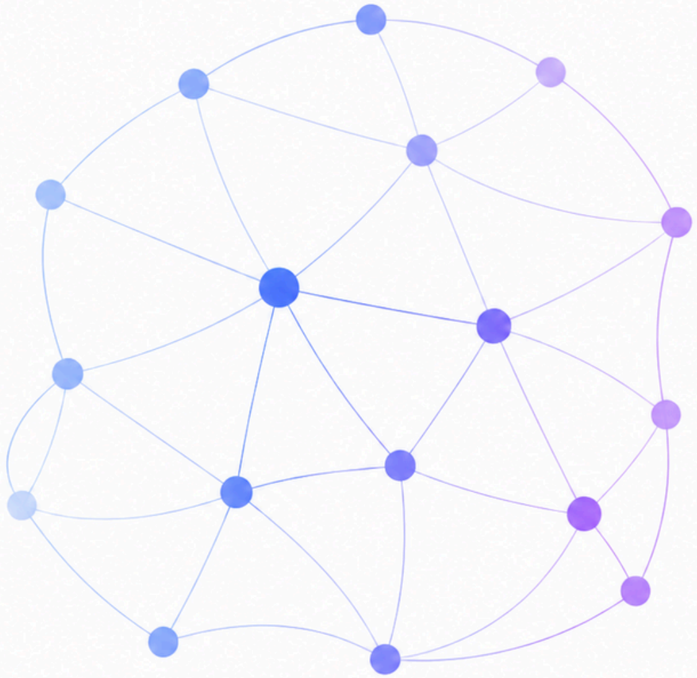
GNNs FOR 3D MODELS GENERATION AND CLASSIFICATION

Applied Machine Learning 2026

Lydia Sakellaridi
Martina Rossi
Tommaso Guarniera



PyG

Can a GNN learn to classify and generate 3D objects?

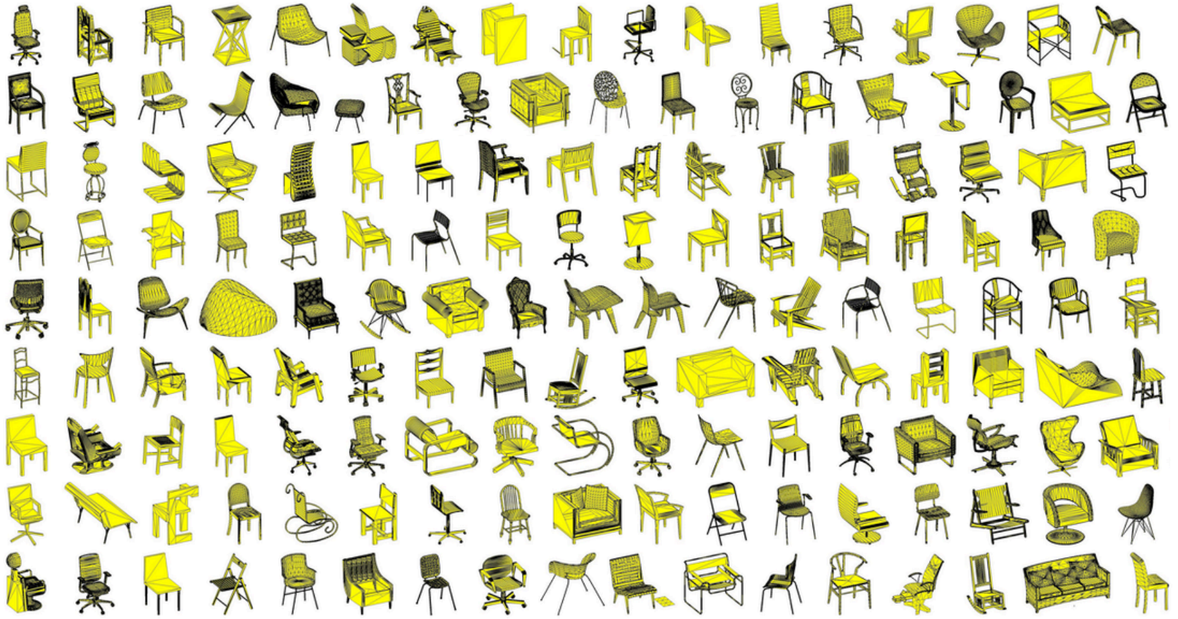
In this project, we present:

1. **Dataset**: ModelNet40 3D objects represented as point clouds
2. **Methodology**: GNN-based classification and generative modeling pipeline
3. **Classification results**: comparison of baseline, AE-pretrained, and SO(3)-invariant models
4. **Generation results**: reconstruction and generation of 3D point-cloud objects - Conditional Variational Graph Autoencoder
5. **Real-world testing**: object scanning experiments and model performance on scanned data

Why do we need models able to classify and generate 3D objects?

