Supplemental Material for Info3D

Aditya Sanghi

Autodesk AI Lab, Toronto, Canada
aditya.sanghi@autodesk.com

1 Training curve comparison

In this experiment, we use the pre-trained weights on ShapeNet dataset and initialize the DGCNN classifier [6] with those weights. We compare the results with randomly initialized weights by showing the test accuracy training curve in Figure 1.

Figure 1 shows how having pre-trained weights using our method helps in fast training of the classifier. We achieve 91% accuracy within 3 epochs whereas it takes about 32 epochs for random initialize weights.

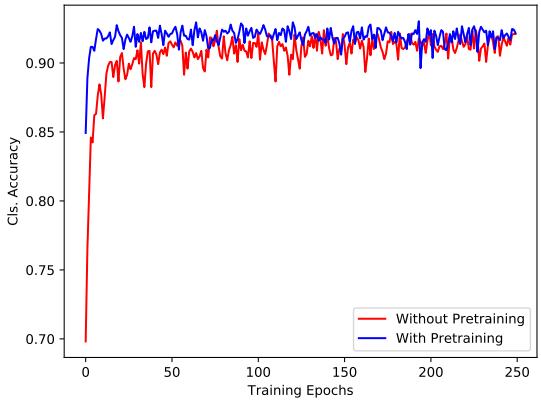


Fig. 1: **Test accuracy curve using pre-trained weights and random weights.**

2 Transfer learning results on unaligned datasets

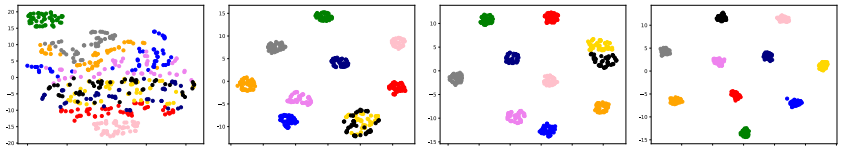
This experiment is similar to the one mentioned in the paper but instead we use unaligned datasets. The results are shown in Table 1. It can be seen that our model again outperforms autoencoder baseline models. This reinforces the idea that our method learns embeddings that are less rotation sensitive.

Method	ModelNet10 (Acc.)	ModelNet40 (Acc.)
Latent-GAN [1]	66.7	50.4
AtlasNet [2]	61.3	45.9
FoldingNet [8]	68.0	48.8
Ours	**84.2**	**73.6**

Table 1: **Transfer Learning Results on Rotated Datasets.**

3 Training progress for unaligned shapes

We conduct the rotation invariant experiment as mentioned in the paper on ModelNet40. Figure 2, provides a illustration to show how clustering evolves during training. We take different checkpoints of the model and do a t-SNE pot with the same experimental setup mentioned above. It can be seen from the figure that as training progresses objects get clustered as expected.



(a) Checkpoint 5 (b) Checkpoint 15 (c) Checkpoint 20 (d) Checkpoint 750

Fig. 2: **t-SNE visualization of training process.** Figure illustrating how 3D shapes and their random poses get clustered as training proceeds

4 ModelNet10 Transfer learning accuracy on aligned dataset

In this experiment we compare with other methods on ModelNet10 dataset. The experiment result are shown in Table 2. As it can be seen, our method performs better than all methods except Latent-GAN [1]. As pointed out by works such as [8], it is not clear how the pointcloud was sampled from [1], and hence making a comparison with that work is inconclusive.

5 Architectures of model and baseline

We use the PointNet [4] architecture and the DGCNN architecture [6] as our encoder. The PointNet architecture is as follows: ConvBlock $(3, 64) \rightarrow$ ConvBlock(64, 64) $\rightarrow$ ConvBlock (64, 64) $\rightarrow$ ConvBlock (64, 128) $\rightarrow$ ConvBlock(128, 1024)

Method	ModelNet10 (Acc.)
Latent-GAN [1]	**95.30**
3D-GAN [7]	91.00
FoldingNet [8]	94.40
Recon. PC (PointNet) [5]	91.61
Recon. PC (DGCNN) [5]	94.52
ours (PointNet)	*93.08*
ours (DGCNN)	*94.64*

Table 2: **Transfer Learning Results on Aligned ModelNet10 Datasets.**

→ MaxPool → FCBlock(1024, 1024) → Linear(1024, 1024). Where ConvBlock stands for 1D convolution followed by BatchNorm [3] and Relu activation. FC block stands for linear layer followed by BatchNorm and Relu activation.

