathbf{x}) = t \circ g(\mathbf{x})$, g is function, model..., t is transformation.
  - Equivariance: $\underbrace{\underbrace{\text{rotation, translation}}_{\text{SE(3) space}}\text{, reflection}}_{\text{E(3) space}}$, permutation
  - **Invariance is a special case of equivariance.**
- Traditional neural network layers for point cloud are not rotation equivariant.

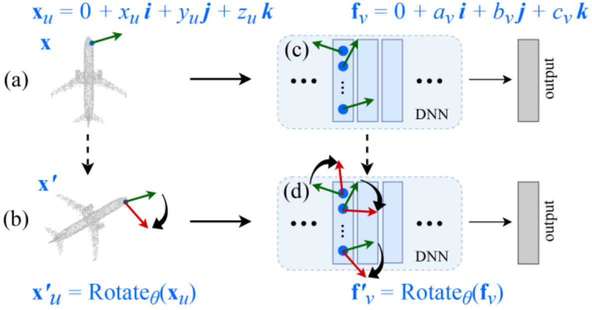
Operation	Rotation-equivariance of features	Rotation-equivariance of feature gradients	Rotation-invariance of the training	Permutation-invariance of features
Convolution	×	×	×	–
ReLU	×	×	×	–
Batch-normalization	×	×	×	–
Max-pooling	×	×	×	–
Dropout	×	×	×	–
Farthest point sampling [30]	invariance	invariance	✓	×
k-NN-search-based grouping [49]	invariance	invariance	✓	✓
Ball-query-search-based grouping [30]	invariance	invariance	✓	×
Density estimation [49]	invariance	invariance	✓	✓
3D coordinates weighting [49]	×	×	×	✓
Graph construction [45]	invariance	invariance	✓	✓

Background

Rotation equivariance is fine-grained in this work.

$$g(t \circ \mathbf{x}) = t \circ g(\mathbf{x}) \text{ , here } t \text{ is the rotation.}$$

- **1: What is the rotation-equivariance of features?**



Don't need data augmentation.

Background

- **2: What is the rotation-equivariance of feature gradients?**

  $$t \circ \frac{\partial \mathsf{Loss}(\mathbf{x})}{\partial \mathbf{f}_l} = \frac{\partial \mathsf{Loss}(t \circ \mathbf{x})}{\partial \mathbf{f}_l},$$ **here $\mathsf{Loss}(\mathbf{x})$ needs to be rotation invariant.**

  This is ensured by layer-wise equivariance:

  $$t \circ \left(\frac{\partial \mathbf{f}_l(\mathbf{x})}{\partial \mathbf{f}_{l-1}}^{\top} \nabla_{f_l} \mathsf{Loss}(\mathbf{x}) \right) = \frac{\partial \mathbf{f}_l((t \circ \mathbf{x})}{\partial \mathbf{f}_{l-1}}^{\top} \left(t \circ \nabla_{f_l} \mathsf{Loss}(\mathbf{x}) \right)$$

Background

- **3: What is the rotation-invariance of the training?**
  This is ensured by rotation-equivariance of feature gradients:

$$\arg\min_w \mathsf{Loss}(t \circ \mathbf{x}, w) = \arg\min_w \mathsf{Loss}(\mathbf{x}, w), \text{ here } w \text{ is the parameter.}$$

$$\frac{\partial \mathsf{Loss}(\mathbf{x})}{\partial w_l} = \frac{\partial \mathsf{Loss}(t \circ \mathbf{x})}{\partial w_l}$$

  Training stability.

Outline

Background

Contributions

Method

Experiment

Discussion

Contributions

Motivated by the problem of traditional real-valued layers and the advantages of rotation equivariance...

- This study proposes a set of generic rules to revise existing neural networks for 3D point cloud processing to rotation equivariant **quaternion** neural networks (REQNNs).
- Experiments have shown that REQNNs outperform traditional neural networks in both terms of classification accuracy and robustness on rotated testing samples.