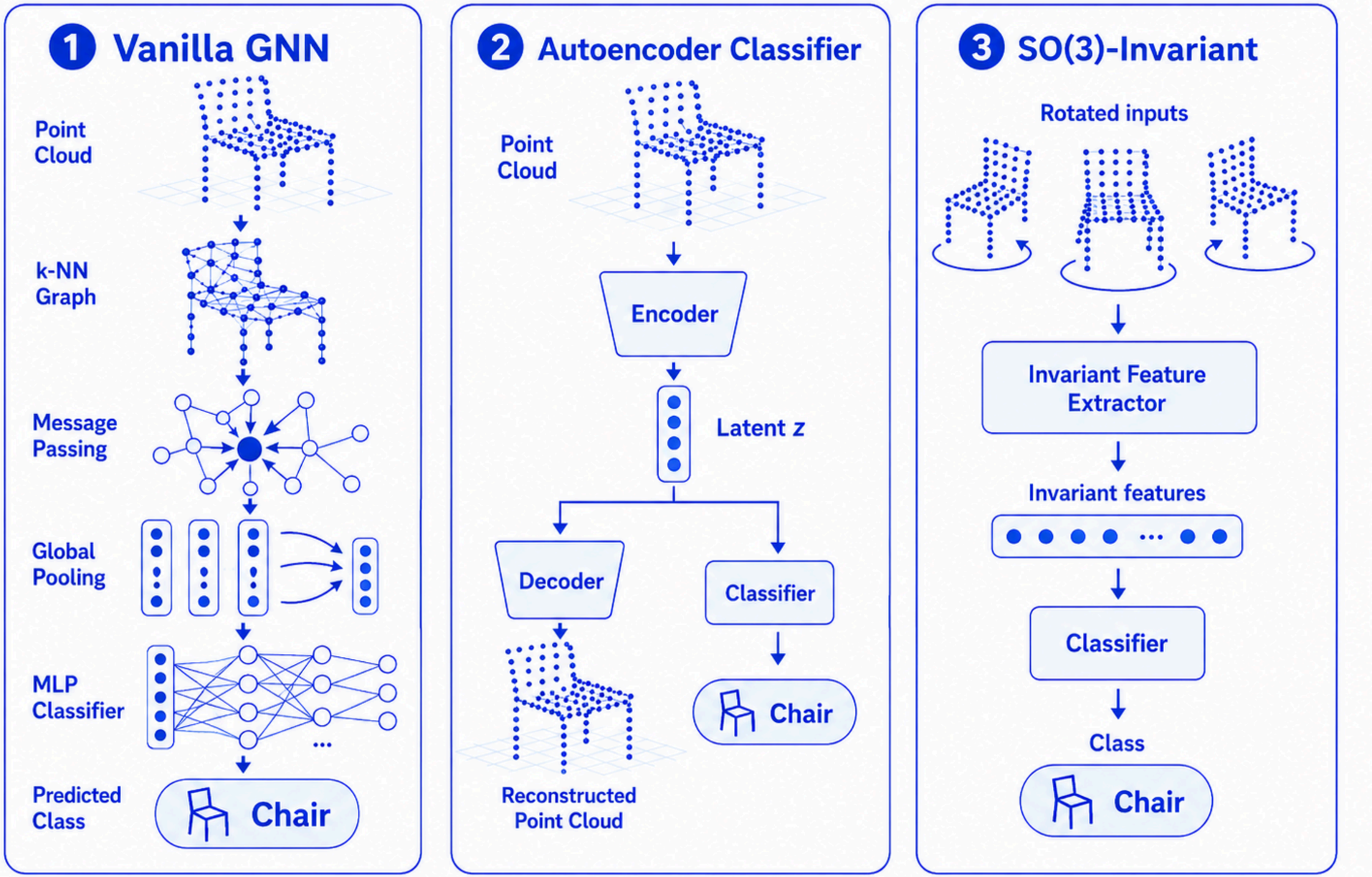
- **Navigation and mobility** - Spatial guidance and obstacle avoidance
- **Robotics** - object recognition and manipulation
- **Healthcare** - Medical imaging analysis
- **Industry & Manufacturin** - Industrial inspection
- **Entertainment & Content Creation** - gaming and virtual word

