Vector Neurons: A General Framework for SO(3)-Equivariant Networks

Wed. reading group

Presenter: Yong-Min Shin

yongminshin.simple.ink

Dent et al., Vector Neurons: A General Framework for SO(3)-Equivariant Networks, ICCV'21

18th Oct. 2023

수학계산학부(계산과학공학)
School of Mathematics and Computing
(Computational Science and Engineering)
MIDaS LAB
Machine Intelligence & Data Science

1. Simple survey on [1]PIML 2. Understanding [2]VNN

Objective for the talk

[1]VNN: Vector neural networks
[2]PIML: Physics-informed machine learning

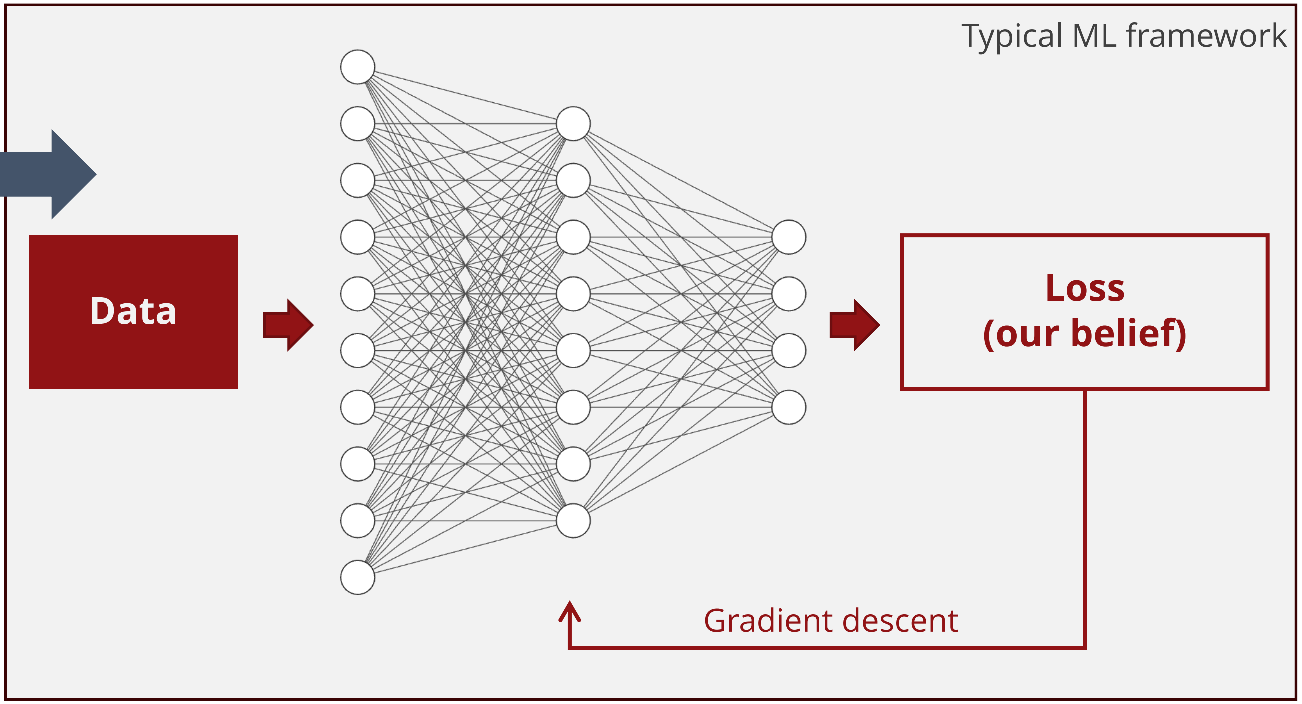
[1]VNNs is part of the field of [2]PIML, a broad area of ML that integrates prior physics knowledge into neural network models.

P▯ML: Field of machine learning that integrates prior physics knowledge [1].

Physical (natural) priors

$$F = m\frac{\partial^2 x}{\partial t^2} \qquad \frac{\partial u}{\partial t} - \nabla u = 0 \quad \cdots$$

→ **What** (task-suitable) priors are injected
& **How** it is injected varies for different subfields.



[1]VNN: Vector neural networks
[2]PIML: Physics-informed machine learning

[1] Meng et al., When physics meets machine learning: A survey of physics-informed machine learning, arXiv (2022)

[1]VNNs is part of the field of [2]PIML, a broad area of ML that integrates prior physics knowledge into neural network models.

