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

(Supplementary Material)

7. Discussions

In this section, we discuss some extensions, alternatives, and explanations to the VN layers in Section 3.

7.1. Non-linearity

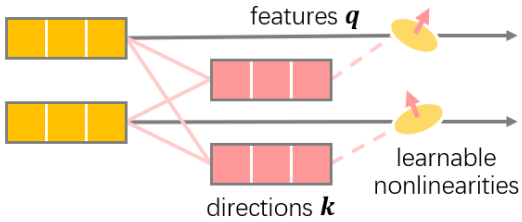

Figure 7: A detached non-linear layer without built-in linear layer.

Linear and non-linear layers – Fig. 7. The VN-ReLU defined in Section 3.2 already consists of a built-in linear layer $\boldsymbol{q} = \mathbf{W}\boldsymbol{V}$ (5) and the the non-linearity is applied to this learned feature $\boldsymbol{q}$. An alternative to this is to construct linear and non-linear layers separately, where the non-linearity is directly applied to each input vector channel $\boldsymbol{v} \in \boldsymbol{V}$ by

$$\boldsymbol{v}' = \begin{cases} \boldsymbol{v} & \text{if } \langle \boldsymbol{v}, \boldsymbol{k} \rangle \geqslant 0 \\ \boldsymbol{v} - \left\langle \boldsymbol{v}, \frac{\boldsymbol{k}}{\|\boldsymbol{k}\|} \right\rangle \frac{\boldsymbol{k}}{\|\boldsymbol{k}\|} & \text{otherwise,} \end{cases} \tag{24}$$

Detaching the linear layer from non-linearity allows more flexibility in constructing neural networks and, in practice, gives better results in some cases. However, this also doubles the network depth and can lead to longer training time compared to the entangled linear-ReLU layer in (6). Experimental comparisons will be shown in Section 8.2.

Other non-linearities. Though we only showed how to define VN-ReLU in Section 3.2, a rich library of equivariant non-linearities can be defined in this manner using the input-dependent direction vector $\boldsymbol{k}$. An immediate extension is VN-LeakyReLU, where instead of clipping $\boldsymbol{q}_\parallel$ to zero we contract it by a factor $\alpha \in (0, 1)$. In the manner of the detached VN-ReLU in (24), the VN-LeakyReLU can be easily expressed as:

$$f_{\text{LeakyReLU}}(\boldsymbol{V}; \alpha) = \alpha \boldsymbol{V} + (1 - \alpha) f_{\text{ReLU}}(\boldsymbol{V}). \tag{25}$$

An entangled layer of VN-Linear and VN-LeakyReLU can also be defined analogous to (6). More generally, given an arbitrary non-linear scalar function $h : \mathbb{R} \to \mathbb{R}$, we can incorporate it into our VN non-linearity framework by applying it to $\boldsymbol{q}_\parallel$ along the $\boldsymbol{k}$ direction, namely,

$$\boldsymbol{v}' = \frac{h(\|\boldsymbol{q}_\parallel\|)}{\|\boldsymbol{q}_\parallel\|} \boldsymbol{q}_\parallel + \boldsymbol{q}_\perp. \tag{26}$$

7.2. Local Pooling

The VN-MAX pooling in Section 3.3 is defined across an entire pointcloud $\mathcal{V} \in \mathbb{R}^{N \times C \times 3}$, but we can also aggregate information locally via local pooling.

In the primal space. For any point $\boldsymbol{x} \in \mathcal{X}$ with feature $\boldsymbol{V} \in \mathcal{V}$ we consider its K nearest neighbours $\{\boldsymbol{x}_k\}_{k=1}^K$ in the primal space and we denote by $\boldsymbol{V}_k \in \mathcal{V}$ the corresponding feature of $\boldsymbol{x}_k$. Similar to global pooling (9), local pooling (in the primal space) is given by:

$$f_{\text{MAX}}\left(\{\boldsymbol{V}_k\}_{k=1}^K\right)[c] = \boldsymbol{V}_{k^*}[c] \tag{27}$$

$$\text{where} \quad k^*(c) = \arg\max_k \langle \mathbf{W}_k \boldsymbol{V}_k[c], \boldsymbol{V}_k[c] \rangle. \tag{28}$$

Feature space locality. As in DGCNN [38], we can also query the K nearest neighbours $\{\boldsymbol{V}_k\}_{k=1}^K$ of feature $\boldsymbol{V}_n \in \mathcal{V}$ in the feature space $\mathbb{R}^{C \times 3}$ directly, followed by local pooling (in the feature space) with exactly the same formulation as (28).

7.3. Batch Normalization

In VN-BatchNorm (10), for each input vector-list feature $\boldsymbol{V}_b$, all entries in its per-channel 2-norm $\boldsymbol{N}_b$ are non-negative, but after normalizing the distributions, the output "2-norm" $\boldsymbol{N}'_b$ can have negative entries. Geometrically, a negative entry $\boldsymbol{n}'_c \in \boldsymbol{N}'_b$ means the orientation of its corresponding vector channel is flipped, that is, $\boldsymbol{v}'_c \in \boldsymbol{V}'_b$ is in the opposite direction of $\boldsymbol{v}_c \in \boldsymbol{V}_b$.