Dataset: ModelNet40 Point Clouds



- 12,311 graphs
- 2,048 sampled nodes
- KNN edges

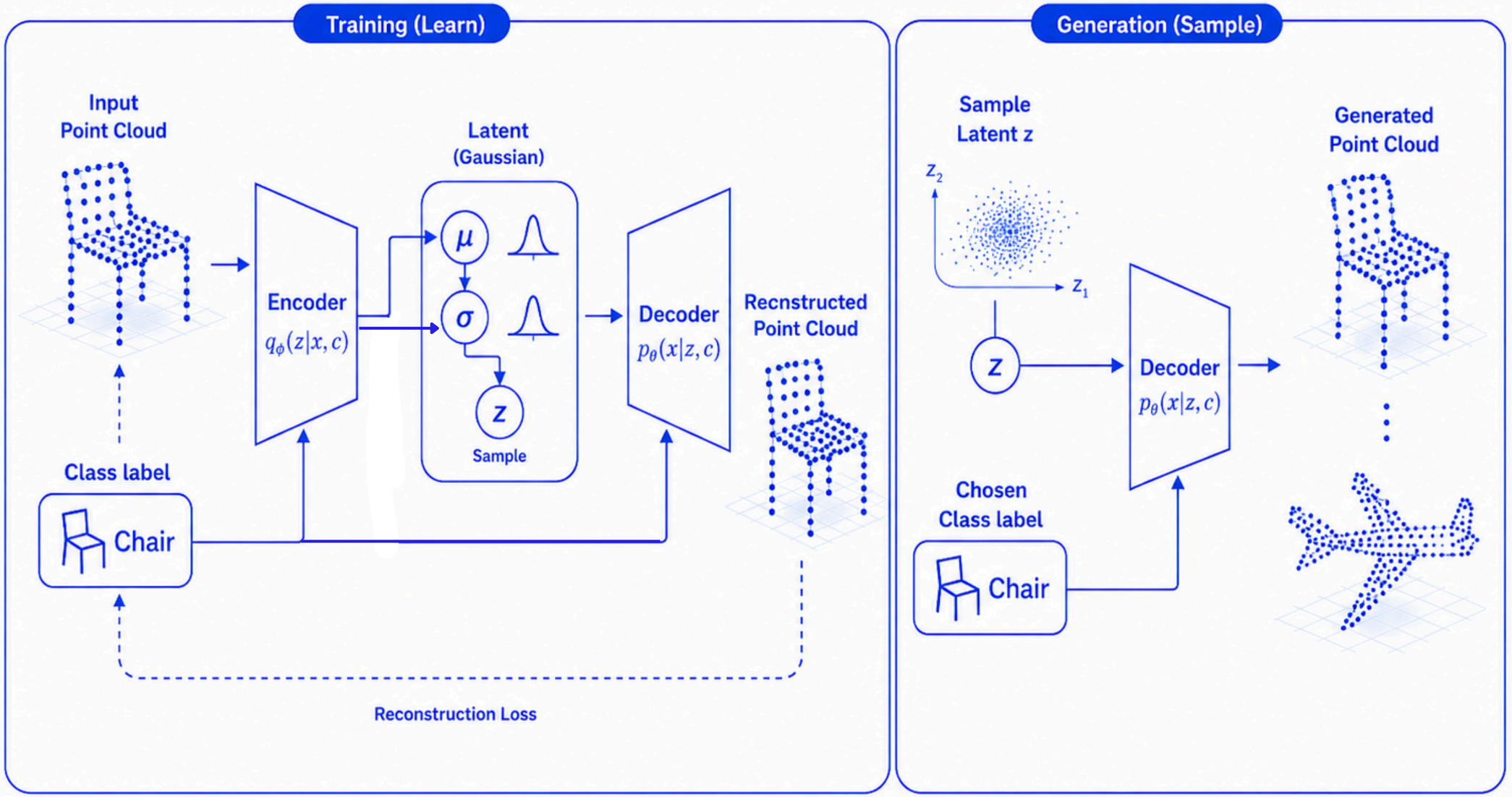
Dataset: ModelNet40 Point Clouds