Outline

Preliminary

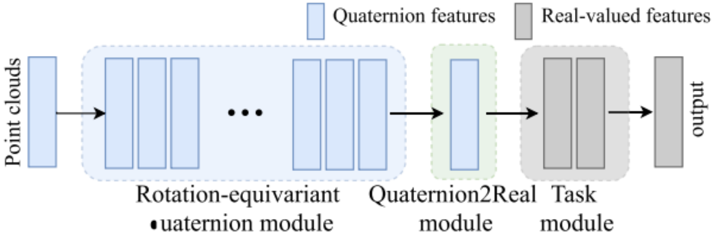
- A quaternion: $\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k \in \mathbb{H}$ is a hyper-complex number.
- $i^2 = j^2 = k^2 = ijk = -1$ and $ij = k, jk = i, ki = j, ji = -k, kj = -i,$ and $ik = -j$.
- The multiplication of two quaternions is **non-commutative**.
- The conjugation of $\mathbf{q}$ is defined as $\overline{\mathbf{q}} = q_0 - q_1 i - q_2 j - q_3 k$.
- Unit quaternion: $\|\mathbf{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$.
- Polar decomposition: $\mathbf{q} = e^{\mathbf{o}\frac{\theta}{2}} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}(o_1 i + o_2 j + o_3 k)$.
- A rotation's orientation is $[o_1, o_2, o_3]^\top$, angle is θ, the according quaternion is

$$\mathbf{R} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}(o_1 i + o_2 j + o_3 k)$$

If applied on another quaternion q

$$\mathbf{q}' = \mathbf{R}\mathbf{q}\overline{\mathbf{R}}$$

Framework



- **Rotation-Equivariant Quaternion Module.**
- **Quaternion2Real module.**
  Given each v-th element of a quaternion feature, $\mathbf{f}_v = 0 + a_v i + b_v j + c_v k$, $\|\mathbf{f}_v\|^2 = a_v^2 + b_v^2 + c_v^2$. $\left[\|\mathbf{f}_1\|^2, \|\mathbf{f}_2\|^2, \ldots, \|\mathbf{f}_d\|^2\right]^\top \in \mathbb{R}^d$, which are rotation invariant.
- **Task module.**
  The last few layers of real-valued model.

Rotation-Equivariant Quaternion Module

Initialization: convert coordinate $x \in \mathbb{R}^3$ to pure quaternion $f_0 \in \mathbb{H}$. In the intermediate layer: $\mathbf{f} = \left[f_1, \ldots, f_d\right]^\top \in \mathbb{H}^d$.

- **Convolution**:
  $w \in \mathbb{R}^{D \times d}$

$$\text{Conv}(\mathbf{f}) = \mathbf{w} \otimes \mathbf{f} = \begin{bmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{D1} & \cdots & w_{Dd} \end{bmatrix} \otimes \begin{bmatrix} f_1 \\ \vdots \\ f_d \end{bmatrix} = \begin{bmatrix} (w_{11} f_1 + \cdots + w_{1d} f_d) \\ \vdots \\ (w_{D1} f_1 + \cdots + w_{Dd} f_d) \end{bmatrix} \in \mathbb{H}^D$$

- **ReLU**
  f_v is the v-th element of $\mathbf{f}$. $(v = 1, 2, \ldots d)$.

$$\text{ReLU}(f_v) = \frac{\|f_v\|}{\max\{\|f_v\|, c\}} f_v$$

Rotation-Equivariant Quaternion Module

- **Batch-Normalization**:
  $\mathbf{f}^{(i)} \in \mathbb{H}^d$ denotes the feature of the i-th sample in the batch.

$$\text{norm}(f_v^{(i)}) = \frac{f_v^{(i)}}{\sqrt{\mathbb{E}_j[\|f_v^{(j)}\|^2] + \epsilon}},$$

- **Max-Pooling**:

$$\text{maxPool}(\mathbf{f}) = f_{\hat{v}} \quad \text{s.t.} \quad \hat{v} = \arg\max_{v=1,\ldots,d} \|f_v\|$$

- **Dropout**:
  If the v-th quaternion element $(\mathbf{f}_v = 0 + ai + bj + ck)$ is dropped, then we set $\mathbf{f}_v = 0 + 0i + 0j + 0k$.

Rotation-Equivariant Quaternion Module

- **3D Coordinates Weighting**:
  Use local structure to reweight features. Given a 3D point $x_0 \in \mathbb{R}^3$ and its K neighbors $\{x_1, \ldots, x_K\} \in \mathbb{R}^{3\times K}$.

$$F' = FW^\top, F \in \mathbb{H}^{d\times K}, W \in \mathbb{R}^{M\times K}$$

$$W = \text{perceptron}([x_1 - x_0, \ldots, x_K - x_0]^\top).$$

  Now, make W or relative coordinates $[x_1 - x_0, \ldots, x_K - x_0]^\top$ be rotation-invariant. **Use the PCA to find out first three eigenvectors of the point cloud i.e., $e_1, e_2, e_3 \in \mathbb{R}^3$.**