PIML includes various subfields, including PINN, PICV, PIGL, Operator learning etc.

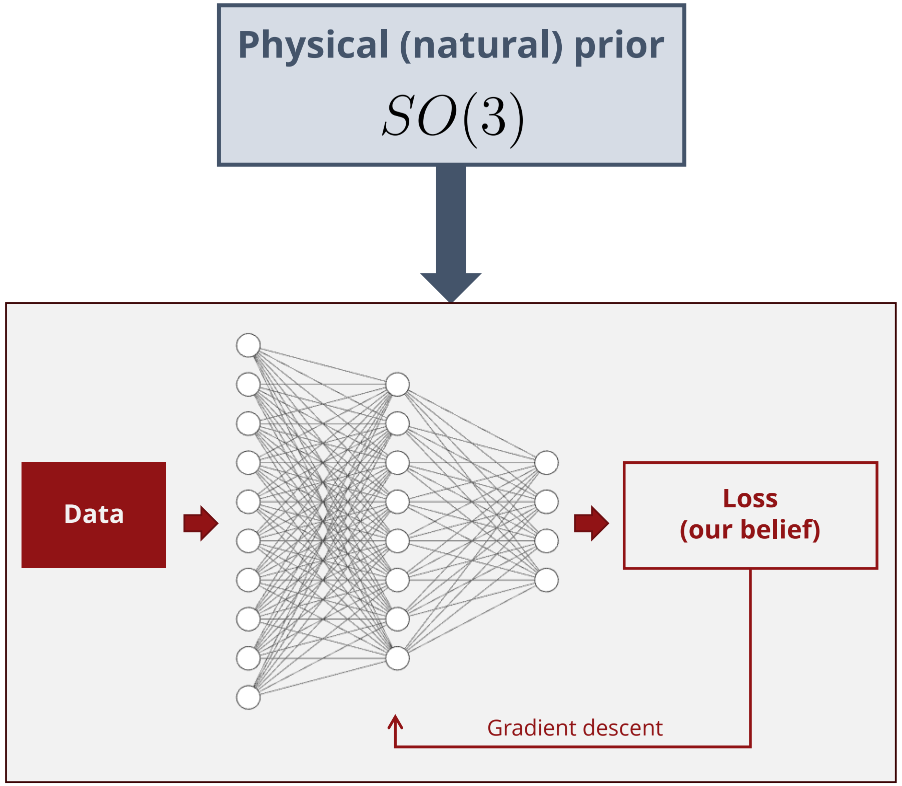
PINN (Physics-informed neural networks)	Neural networks that encode model equations (e.g., PDE) [1] Typically, PDEs are injected through the loss function	(Raissi et al., 2019) [2]
PICV (Physics-informed computer vision)	PIML specifically dedicated to CV models and applications (e.g., imaging, super-resolution, segmentation) [3]	(Yuan et al., 2021) [4]
PIGL (Physics-informed graph learning)	PIML for graph learning (e.g., molecular representation, dynamic particle simulation) [2]	(Sanchez-Gonzalez et al., 2020) [5]
Operator learning	Using neural networks to learn mappings between infinite dimensional function spaces [6]	DeepONets [7] & FNOs [8]