# params:	851,624	2,034,232	853,288

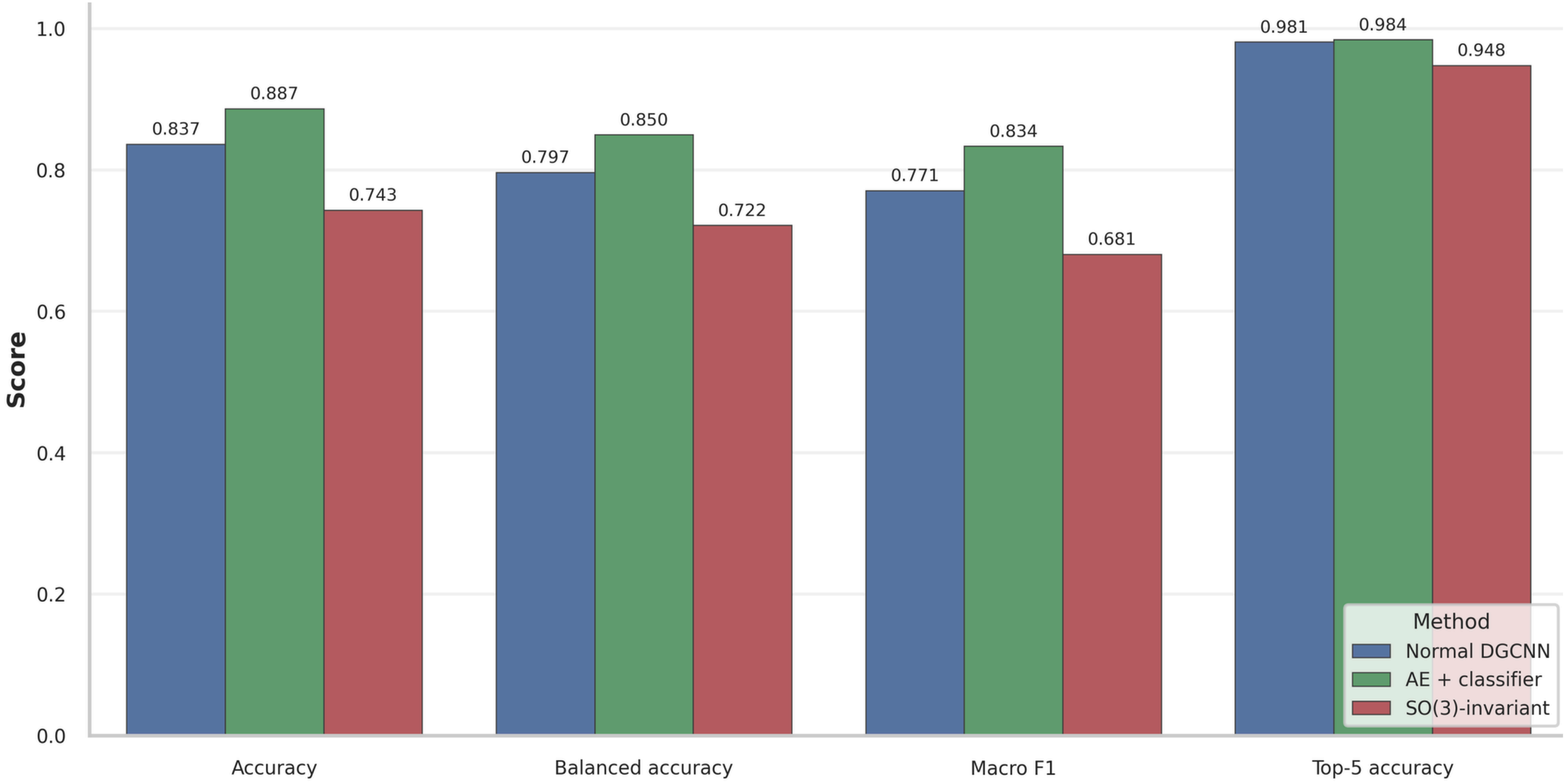
Generation Method

Conditional VAE (CVAE) # params: 2,572,295



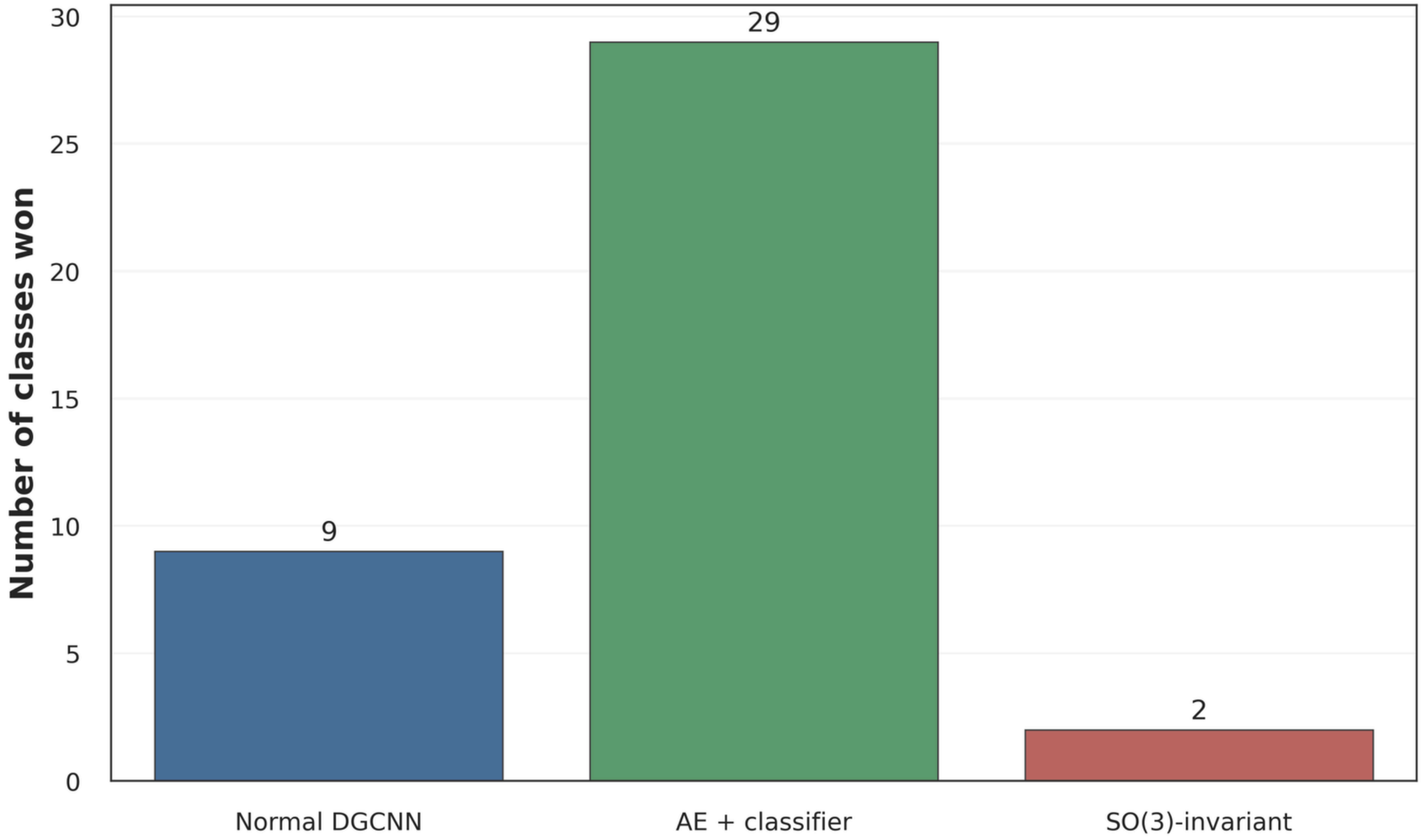