To avoid the negative 2-norms, an alternative is to take logarithms on all entries of $\boldsymbol{N}_b$ and then apply the standard BatchNorm to $\log(\boldsymbol{N}_b)$. So the VN batch normalization becomes:

$$\boldsymbol{N}_b = \text{ElementWiseNorm}(\boldsymbol{V}_b) \in \mathbb{R}^{N \times 1} \tag{29}$$

$$\{\boldsymbol{N}'_b\}_{b=1}^B = \text{BatchNorm}\left(\{\log(\boldsymbol{N}_b)\}_{b=1}^B\right) \tag{30}$$

$$\boldsymbol{V}'_b[c] = \boldsymbol{V}_b[c] \frac{\exp(\boldsymbol{N}'_b[c])}{\boldsymbol{N}_b[c]}, \tag{31}$$

where log and exp act element-wise. However, taking log and exp brings a lot of instability and in practice can cause gradient explosion. Also, logarithms cannot be computed for vectors with zero 2-norms.

Method	I/I	I/z	I/SO(3)
PointNet	90.7	23.1	7.9
DGCNN	**92.9**	37.2	16.6
VN-PointNet	77.2	77.2	77.2
VN-DGCNN	90.0	**90.0**	**90.0**

Table 4: Test classification accuracy (%) on the ModelNet40 dataset [41] with training on aligned data. I stands for no-rotations.

Method	I/I	I/z	I/SO(3)
PointNet	78.7	36.7	30.3
DGCNN	**85.2**	43.8	36.1
VN-PointNet	73.0	73.0	73.0
VN-DGCNN	81.5	**81.5**	**81.5**

Table 5: ShapeNet part segmentation results (mIoU). Training is done on aligned data without rotation augmentation.

Non-lin	z/z	z/SO(3)	SO(3)/SO(3)
VN-PointNet			
Built-in	77.5	77.5	**77.2**
Detached	**78.2**	**78.1**	76.8
VN-DGCNN			
Built-in	89.5	89.5	**90.2**
Detached	**90.8**	**90.7**	**90.2**

Table 6: **Non-linearity –** We compare the performances of entangled linear-ReLU (or linear-LeakyReLU) layers in (6) with 2-tuples of a linear layer plus a separate non-linearity in (24). "Built-in" stands for non-linearities with built-in linear transformations, while "detached" stands for tuples of detached linear and non-linear layers in (24). In most cases, with either the VN-PointNet or the VN-DGCNN backbone, disentangling linear and non-linear layers leads to slightly better results. But this is also at the cost of a doubled network depth and a longer training time (roughly $\geqslant 1.5$ times to the entangled versions).

Pooling	z/z	z/SO(3)	SO(3)/SO(3)
VN-PointNet			
VN-MAX	76.7	76.7	**77.7**
MEAN	**77.5**	**77.5**	77.2
VN-DGCNN			
VN-MAX	88.9	89.0	88.6
MEAN	**89.5**	**89.5**	**90.2**

Table 7: **Mean and max pooling –** Comparisons between the VN-MAX aggregation defined in Section 3.3 and the standard mean aggregation (MEAN) which naturally preserves equivariance. The two aggregations give comparable results, while MEAN pooling performs slightly better than VN-MAX in more cases. Note that VN-MAX also introduces additional learnable weights compared to the mean aggregation.

VN-In	z/z	z/SO(3)	SO(3)/SO(3)
VN-PointNet			
VN-lin	75.7	75.8	75.3
VN-lin + $\overline{\boldsymbol{V}}$	77.1	77.2	76.7
VN-MLP	**78.0**	**77.8**	**77.3**
VN-MLP + $\overline{\boldsymbol{V}}$	77.5	77.5	77.2
VN-DGCNN			
VN-lin	88.8	88.8	89.8
VN-lin + $\overline{\boldsymbol{V}}$	89.7	89.7	89.7
VN-MLP	**89.9**	**89.9**	90.1
VN-MLP + $\overline{\boldsymbol{V}}$	89.5	89.5	**90.2**

Table 8: **Invariance –** Table 8 shows our ablation study on the invariant layer (VN-In) in Section 3.5. Specifically, in computing the equivariant coordinate systems $\boldsymbol{T}_n$ following (13), we compare the combinations of the following options: whether or not concatenating the global mean $\overline{\boldsymbol{V}}$ to the local feature $\boldsymbol{V}$, and whether the VN-MLP is a 3-layer VN-MLP (VN-MLP) or a single VN linear layer (VN-lin). Improvements in performance with both the global mean $\overline{\boldsymbol{V}}$ and the 3-layer VN-MLP are minor.

8. Additional Experiments

8.1. Training on Aligned Data

In Section 5, we adopt the three train/test settings z/z, z/SO(3), SO(3)/SO(3) from prior works to standardize the comparisons between different methods. However, it is also interesting to see how each method performs when trained without any augmentation (no-rotation setting I) but tested on rotated shapes. Our additional results in classification and part segmentation on I/I, I/z, and I/SO(3) are shown in Table 4 and Table 5 respectively. Compared to the z-trained settings in Table 1 and Table 2, results here further highlight the robustness of our VN networks on test-time rotations in contrast to their rotation-sensitive counterparts.

8.2. Ablation Studies

Table 6, 7, and 8 show our ablation studies on non-linearity, pooling, and the invariant layer in VN networks on ModelNet40 classification.