[1]VNN: Vector neural networks
[2]PIML: Physics-informed machine learning

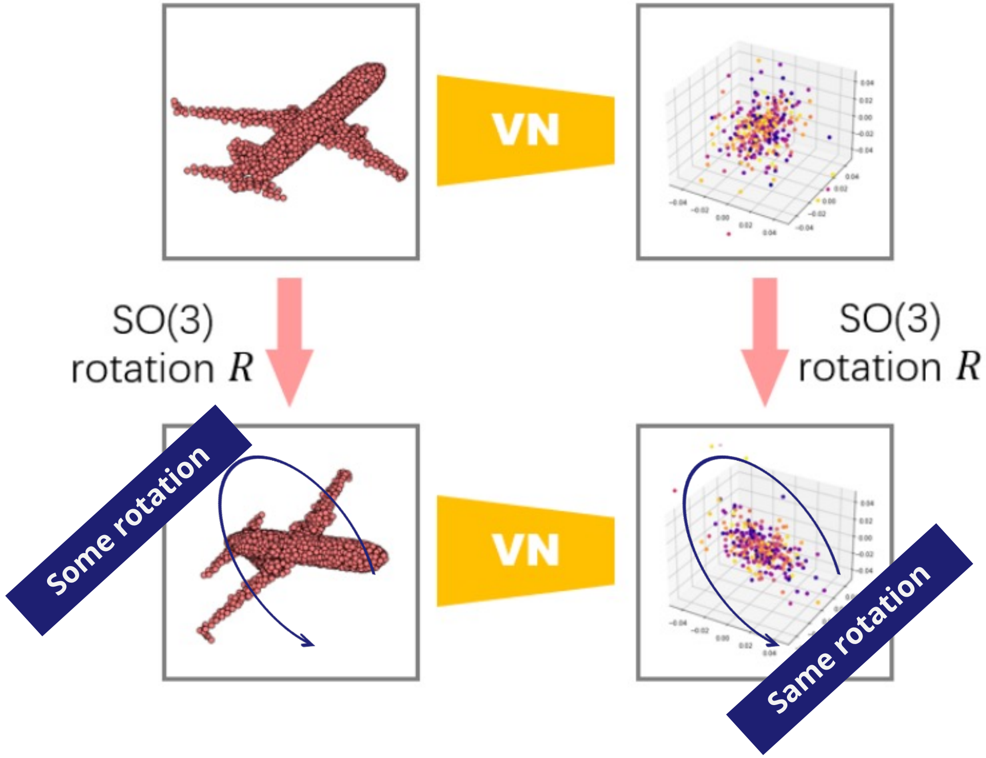
[1] Cuomo et al., Scientific machine learning through physics-informed neural networks: Where we are and what's next. J. Sci. Comput. 92(3): 88 (2022)
[2] Raissi et al., Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378: 686-707 (2019)
[3] Banerjee et al., Physics-informed computer vision: A review and perspectives, arXiv (2023)
[4] Yuan et al., SimPoE: Simulated Character Control for 3D Human Pose Estimation, CVPR 2021
[5] Sanchez-Gonzalez et al., Learning to simulate complex physics with graph networks, ICML 2020
[6] Kovachki et al., Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs, J. Mach. Learn. Res. 24: 89:1-89:97 (2023)
[7] Lu et al., DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators, arXiv (2019)
[8] Li et al., Fourier Neural Operator for Parametric Partial Differential Equations, ICLR 2021

[1]VNNs is part of the field of [2]PIML, a broad area of ML that integrates prior physics knowledge into neural network models.

VNN integrates SO(3) symmetry into the neural network architecture.



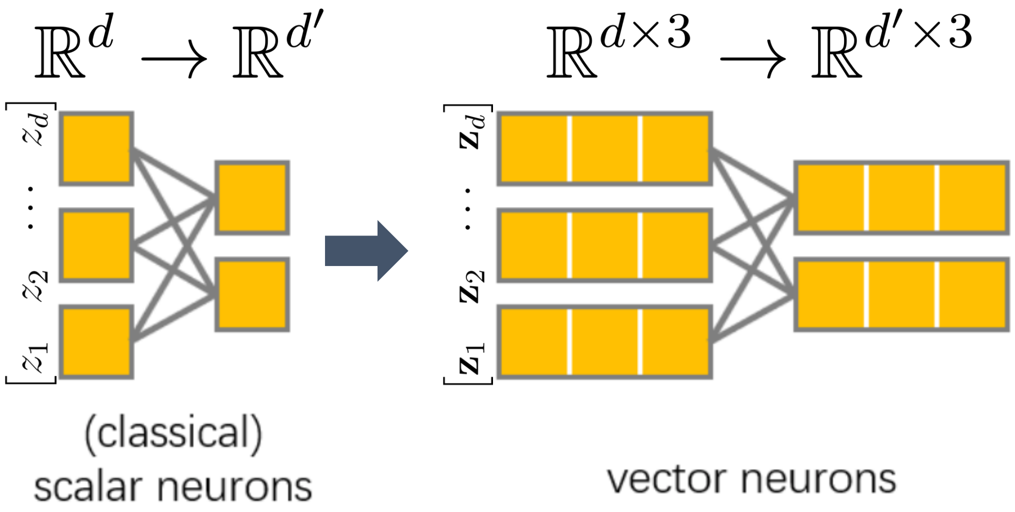
SO(3) symmetry is a natural prior for data points with no canonical ordering (ex. Point clouds).