Results: Classification

Classification performance comparison



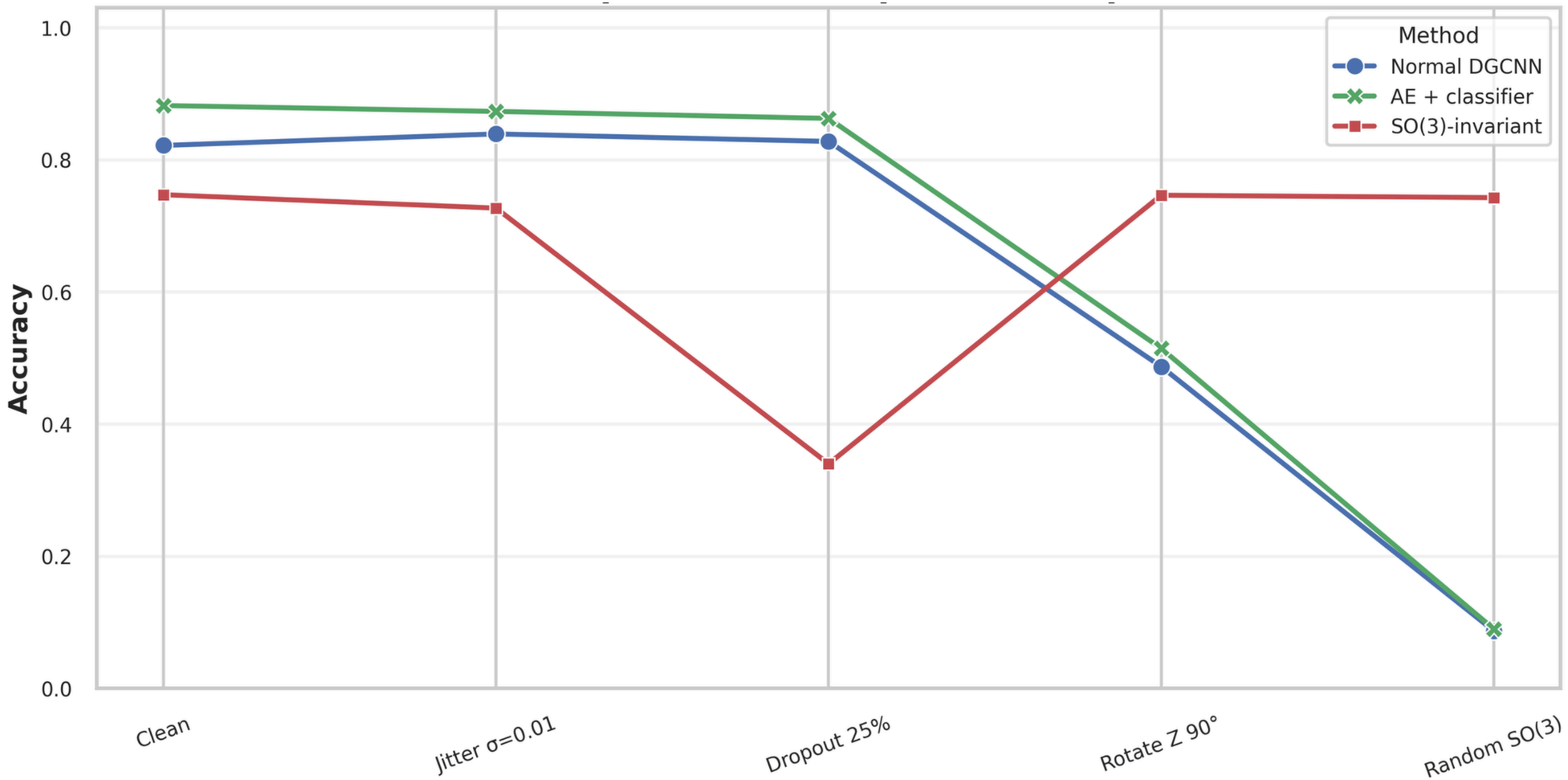
Results: Classification

How many classes each method wins (per-class F1)