$$x'_k = [x_k^\top e_1, x_k^\top e_2, x_k^\top e_3]^\top \text{ is invariant.}$$

$$[x'_1 - x'_0, \ldots, x'_K - x'_0]^\top \text{ is also invariant.}$$

Outline

Experiment

- ModelNet40, 3D MNIST, ShapeNet

Architecture	ModelNet40 dataset			3D MNIST dataset			ShapeNet dataset		
	Ori. DNN trained w/o rotations	Ori. DNN trained w/ rotations	REQNN trained w/o rotations	Ori. DNN trained w/o rotations	Ori. DNN trained w/ rotations	REQNN trained w/o rotations	Ori. DNN trained w/o rotations	Ori. DNN trained w/ rotations	REQNN trained w/o rotations
PointNet++[5]	25.87[7]	29.25	**62.03**	44.19	51.48	**72.01**	41.60	43.53	**94.42**
DGCNN[6]	32.08[7]	33.78	**84.57**	45.90	50.00	**85.07**	44.06	50.39	**96.90**
PointConv	25.01	26.46	**81.93**	45.51	48.08	**85.71**	37.03	39.60	**97.59**

"Ori. DNN trained w/o rotations" indicates the original neural network learned without rotations. "Ori. DNN trained w/ rotations" indicates the original neural network learned with the y-axis rotation augmentation (the y-axis rotation augmentation has been widely applied in [30], [45]). "REQNN trained w/o rotations" indicates the REQNN learned without rotations. Note that the accuracy of shape classification reported in [30], [45], [49] was obtained under the test without rotations. The accuracy reported here was obtained under the test with rotations. Therefore, it is normal that the accuracy in this paper is lower than the accuracy in those papers.

- NR/AR

Method	**NR/NR** (do **not** consider rotation) in testing)	**NR/AR** (consider rotation in testing)
PointNet [29]	88.45	12.47
PointNet++ [30]	89.82	21.35[7]
Point2Sequence [24]	92.60	10.53
KD-Network [21]	86.20	8.49
RS-CNN [25]	92.38	22.49
DGCNN [45]	**92.90**	29.74[7]
PRIN [53]	80.13	68.85
QE-CapsuleNet [59]	74.73	74.07
REQNN (revised from DGCNN[6])	84.64 ≈	**84.57**

NR/NR denotes that DNNs were learned and tested with **N**o **R**otations. **NR/AR** denotes that DNNs were learned with **N**o **R**otations and tested with **A**rbitrary **R**otations. Experimental results show that the REQNN exhibited the highest rotation robustness.

Experiment

- The number of parameters.

	PointNet++[5] [30]			DGCNN[6] [45]			PointConv [49]		
	#Param.(M)	#Feat. dim.(M)	#FLOPs(G)	#Param.(M)	#Feat. dim.(M)	#FLOPs(G)	#Param.(M)	#Feat. dim.(M)	#FLOPs(G)
Ori.	1.48	9.21	0.86	2.86	16.78	3.76	19.57	13.04	1.22
REQNN	1.47	26.44	2.51	2.86	39.85	9.04	20.61	33.66	3.58

All neural networks were tested on the ModelNet40 dataset. "FLOPs" denotes the floating-point operations per second.

- Average training time (s).

	PointNet++[5]	DGCNN[6]	PointConv
Ori.	53.04	66.32	62.35
REQNN	132.53	269.51	140.24

- Rotation robustness

Architecture	ModelNet40 dataset			3D MNIST dataset			ShapeNet dataset		
	Ori. DNN trained w/o rotations	Ori. DNN trained w/ rotations	REQNN trained w/o rotations	Ori. DNN trained w/o rotations	Ori. DNN trained w/ rotations	REQNN trained w/o rotations	Ori. DNN trained w/o rotations	Ori. DNN trained w/ rotations	REQNN trained w/o rotations
PointNet++[5]	0.034	0.050	**1.0**	0.044	0.051	**0.987**	0.031	0.038	**0.999**
DGCNN[6]	0.027	0.031	**0.999**	0.056	0.019	**0.972**	0.012	0.028	**1.0**
PointConv	0.028	0.034	**1.0**	0	0.029	**0.971**	0.007	0.007	**1.0**

A high failure rate indicated that the DNN was robust to rotation attacks. Failure rates of REQNNs were 1.0 or very close to 1.0 (more than 0.97). Sometimes, the failure rate was not exactly 1.0, which was caused by the accumulation of tiny systematic errors of computation in the forward propagation process. In comparison, traditional DNNs could be easily attacked by rotations.

Outline

Discussion

Although it was accepted by TPAMI, its quality did not meet my expectations.

- It resembles a real-valued neural network disguised as a non-real one.
- The learning process seems challenging to extend to 3D data beyond point clouds.
- The training time has doubled, and the parameter count is comparable or even slightly higher.
- Hamilton product was not utilized, raising doubts about its capability to capture geometric information.