Efficient Continuous Group Convolutions for Local SE(3) Equivariance in 3D Point Clouds

Supplementary Material

A. Proof of Theorem 1

$\Phi_{\mathcal{F}}$ is equivaraint to SE(3) iff $\forall g \in \mathrm{SE}(3) : (\Phi_{\mathcal{F}} \circ \rho^{\mathcal{X}(g)})(f) = (\rho^{\mathcal{Y}}(g) \circ \Phi_{\mathcal{F}})(f)$.

Proof. Using the definition

$$k_{\mathrm{R}}(\mathrm{x}, \mathrm{R'}) := k(\mathrm{R}^{-1}\mathrm{x}, \mathrm{R}^{-1}\mathrm{R'}), \tag{1}$$

omitting the normalization constant $\frac{1}{|\mathcal{F}(x_j)|}$ for brevity and using $g = (\mathrm{t}_g,\ \mathrm{R}_g)$ we can write

$$\begin{aligned}
&\big[(\Phi_{\mathcal{F}} \circ \rho^{\mathcal{X}(g)})(f)\big](\mathrm{x}, \mathrm{R}) = \\
&\sum_j \sum_{(\mathrm{t}, \mathrm{R'}) \in \mathcal{F}(x_j)} f(g^{-1}(\mathrm{t}, \mathrm{R'}))k((\mathrm{x}, \mathrm{R})^{-1}(\mathrm{t}, \mathrm{R'})) = \\
&\sum_j \sum_{(\mathrm{t}, \mathrm{R'}) \in \mathcal{F}(x_j)} f(g^{-1}(\mathrm{t}, \mathrm{R'}))k_{\mathrm{R}}(\mathrm{t} - \mathrm{x}, \mathrm{R'}) = \\
&\sum_j \sum_{(\mathrm{t}, \mathrm{R'}) \in \mathcal{F}(x_j)} f(\mathrm{R}_g^{-1}(\mathrm{t} - \mathrm{t}_g), \mathrm{R}_g^{-1}\mathrm{R'})k_{\mathrm{R}}(\mathrm{t} - \mathrm{x}, \mathrm{R'}) \overset{x_j \leftarrow \mathrm{R}_g x_j + \mathrm{t}_g}{=} \\
&\sum_j \sum_{(\mathrm{t}, \mathrm{R'}) \in \mathcal{F}(\mathrm{R}_g x_j + \mathrm{t}_g)} f(\mathrm{R}_g^{-1}(\mathrm{t} - \mathrm{t}_g), \mathrm{R}_g^{-1}\mathrm{R'})k_{\mathrm{R}}(\mathrm{t} - \mathrm{x}, \mathrm{R'}) \overset{\text{equiv. of } \mathcal{F}}{=} \\
&\sum_j \sum_{(\mathrm{t}, \mathrm{R'}) \in \mathrm{R}_g \mathcal{F}(x_j) + \mathrm{t}_g} f(\mathrm{R}_g^{-1}(\mathrm{t} - \mathrm{t}_g), \mathrm{R}_g^{-1}\mathrm{R'})k_{\mathrm{R}}(\mathrm{t} - \mathrm{x}, \mathrm{R'}) = \\
&\sum_j \sum_{(\mathrm{R}_g\mathrm{t} + \mathrm{t}_g, \mathrm{R}_g\mathrm{R'}) \in \mathcal{F}(x_j)} f(\mathrm{R}_g^{-1}(\mathrm{t} - \mathrm{t}_g), \mathrm{R}_g^{-1}\mathrm{R'})k_{\mathrm{R}}(\mathrm{t} - \mathrm{x}, \mathrm{R'}) = \\
&\sum_j \sum_{(\mathrm{t}, \mathrm{R'}) \in \mathcal{F}(x_j)} f(\mathrm{t}, \mathrm{R'})k_{\mathrm{R}}(\mathrm{R}_g\mathrm{t} + \mathrm{t}_g - \mathrm{x}, \mathrm{R}_g\mathrm{R'}) = \\
&\sum_j \sum_{(\mathrm{t}, \mathrm{R'}) \in \mathcal{F}(x_j)} f(\mathrm{t}, \mathrm{R'})k((\mathrm{x}, \mathrm{R})^{-1}(\mathrm{t}_g,\ \mathrm{R}_g)(\mathrm{t},\ \mathrm{R'})) = \\
&\sum_j \sum_{(\mathrm{t}, \mathrm{R'}) \in \mathcal{F}(x_j)} f(\mathrm{t}, \mathrm{R'})k\big(((\mathrm{t}_g,\ \mathrm{R}_g)^{-1}(\mathrm{x}, \mathrm{R}))^{-1}(\mathrm{t}, \mathrm{R'})\big) = \\
&\big[(\Phi_{\mathcal{F}})(f)\big](g^{-1}(\mathrm{x}, \mathrm{R})) = \big[(\rho^{\mathcal{Y}}(g) \circ \Phi_{\mathcal{F}})(f)\big](\mathrm{x}, \mathrm{R})
\end{aligned} \tag{2}$$