The DGCNN encoder is as follows: Conv2DBlock(6, 64) → Conv2DBlock(128, 64) → Conv2DBlock(128, 128) → Conv2DBlock(256, 256) → ConvBlock(512, 1024) → MaxPool → FCBlock(1024, 1024) → Linear(1024, 1024). As in [6], we use the neighborhood information. Here Conv2DBlock stands for 2D convolution followed by BatchNorm2D and Relu activation.

As mentioned earlier we use the same PointNet encoder as our model for all the baselines. In the case of the decoder, we try to replicated the architecture from the paper or from the github repository if provided. We reconstruct the shapes for all the baselines using Chamfer Loss. The latent vector size used for all experiments is 1024.

In the case of Latent-GAN, we use the model as mentioned in their SVM experiments. As we are not creating generative models, we do not train a GAN on top of the autoencoder and just use the autoencoder for representation learning. The decoder consists of three layers: FCBlock (1024, 1024) → FCBlock(1024, 2048) → FCBlock (2048, 2048 * 3). FCBlock stands for linear layer followed by BatchNorm and Relu activation.

For experiments involving AtlasNet, we use the 25 patches model. We take the decoder code from https://github.com/ThibaultGROUEIX/AtlasNet and integrate it with our training settings. For each patch a MLP block is used. Each MLP block consists of ConvBlock($1024 + 2, 1026$) → ConvBlock($1026, 513$) → ConvBlock($513, 256$) → ConvBlock($256, 3$). Where ConvBlock stands for 1D convolution followed by BatchNorm and Relu activation.

Finally, for FoldingNet experiments we implement the model as mentioned in the paper. The code from http://www.merl.com/research/license#FoldingNet, inspired our implementation. We used the settings of regular 2D grid as mentioned in the paper. The decoder consists of 2 folding layers. The first folding operation consist of ConvBlock (1024 + 2, 512) → ConvBlock(512, 512) → ConvBlock (512, 3). The second folding operation consists of ConvBlock (1024 + 3, 512) → ConvBlock(512, 512) → ConvBlock (512, 3). Where ConvBlock stands for 1D convolution followed by BatchNorm and Relu activation.

6 Shape retrieval

We show more results for shape retrieval on ABC dataset and a model trained on ModelNet40 dataset. The ABC retrieval results are shown in Figure 3. The results show many duplicate retrievals for some objects. This indicates a lot of objects in ABC dataset are similar. We also compare our method with other autoencoder baselines on the ModelNet40 dataset. The baselines we use are as above. The results are shown in Figure 4 and Figure 5. We also show some failure cases for our model in Figure 6.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. arXiv preprint arXiv:1707.02392 (2017)
2. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: Atlasnet: A papier-m\^ ach\'e approach to learning 3d surface generation. arXiv preprint arXiv:1802.05384 (2018)
3. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
4. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017)
5. Sauder, J., Sievers, B.: Context prediction for unsupervised deep learning on point clouds. arXiv preprint arXiv:1901.08396 (2019)
6. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) **38**(5), 146 (2019)
7. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Advances in neural information processing systems. pp. 82–90 (2016)
8. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 206–215 (2018)