[1]VNN: Vector neural networks
[2]PIML: Physics-informed machine learning
Right figure: Deng et al., Vector neurons: A general framework for SO(3)-equivariant networks, ICCV 2021

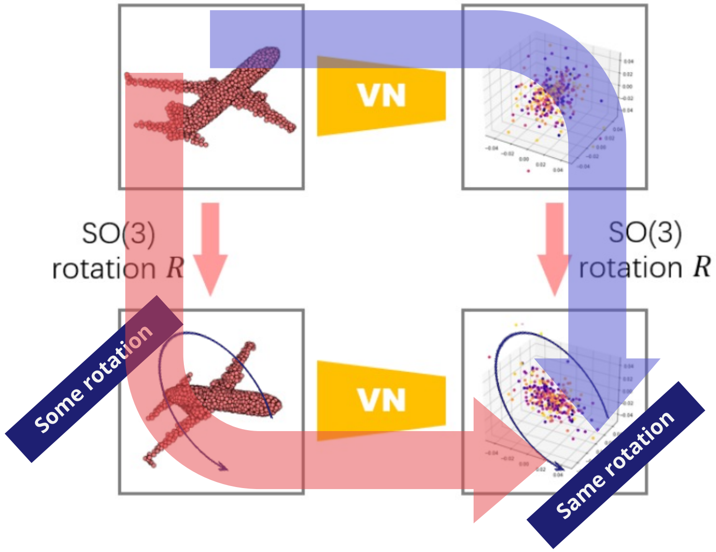
VNNs expand all neurons as a 3D vector, which produces SO(3) rotation equivariant linear layers.

VNNs treat each dimension of its representation as a 3D vector.



Making each neuron 3D has the advantage of being naturally SO(3) symmetric.

**This type of diagram is also the* *definition* *of equivariance & invariance properties*



$$f_{\text{lin}}(\mathbf{V}R;\mathbf{W}) = \mathbf{W}\mathbf{V}R = f_{\text{lin}}(\mathbf{V};\mathbf{W})R = \mathbf{V}'R$$

This seems trivial, but cannot be done for scalar neurons (see **red** box; dimensions cannot be matched with the rotation matrix ***R***.)

***Example: Counterclockwise rotation w.r.t. positive z axis is an element in group SO(3).*

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \in SO(3)$$

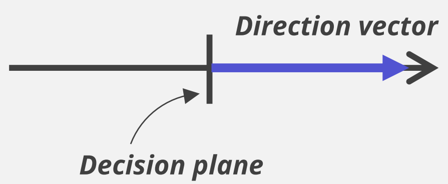
Left figure: Deng et al., Vector neurons: A general framework for SO(3)-equivariant networks, ICCV 2021

VNNs also provide a generalized ReLU function that is compatible with 3D neurons while being SO(3) equivariant.

A classical generalization procedure:
Observation → Abstraction → Generalization.

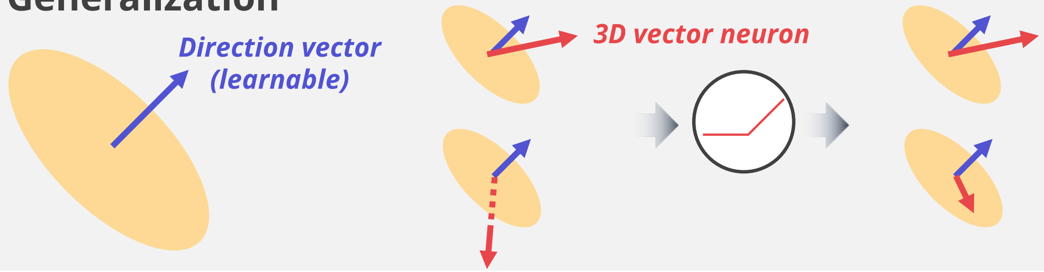
Observation

Abstraction: How to decide when to 'clip'?