□

B. Relation to concurrent work

Previous works have also achieved (piecewise) equivariance by averaging over a frame [13, 17]. While Xiao *et al.* [17] only explore this idea to obtain global rotation equivariance, Puny *et al.* [13] propose a more general framework based on the same idea. The concurrent work of Atzmon *et al.* [3] uses this idea to achieve piecewise (local) equivariance by applying frame averaging for partitions of individually transforming regions. In these works, the same frame is used for each point in the point cloud (global) or partitions (piecewise/local) for the symmetrization of, e.g., a neural network by transforming their (partitions') domain. For Atzmon *et al.* [3], a partition prediction network is necessary since simply increasing the number of partitions to reduce partition errors limits the expressivity of the resulting equivariant point network; when each point belongs to one partition, the only shared equivariant function is constant. In contrast, this work is based on group convolutions, where point convolutions in a neural network are lifted to the SE(3) group so the kernel can detect rotated patterns. In this context, the concept of equivariant frames is used to define a point-specific grid on SE(3) to solve the convolution integral over the SE(3) group efficiently. Since we create point-specific frames, we avoid the need to group points into regions that can rotate jointly. The locality of the features created by the proposed convolution operator is determined by the point neighborhoods used for feature aggregation. Stacking several layers of such convolution operators results in a network capable of detecting local equivariant features up to features equivariant to the accumulated receptive field among the layers. Using efficient SE(3) equivariant convolutions to construct a network, while it can potentially overfit to global features, provides a more general framework than applying symmetrization of the network for constructed regions.

C. Pose estimation

Additionally, we evaluate our convolutions in the pose estimation task. In this task, the model aims to recover the relative rotation between two point clouds of the same shape.

Dataset. We follow Zhu et al. [20] and use the airplane category from the ModelNet40 dataset for this task, composed of 626 models in the training set and 100 in the test set. As Zhu et al. [20], we also sample the surface of the shapes with 1024 points to generate the point clouds, which are randomly rotated to form a pair.

Experimental setup. Existing methods such as EPN [5] and E2PN [20] rely on the discretization of the SO(3) group to achieve equivariance. Therefore, their pose estimation network must predict an assignment between the two discrete sets of rotations plus a displacement to cover the full SO(3) space. Our convolutions, on the other hand, do not rely on a discrete sampling of the SO(3) space and work directly in continuous space. Therefore, our models only need to predict an assignment between the 4 reference frames, which can be framed as a contrastive learning problem.

Table 1. Error in degrees of different methods on the task of pose estimation on ModelNet40.

Metrics	# samp.	Mean(°)	Median(°)	Max(°)
EPN [5]		1.10	1.36	7.06
E2PN [20]		1.20	0.96	6.71
	4	4.4×10^{-5}	6.7×10^{-5}	2×10^{-3}
Ours	2	4.9×10^{-5}	6.7×10^{-5}	2×10^{-3}
	1	4.9×10^{-5}	6.7×10^{-5}	2×10^{-3}

Main results. Table 1 presents the results of this experiment. We can see that our model, thanks to operating in continuous space, can achieve an angular error orders of magnitude smaller than the existing methods for any number of samples used in the layers of the network.

D. Dataset details

Shape classification. The ModelNet40 dataset [16] is composed of synthetic CAD models from 40 different classes. The dataset is divided into two splits, where $9,843$ objects are used for training and $2,468$ for testing. Since each model is composed of multiple faces, we sample $4,096$ points using farthest point sampling on the surface.