Results: Classification

Robustness comparison under point-cloud pertubations

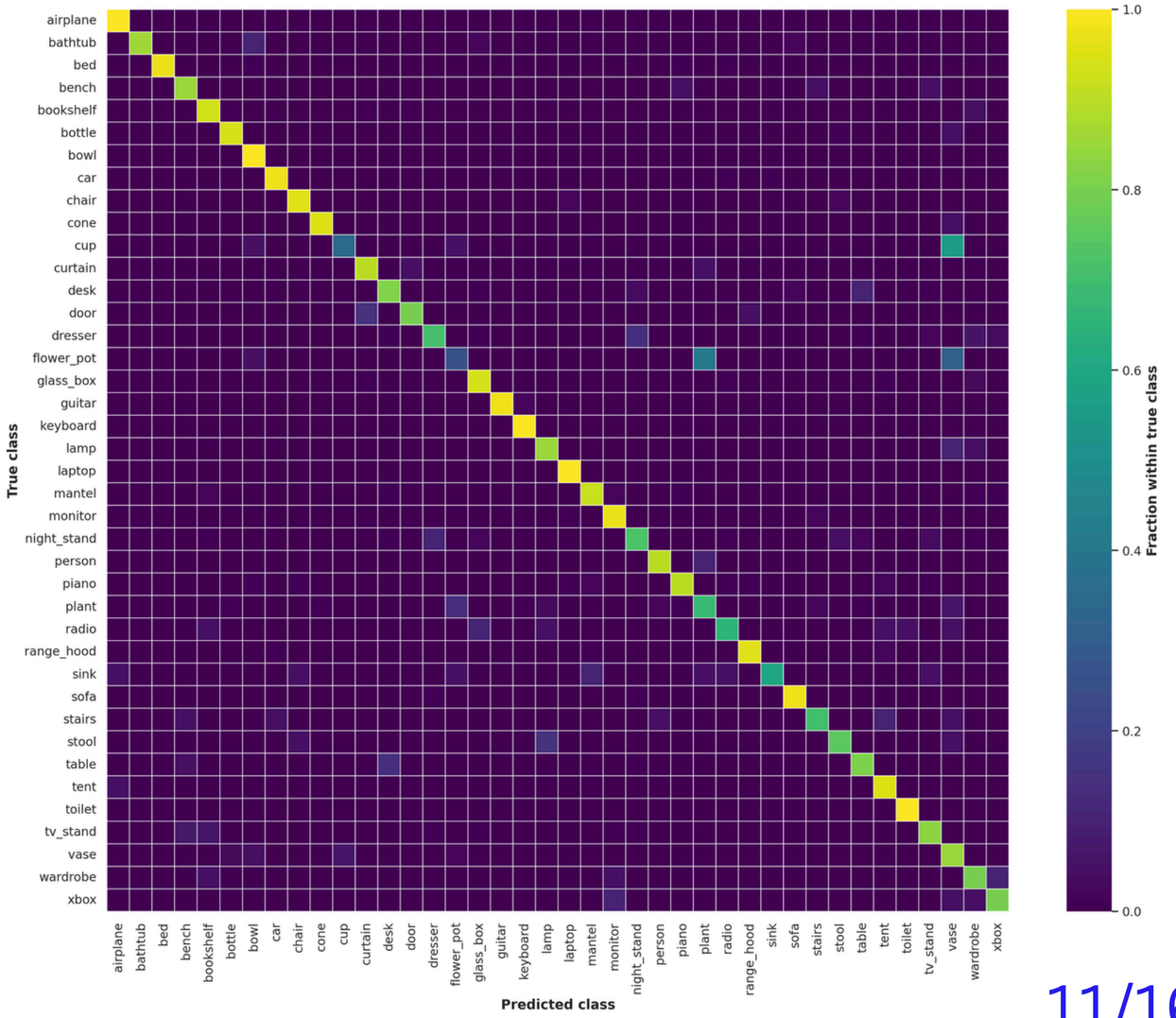


Results: Classification

Normalized confusion matrix - AE classifier

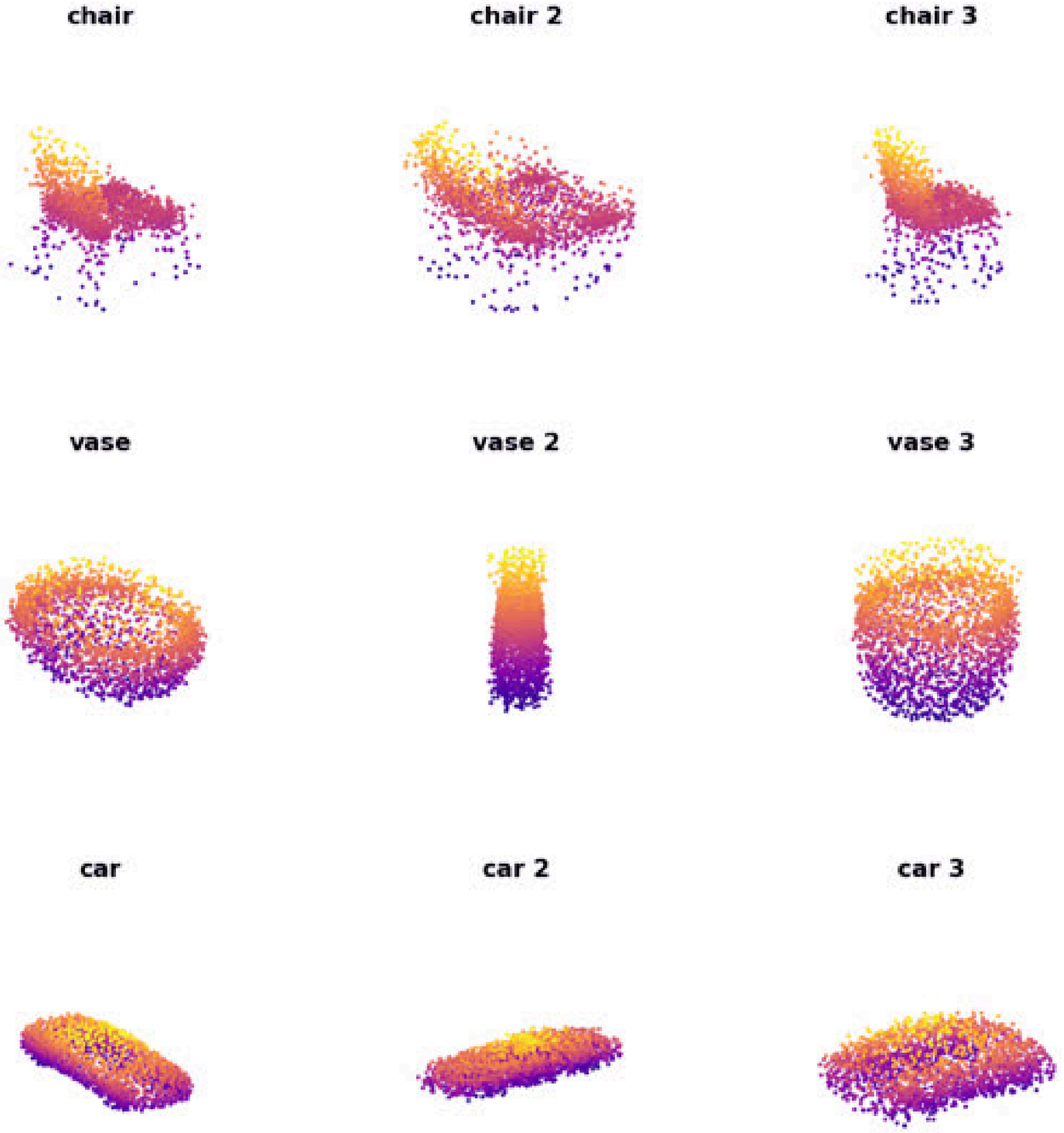
Top 5 misclassification pairs:

1. plant → flower_pot
2. table → desk
3. dresser → night_stand
4. cup → vase
5. night_stand → dresser