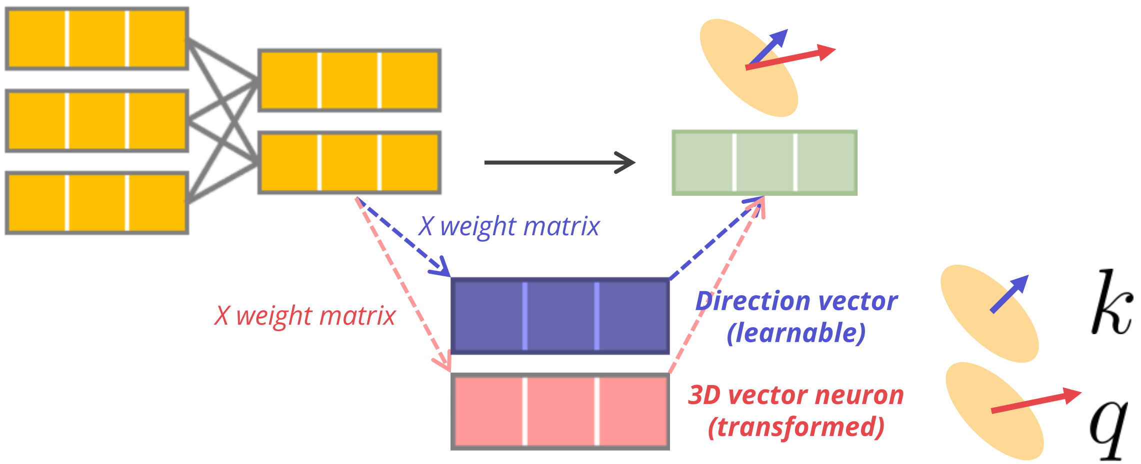


1. A *decision plane* is set
2. Vectors 'under' the decision plane gets *projected to the plane*

Generalization



Straightforward calculation shows that the new ReLU is rotation equivariant.



We then define the output VN as:

$$\boldsymbol{v}' = \begin{cases} \boldsymbol{q} & \text{if } \langle \boldsymbol{q}, \boldsymbol{k} \rangle \geqslant 0 \\ \boldsymbol{q} - \left\langle \boldsymbol{q}, \frac{\boldsymbol{k}}{\|\boldsymbol{k}\|} \right\rangle \frac{\boldsymbol{k}}{\|\boldsymbol{k}\|} & \text{otherwise,} \end{cases}$$

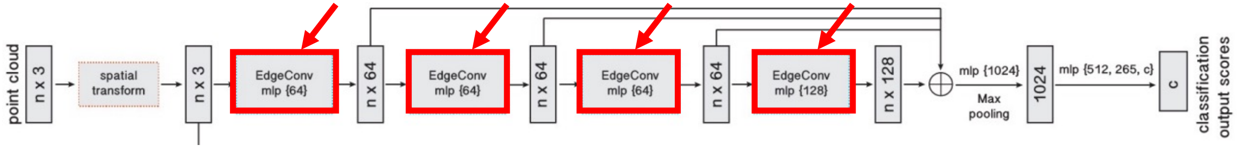
$$\langle \boldsymbol{q}\boldsymbol{R}, \boldsymbol{k}\boldsymbol{R} \rangle = \langle \boldsymbol{q}, \boldsymbol{k} \rangle$$

**Think of the previous example element of SO(3):* $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \in SO(3)$

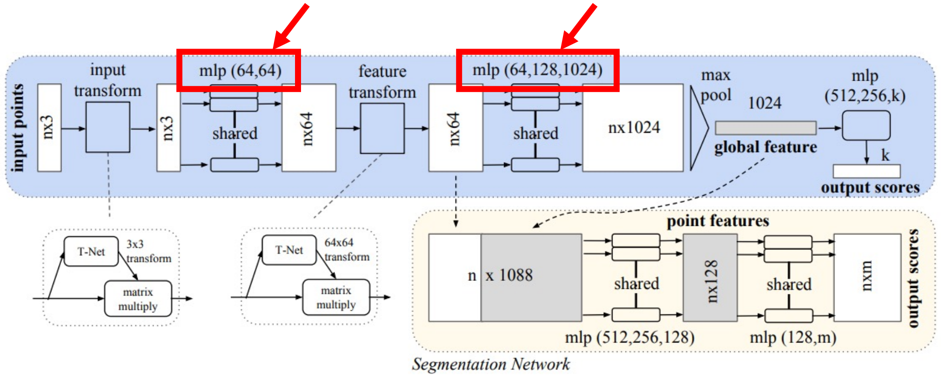
VNNs expand all neurons as a 3D vector, which produces SO(3) rotation equivariant layers.

By replacing linear layers (effectively, MLPs), it is used in other architectures as well.