Semantic segmentation: human body parts. For training, we use the train split of DFAUST [4] used in [2, 6] and follow Feng *et al*. [9] to create $15,430$ point clouds by sampling $4,096$ points across the mesh surface. The PosePrior dataset [1] consists of challenging poses significantly divergent from those executed in DFAUST, which we use to test our model for generalization to unseen, out-of-distribution poses. Following the procedure of the train set, we derive $3,760$ point clouds with $4,096$ points each from this dataset for testing.

Semantic segmentation: scene understanding. We follow the standard train and validation split of ScanNet [7] and use color $[r, g, b]$ as input point features in addition to the 3D coordinates.

E. Implementation details

In this section, implementation details are given, and the architecture of the network used is introduced. Classification, pose estimation, and segmentation tasks share the same encoder structure, yet a decoder is used to provide point-wise predictions for the latter.

Frame computation. To compute the local PCA for each point in the point cloud, we select 16 neighbors using *k* nearest neighbors (*k*NN). We compute the covariance matrix from the points and define the frame from the axes of PCA; Tab. 5 shows an ablation of k.

Rotation representation. To represent the relative rotations between neighboring points that we give as input to the learnable kernel, we use the 6D representation proposed by Zhou *et al*. [19]. However, other viable representations, such as quaternions or rotation matrices, could be used.

E.1. Network architecture

For our experiments, our model uses ResNetFormer blocks [18] as the main computational blocks in the encoder and an FPN decoder [10] for tasks requiring per-point predictions. The different point cloud resolutions are computed using Cell Averaging [15]. In our convolution operation, we define our kernel as a single layer Multi-Layer Perceptron (MLP) with 32 hidden neurons and GELU activation functions. The output of our network is several feature vectors for each point that correspond to the sampled rotations. We use mean pooling as the projection layer Eq. (4) to get the final output feature per point, but any other pooling can be used.

E.1.1 Encoder

The input point cloud is transformed into n down-scaled versions of itself using the Cell Average (CA) method [15]. For the first down-scaling, the size of the voxel cells used in the CA algorithm is a hyper-parameter, d, which is then sequentially doubled for each of the following down-scaling steps. The initial features are obtained with a patch encoder similar to the one used in vision transformers [8]. The patch encoder allows us to extract features from a smaller cell size, which usually increases the model's performance as more points are available while keeping computational costs within limits. We use one additional level with two convolutions for the patch encoder for the classification and segmentation task on DFAUST; for ScanNet20, we skip the patch encoder. The (extracted) initial features are further processed with a set of Metaformer blocks [18] before being transferred down to the next down-scaled point cloud via a convolution operation. This procedure is iterated until we reach the final down-scaled point cloud. For pose estimation and classification, we use $n = 5$ and mean-aggregation of features in the case of classification. The aggregated feature vector is then passed through a linear layer to perform the final prediction. For the segmentation task on DFAUST and ScanNet20, the features of each level of the encoder serve as input to the decoder, with $n = 4$ and $n = 5$, respectively.

Metaformer blocks. We incorporate the block design defined by Yu *et al*. [18] into our architecture, replacing the attention module of transformers with our point convolution. Each block consists of two residual blocks. In the first one, feature updates are computed using point convolution, while in the second one, updates are determined through a point-wise MLP with two layers. In this MLP, the initial layer doubles the feature count, while the second layer reduces it to the desired output number.

E.1.2 Decoder

Our Decoder architecture is based on the feature pyramid network proposed by Kirillov *et al*. [10]. The input to the decoder is the feature map of the down-scaled point cloud for which a stepwise up-sampling with our point convolutions is employed, progressing from the lowest level to the first down-scaled point cloud. To enhance information and gradient flow, we incorporate skip connections, where features from both the encoder and decoder are summed, producing a distinct feature map for each down-scaled point cloud. Subsequently, each feature map is up-sampled to the initial down-scaled point cloud through a singular up-sampling operation. The resulting n feature maps are then aggregated through summation. If applicable, this feature map is put through a patch decoder, inverting the patch encoder operation. Finally, it is up-sampled to the intended prediction positions by a final convolution and processed by a one-layer MLP to obtain the point-wise predictions.

F. Experimental setup

In all experiments, we use AdamW [12] as optimizer with a weight decay value of 1^{-4} and OneCycleLr [14] as learning rate scheduler. Moreover, we employ drop residual paths depending on the depth of the layer [11] and gradient clipping for gradient norms exceeding 100. We used label smoothing with a parameter of 0.2 to prevent overfitting. All models were trained on a single NVIDIA GeForce RTX 3090. The experiment-specific setup is given below.