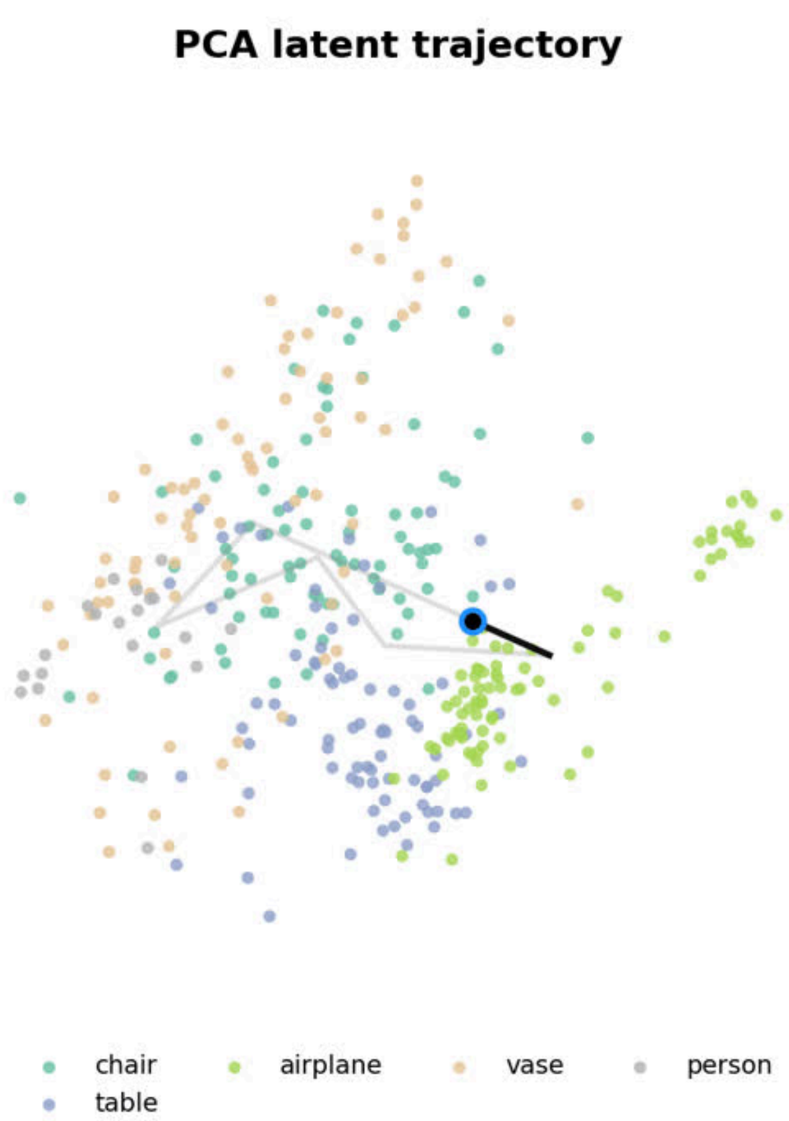


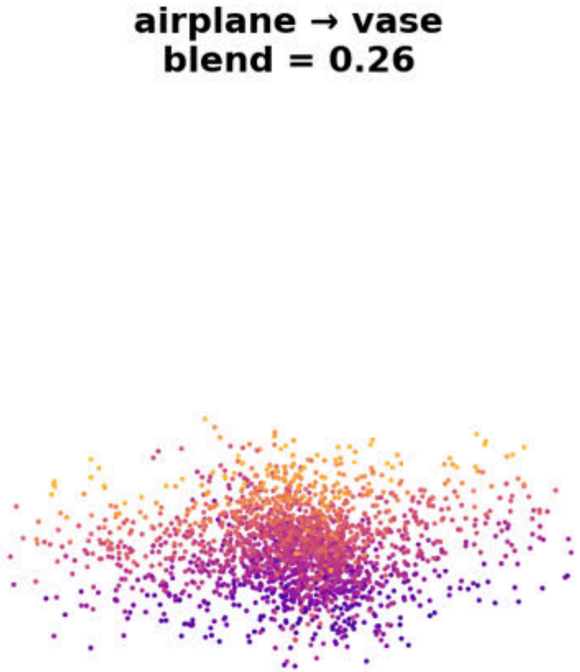
Results: Generation

Gallery of generated samples using CVAE

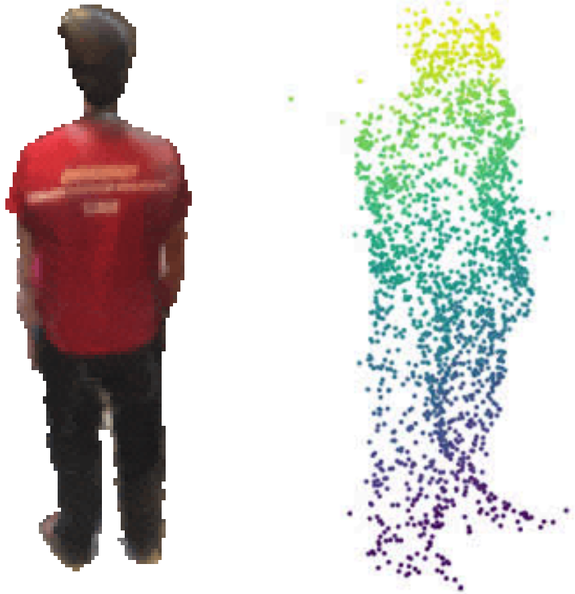


Results: Generation





Results: Real world objects



Vanilla GNN Classifier

Best: person (80.8%)

1. person	80.8%
2. plant	10.5%
3. stool	4.7%
4. door	2.0%
5. curtain	0.7%

Autoencoder Classifier

Best: person (86.2%)