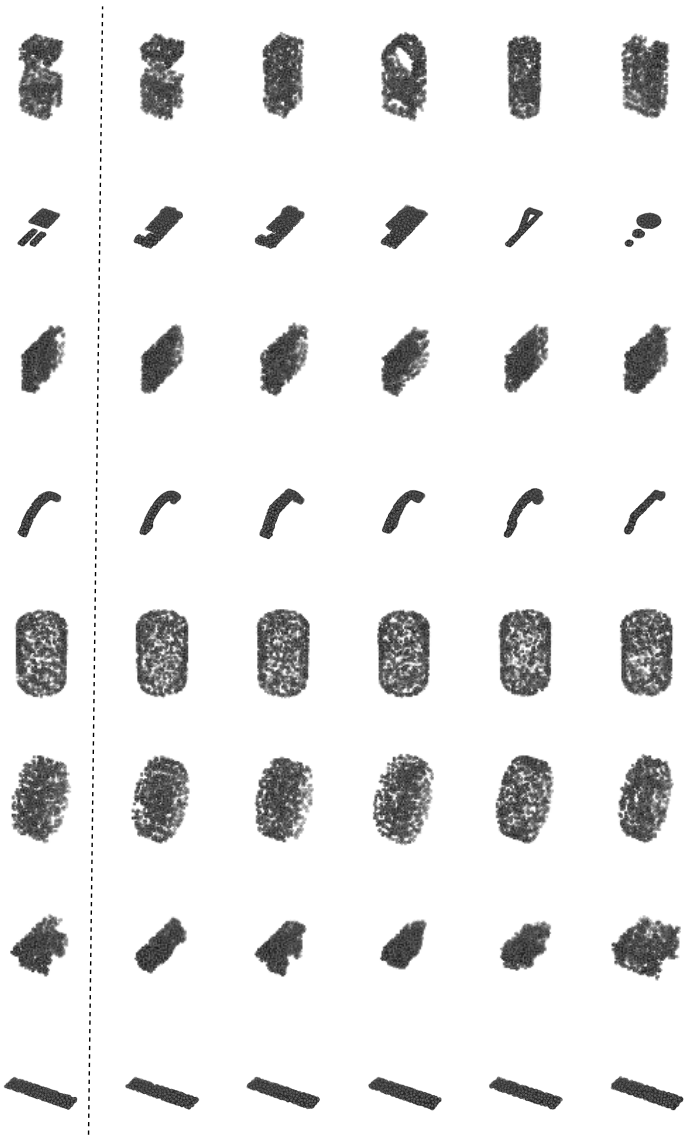


Fig. 3: **Shape Retrieval.** First object in the row represents the query object and next five object are the retrieved objects from the ABC dataset.