Classification. For the point cloud classification experiments on ModelNet40, we use the encoder architecture explained in Appendix E.1.1 with the number of blocks and the number of features for each level equal to [2, 3, 4, 6, 4] and [32, 64, 128, 256, 512], respectively. The initial grid resolution was $d = 0.05$ and a maximum drop rate of 0.2. All models were trained for 500 epochs using a batch size of 12, with a learning rate of 0.01 and an initial and final division factor of 100 and 10000. We used jitter coordinates, mirroring, and random scale augmentation during training.

Pose estimation. For the pose estimation experiment on the airplane category of ModelNet40, we use the same model as for classification. Note that the projection layer is omitted for this experiment since equivariance instead of invariance is needed. The model using all four frame elements was trained for 500 epochs with a batch size of 8. Using only 2 or 1 element takes longer for the model to converge; we trained for 2k and 4k epochs with batch sizes of 16 and 64, respectively.

Segmentation. For the segmentation task on the DFAUST dataset, we again used the encoder architecture introduced in Appendix E.1.1 with two blocks per level and a number of features equal to [32, 64, 128, 256]. We trained all models with a batch size of 32 for 150 epochs. The maximum learning rate was 0.005, with an initial division factor of 10 and 1000 as the final factor. Jitter coordinates are used as augmentation during training; the initial grid resolution was $d = 0.05$, and a maximum drop rate of 0.5. To get the point-wise prediction, the decoder architecture of Appendix E.1.2 was employed.

For the segmentation task on the ScanNet20 dataset, we used the same encoder and decoder architecture as for DFAUSt, described in Appendix E.1.1 and Appendix E.1.2, but used five levels with [2, 3, 4, 6, 4] blocks and [64, 128, 192, 256, 320] feature dimensions. The initial grid resolution was $d = 0.1$, and all models were trained using 250 batches for 600 epochs using standard augmentations such as jitter coordinates, mirroring, random scaling, elastic distortion, and translation.

Projection layer. Our proposed method provides features for each sample of the SO3 group; if four samples are used, four features per point result. Hence, we must aggregate those features to get to the final point-wise prediction or before averaging in the classification task to get the point-wise invariant feature vectors. We use mean-pooling over the features corresponding to the same coordinates.

G. Additional qualitative results

Figure 1 and Fig. 2 provide additional qualitative results.

H. Additional ablations

In this section, we provide ablation experiments to evaluate the robustness of our approach against the number of neighbors used in the PCA computation and against noise and point density. Further, we provide results for allowing different numbers of samples to be used for the approximation of the SO(3) integral during training, called *sample mixing*. Finally, we discuss the effect of stochastic sampling of the point-specific grid on SE(3).

Sample mixing. To achieve the best results with minimal training and inference time, we explored *sample mix-*