VN-DGCNN: Replace MLPs with VNNs



VN-PointNet: Replace MLPs with VNNs



Train time rotation augmentation — *Test* time rotation augmentation

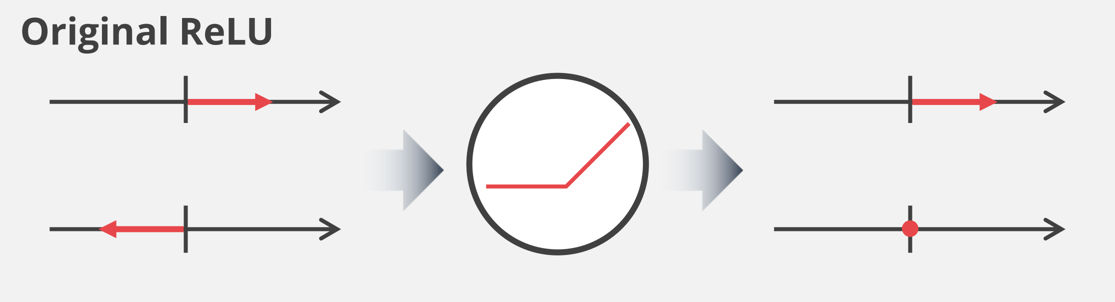
Methods	z/z	$z/SO(3)$	$SO(3)/SO(3)$
Point / mesh inputs			
PointNet [25]	85.9	19.6	74.7
DGCNN [35]	90.3	33.8	88.6
VN-PointNet	77.5	77.5	77.2
VN-DGCNN	89.5	**89.5**	**90.2**

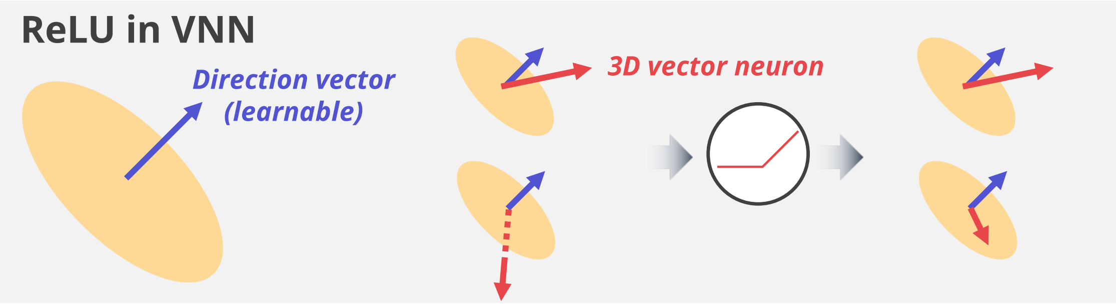
Table 1: Point cloud classification

- Learning rotation with **proper** augmentation is very critical (see z/z vs. z/SO(3)).
- In z/SO(3), **VNNs effectively generalize to other rotations** even when only z-axis rotations were introduced during training (see z/z vs. z/SO(3) & z/SO(3)).
- **Additional rotation prior is still beneficial** even if the training set augmentation is proper (see SO(3)/SO(3)).

Upper left figure: Wang et al., Dynamic graph CNN for learning on point clouds, ACM Tran. Graph., 38(5):1-12 (2019)
Lower right figure: Qi et al., PointNet: Deep learning on point sets for 3D classification and segmentation, CVPR 2017
Lower left table: Deng et al., Vector neurons: A general framework for SO(3)-equivariant networks, ICCV 2021

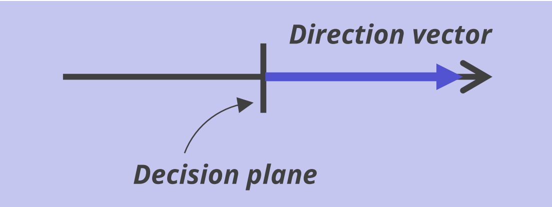
Design choice of learnable decision boundaries is not warranted and requires further discussion.



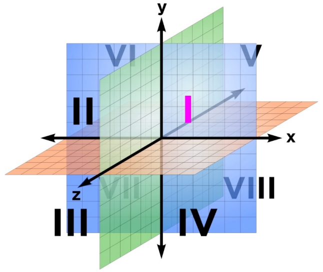


*Decision plane is **fixed** to the positive direction*

*Decision plane is **learnable** with additional model parameters*



- Previous neural networks were just fine with fixed decision planes: Do we really need learnable decision planes?
- Can we fix the direction vector to a trivial one (i.e., [1, 1, 1]) and still get the same results?
- Can we even set octant-based activation regions? (May need to perform multiple procedures in parallel)
- Apparently, no analysis & ablation has been done by the authors



Besides this point, VNN is a solid research with good practical implications.

Bottom right image: https://www.youtube.com/watch?app=desktop&v=5sJdfciNM20

Takeaway messages

- *VNNs is a PIML research that incorporates SO(3) symmetry into neural networks*
- *Core idea: Represent each neuron as 3D vectors*
- *Design of compatible ReLUs and other techniques makes the model very applicable*
- *Mainly used as a building block for other architectures*

Thank you!