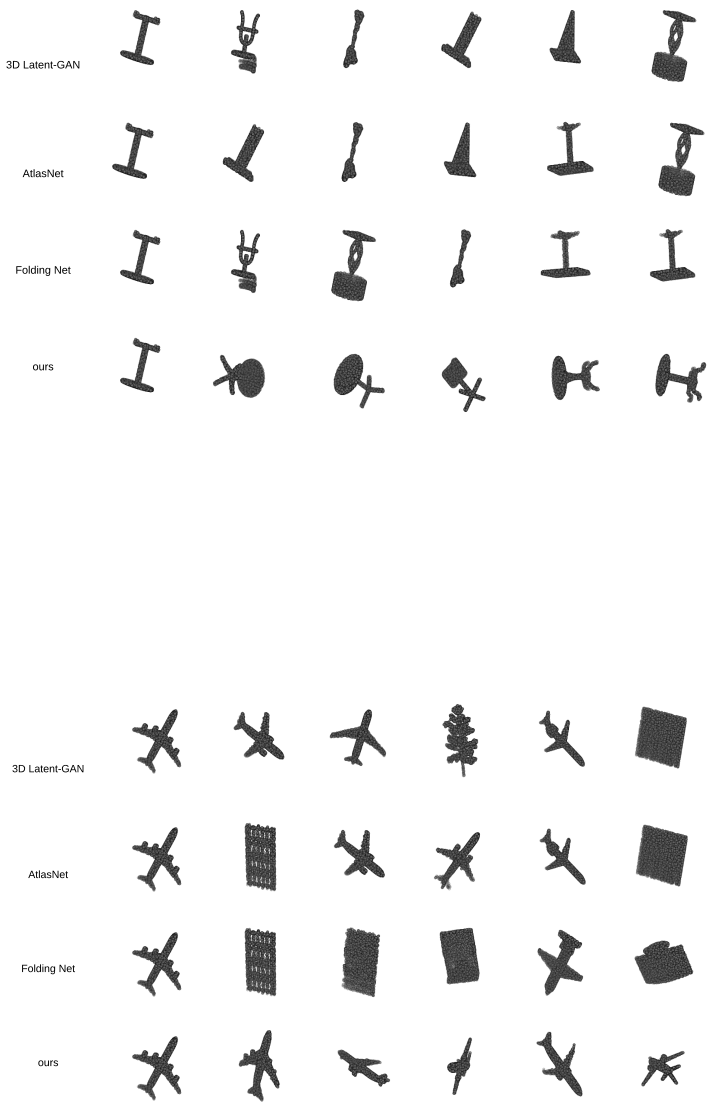


Fig. 4: **Shape Retrieval.** First object in the row represents the query object and next five object are the retrieved objects on ModelNet40 dataset.

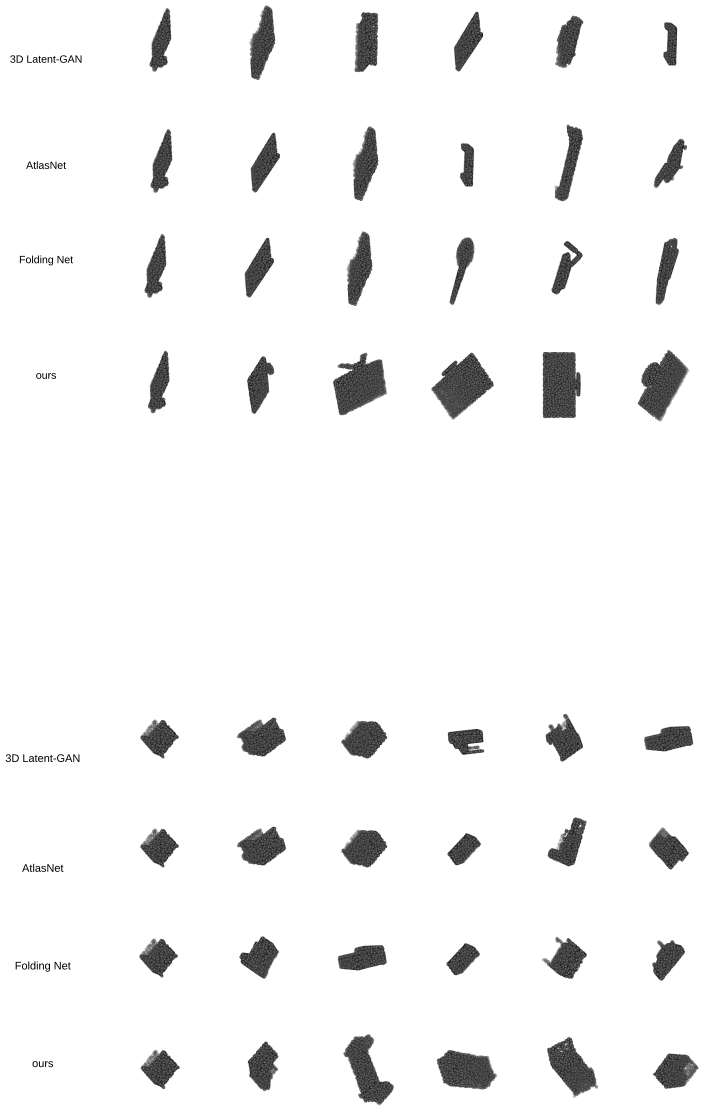


Fig. 5: **Shape Retrieval.** First object in the row represents the query object and next five object are the retrieved objects on ModelNet40 dataset.

Fig. 6: **Failure Retrieval.** All the three rows use our model where first object in the row represents the query object and next five object are the retrieved objects on ModelNet40 dataset.