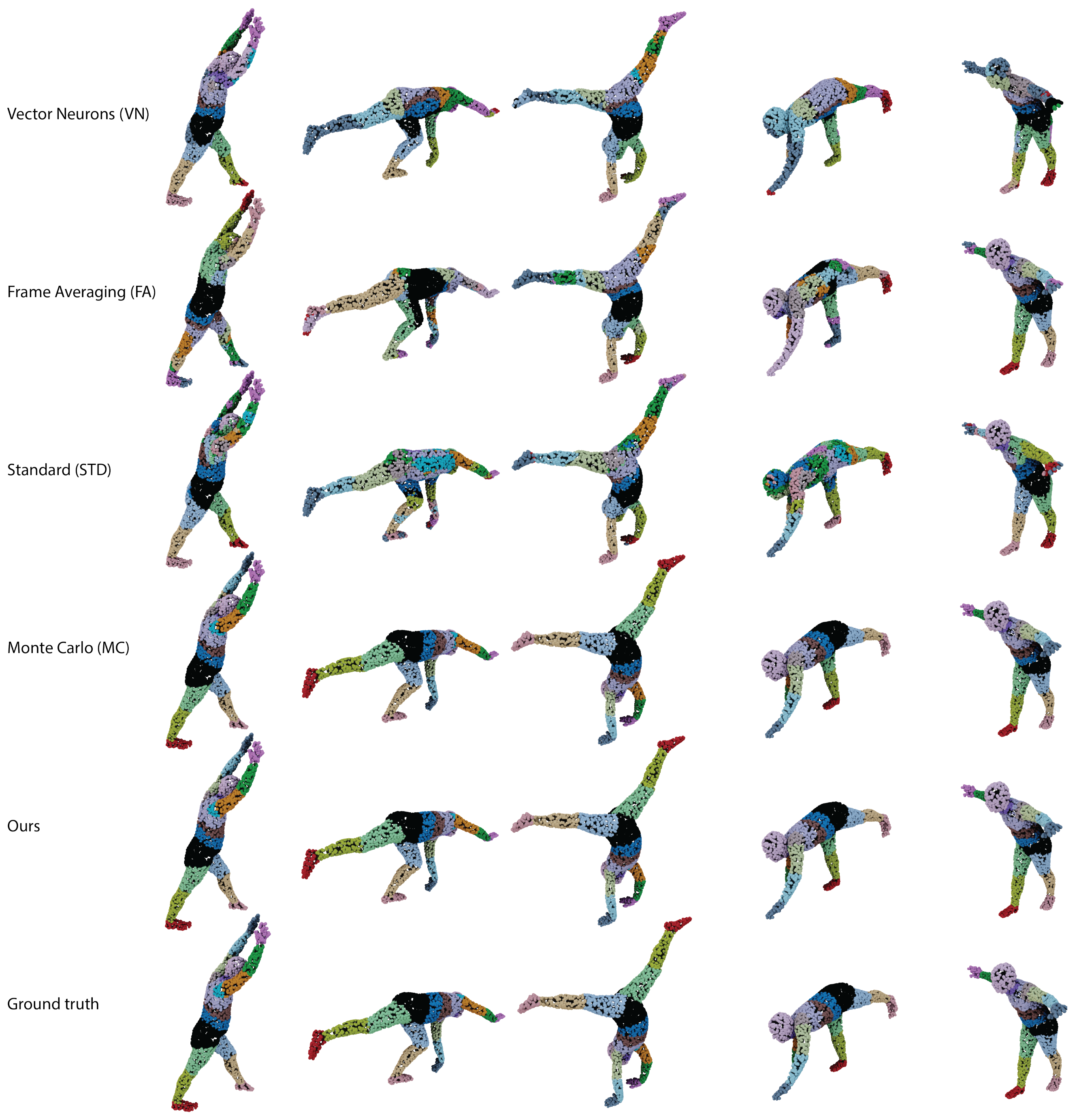

Figure 1. **Additional Qualitative results.** Global equivariant methods such as VN, or FA struggle with out-of-distribution models, especially up-side down models. Our method, on the other hand, achieves almost perfect predictions. Lastly, MC also achieves good performance but falls behind our method, as seen in the leftmost columns when looking at the left upper arm prediction.

ing. Instead of always using the same amount of samples of SO3 elements, the number of samples used changes per step/forward pass and is chosen with the following probability. 1 sample is used with a probability of 50%, 2 samples with a 35% and 4 with a 15% probability. Table 2 and Tab. 3 show the results for the classification and segmentation tasks, respectively. We can see that training with sample mixing and testing with one sample equals or exceeds the performance of training with one sample only for MC and Ours. Further, the results are more stable concerning