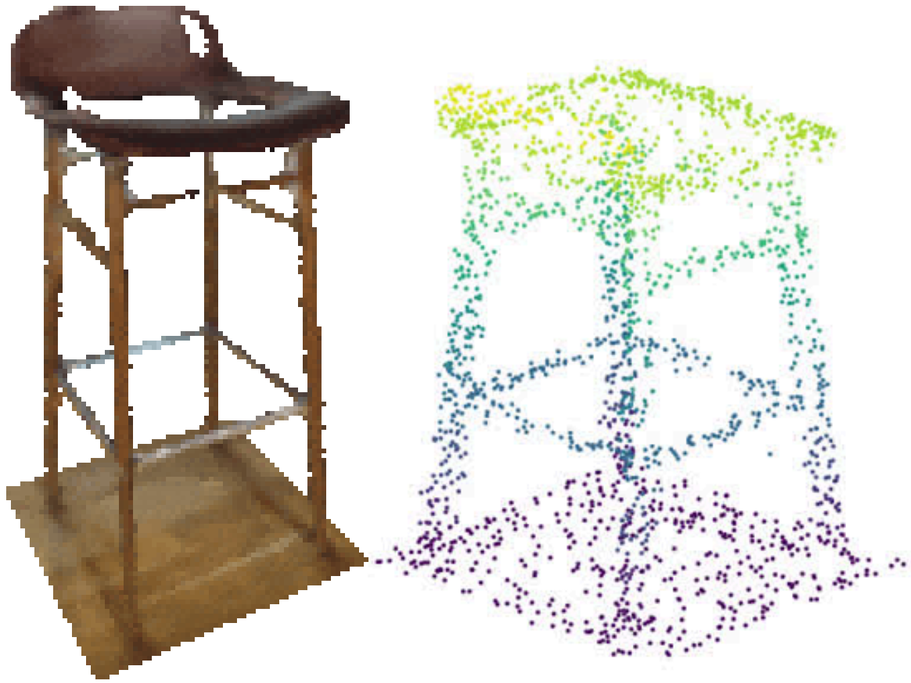
1. person	86.2%
2. flower_pot	4.9%
3. plant	4.3%
4. vase	2.0%
5. curtain	1.2%

SO(3) Invariant Classifier

Best: person (96.1%)

1. person	96.1%
2. airplane	3.1%
3. vase	0.3%
4. bottle	0.1%
5. tent	0.1%

Results: Real world objects



Vanilla GNN Classifier

Best: stool (96.7%)

1. stool	96.7%
2. table	1.0%
3. chair	1.0%
4. sink	0.5%
5. lamp	0.5%

Autoencoder Classifier

Best: stool (97.8%)

1. stool	97.8%
2. bookshelf	0.5%
3. vase	0.5%
4. chair	0.4%
5. night_stand	0.3%

SO(3) Invariant Classifier

Best: stool (57.0%)

1. stool	57.0%
2. desk	9.7%
3. sink	7.2%
4. table	6.1%
5. flower_pot	3.2%

PROBLEMS AND DIRECTIONS

Current problems:

1. Limited data for generation

2. Ambiguous object classes

3. High compute power needed

Future directions:

1. Train on a larger and more diverse dataset

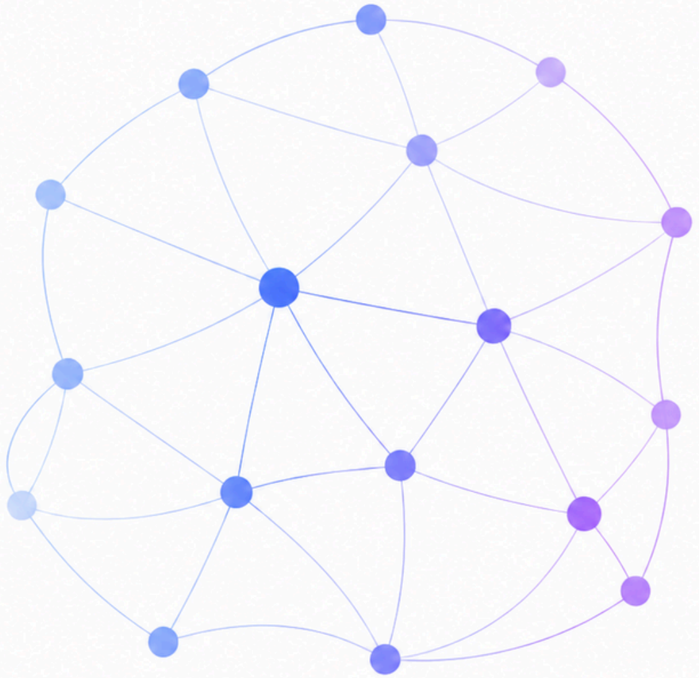
2. Improve Chamfer distance efficiency

3. Implement SO(3) invariance in autoencoders

THANKS!

Applied Machine Learning 2026

Lydia Sakellaridi
Martina Rossi
Tommaso Guarniera



PyG

APPENDIX

3D Point-Cloud Model Setup Details

- Dataset: ModelNet40-style point clouds
- Number of classes: 40
- Points per object: 2,048
- Point features: XYZ coordinates
- Graph construction: k-nearest neighbors
- K value: 20

APPENDIX

1. Vanilla GNN Classifier

Total parameters: 851,624

Architecture

- Input point cloud: 2048 × 3
- Build dynamic KNN graph
- Apply DGCNN / EdgeConv blocks
- Concatenate learned point features
- Apply global mean pooling and max pooling
- Use classifier MLP
- Output: 40 class probabilities

APPENDIX

2. Autoencoder Classifier

Total parameters: 2,034,232

Architecture

- Input point cloud: 2048 × 3
- DGCNN-style encoder extracts point-cloud features
- Bottleneck compresses the object into a latent vector
- Decoder reconstructs the point cloud
- Classification head uses the latent representation
- Output: 40 class probabilities

APPENDIX

3. SO(3)-Invariant Classifier

Total parameters: 853,288

Architecture

- Input point cloud: 2048 × 3
- Center and normalize point cloud
- Build KNN graph
- Use rotation-invariant scalar features:
  - radius
  - radius²
  - distance between points
- Apply invariant message-passing blocks
- Apply global mean pooling and max pooling
- Use classifier MLP
- **Output: 40 class probabilities**

APPENDIX

4. Graph CVAE

Total parameters: 2,572,295

Architecture

- Input point cloud: 2048 × 3
- Input class label: one of 40 classes
- Label embedding adds class information
- DGCNN-style encoder extracts object features
- Encoder predicts latent distribution:
  - mean
  - log variance
- Sample latent vector z
- Decoder generates a point cloud conditioned on:
  - latent vector
  - class label