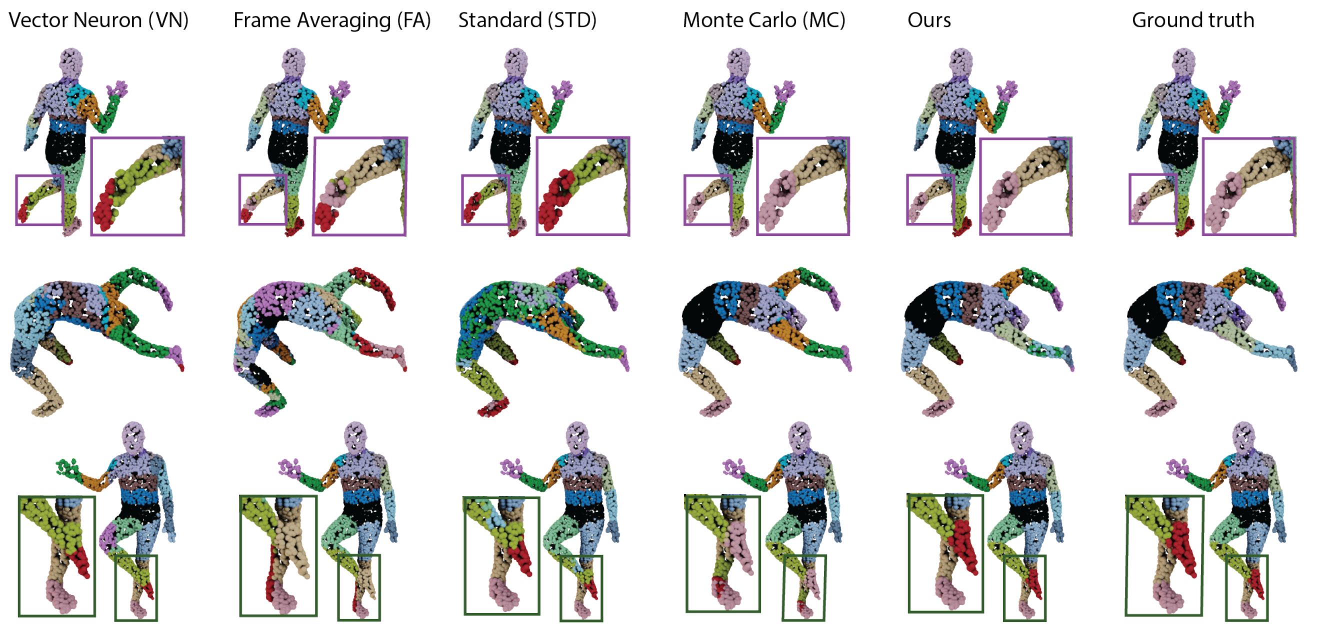

Figure 2. **Additional Qualitative results.** Global equivariant methods such as VN, or FA struggle with out-of-distribution models. Our method, on the other hand, achieves almost perfect predictions. Lastly, as seen in all three examples, MC also achieves good performance but falls behind our method.

different numbers of samples used during testing than when training with a fixed number of samples. Sample mixing is thus a viable option when limited resources are available. Moreover, training with larger batch sizes becomes feasible by further allowing different numbers of samples not only between but also within batches. Table 4 shows the number of minutes each epoch approximately takes during training with the mixing strategy, 1, 2, and 4 samples. Sample mixing with the proposed probabilities takes, on average, as long as training with two samples while delivering a more robust approximation of the integral.

PCA computation. We also analyze the effect of the receptive field used to compute the PCA for each point on the final performance of the network using one sample to estimate the integral over SO(3). We can see in Tab. 5 that using a low number of neighboring points makes the resulting frames noisy and hampers the model's performance, becoming similar to the results of MC using a random grid. However, with 16 neighbors, the PCA computation becomes robust, and we do not see significant improvement when we increase this receptive field.

Robustness to noise and density. We experimented with the robustness of our model w.r.t. noise and density variations in the input point cloud. Results on the DFAUST dataset using two samples are reported in Tab. 6 and Tab. 7, respectively. We can see that our model is robust against increased levels of noise and reduced point density during testing. However, if the noise increases significantly (0.015 std. dev.) or the number of points is reduced substantially (1024 points), the PCA computation is affected, and the model's performance decreases. This can be easily solved by training the model with high noise levels or with a reduced number of points as shown in Tab. 6 and Tab. 7, where the model achieves similar performance to the model trained without point corruptions.

Effects of stochastic sampling One of the main contributions of our work is to sample one or two LRF stochastically during training. How the learning is affected by this sampling boils down to how noisy the gradient estimation for the kernel parameters is. If this gradient were computed for a single point, the gradient would be noisy. However, this noise is significantly reduced since the gradient is computed as the expectation over multiple samples, multiple points, and multiple point clouds in the batch. In our experiments, models trained with 1 LRF or 2 LRF during training perform only marginally worse than those trained with 4. During testing, sampling 1 or 2 LRF can result in noisy predictions. However, these predictions remain equivariant since the same sampling of LRF will produce the same results for random SO(3) rotations.

Table 2. Results for mixing the number of used samples throughout training for the classification task on the ModelNet40 dataset.

Method	# samp.	I / I			I / SO(3)			SO(3) / SO(3)		
	train ↓/ test →	1	2	4	1	2	4	1	2	4
MC	mix	84.9	85.7	85.9	78.1	77.9	77.51	86.9	86.8	86.6
Ours	mix	87.2	86.9	87.0	86.3	86.2	86.3	88.4	88.5	88.4
STD			90.7			12.3			87.5	

Table 3. Results for mixing the number of used samples throughout training for the segmentation task on the DFAUST dataset.

Method	# samp.	mAcc			mIoU		
	train ↓/ test →	1	2	4	1	2	4
MC	mix	93.8	93.7	93.7	88.7	88.6	88.5
Ours	mix	94.0	94.1	94.1	89.2	89.3	89.3
STD			85.3			74.5	

Table 4. Time of 1 epoch in minutes during training with different numbers of samples on the DFAUST dataset.

# samp.	mix	1	2	4
time (min)	6.1	3.6	6.1	15.0

Table 5. Effect of the k chosen for the kNN operation in the PCA computation on the model's performance with one frame element on the ModelNet40 dataset.

4	8	16	32	64
80.6	83.1	85.5	85.4	85.9

Table 6. Robustness w.r.t. noise variations.

Noise			
train	test	mAcc	mIoU
0.005	0.005	94.8	90.6
	0.010	93.9	90.0
	0.015	38.6	25.1
0.015	0.015	93.2	88.2

I. Limitations

Although the proposed convolution operation is local equivariant via the restricted receptive field, when multiple layers are combined in a deep network, the whole model does not become local equivariant and remains global equivariant. However, from the experiments presented in **??**, where the network aims to perform local predictions, our model shows robustness to such scenarios, indicating that the model relies on local features to perform the predictions.

Table 7. Robustness w.r.t. density variations.

Point Density			
train	test	mAcc	mIoU
4096	4096	94.5	89.7
	2048	93.3	87.7
	1024	30.4	20.1
1024	1024	92.7	86.8

References

[1] Ijaz Akhter and Michael J Black. Pose-conditioned joint angle limits for 3d human pose reconstruction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1446–1455, 2015. 2

[2] Matan Atzmon, Koki Nagano, Sanja Fidler, Sameh Khamis, and Yaron Lipman. Frame averaging for equivariant shape space learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 631–641, 2022. 2

[3] Matan Atzmon, Jiahui Huang, Francis Williams, and Or Litany. Approximately piecewise e (3) equivariant point networks. In *The Twelfth International Conference on Learning Representations*, 2024. 1

[4] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J Black. Dynamic faust: Registering human bodies in motion. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6233–6242, 2017. 2

[5] Haiwei Chen, Shichen Liu, Weikai Chen, Hao Li, and Ran-

dall Hill. Equivariant point network for 3d point cloud analysis. pages 14514–14523, 2021. 1, 2

[6] Xu Chen, Yufeng Zheng, Michael J Black, Otmar Hilliges, and Andreas Geiger. Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11594–11604, 2021. 2

[7] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *CVPR*, 2017. 2

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arxiv 2020. *arXiv preprint arXiv:2010.11929*, 2010. 2

[9] Haiwen Feng, Peter Kulits, Shichen Liu, Michael J Black, and Victoria Fernandez Abrevaya. Generalizing neural human fitting to unseen poses with articulated se (3) equivariance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7977–7988, 2023. 2

[10] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 3

[11] Gustav Larsson, Michael Maire, Gregory Shakhnarovich, et al. Ultra-deep neural networks without residuals. In *Int. Conf. on Learning Representations, arXiv, Toulon, France*, page 1605, 2017. 3

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 3

[13] Omri Puny, Matan Atzmon, Edward J. Smith, Ishan Misra, Aditya Grover, Heli Ben-Hamu, and Yaron Lipman. Frame averaging for invariant and equivariant network design. In *International Conference on Learning Representations*, 2022. 1

[14] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, pages 369–386. SPIE, 2019. 3

[15] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *ICCV*, 2019. 2

[16] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2

[17] Z. Xiao, H. Lin, R. Li, L. Geng, H. Chao, and S. Ding. Endowing deep 3d models with rotation invariance based on principal component analysis. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020. 1

[18] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3

[19] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 2

[20] Minghan Zhu, Maani Ghaffari, William A Clark, and Huei Peng. E2pn: Efficient se (3)-equivariant point network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1223–1232, 2023. 1, 2