A BACKGROUND: MATRIX LIE GROUPS

Group action definition If G is a group with identity element e, and $\mathcal{Z}$ is a set, then a group action ϕ is a function $\phi : G \times \mathcal{Z} \to \mathcal{Z}$ such that the following conditions hold:

1. $\phi(e, z) = z$
2. $\phi(g_1, \phi(g_2, z)) = \phi(g_2 g_2, z) \forall g_1, g_2 \in G$

The action ϕ is a left-group action in this case.

Matrix Lie groups Lie groups are continuous groups described by a set of real parameters (Hall, 2003). A matrix Lie group G is a *closed* subgroup of the general linear group $\mathrm{GL}(m, \mathbb{C})$ (the set of invertible complex matrices of size m), with matrix multiplication as group operation. Furthermore, if G is connected, then we can continuously traverse its elements.

Interestingly, many continuous transformations form matrix Lie groups. However, matrix Lie groups lack the typical structure of a vector space. For example adding two rotation matrices does not produce a rotation matrix. For each matrix Lie group however, there's a corresponding Lie algebra $\mathfrak{g}$ which is the tangent space of the group G at the identity. A Lie algebra forms vector space where addition is a closed operation and where the space can be described using basis matrices. Using the matrix exponential $\exp : \mathfrak{g} \to G$, the Lie algebra is the set of all matrices $\mathbf{M}$ such that $\exp(t\mathbf{M}) \in G$ for all real numbers t. If G is connected, then every element of G can be expressed as a product of exponential of elements of the Lie algebra. If we further assume that G is connected and compact, the exponential map is surjective i.e. every element of the group can be expressed as a single exponential of an element of the Lie algebra.

We refer the reader to (Hall, 2003; Stillwell, 2008) for a more detailed presentation of Lie algebra of matrix Lie groups.

B EXPERIMENTAL DETAILS

B.1 TRAINING DETAILS

Generalization gaps for SoTA vision models To measure the generalization gaps of SoTA vision models, we use ImageNet pre-trained models (with the exception of CLIP). For SimCLR we use the pretrained model available on PyTorch Lightning Bolts. For CLIP, we use the pretrained CLIP model available from Hugging Face Transformers. For ViT, we use the pretrained model available in Timm Wightman (2019). For MAE we use the weights and model architecture available from the official repo. For VICReg, we use the weights and model architecture available from the the official repo. For supervised linear evaluation, we train all models for 100 epochs across 7 logarithmically scaled learning rates between 1e-1 and 1-6 using both sgd and Adam with a batch size of 32. We finetune models using the same setup with 7 logarithmically scaled learning rates between 1e-2 and 1e-6. We then evaluate models trained in both setups by measuring the gap between the Top-1 accuracy of training instances in the typical pose and diverse poses in 2.

Lie models and their baselines For MAE/VICReg Lie and their baselines we also use ImageNet pre-trained models. For the self-supervised finetuning phase, we train models for 100 epochs followed by supervised finetuning or linear evaluation of 10,000 or 20,000 steps respectively. During the self-supervised phase, all models are trained on data where half of the instances are seen undergoing pose changes. During the supervised training phases, we vary this proportion and show performance results in the main paper.

Computation : We find our Lie models achieved comparable runtimes on 8 GPUs relative to the baselines for the self-supervised training phase of approximately 2 hours for MAE Lie and 3 hours on VICReg Lie. The overall compute required for the self-supervised phase experiments consisted of the sweeps described in Appendix B.1.2 of approximately 75 runs each requiring 2-3 hours of training on 8 GPU machines. We did find using Option 2 during supervised training (see Appendix B.1.1), to increase runtime since this requires sampling t and computing a full matrix exponential for each sample (e.g., approximately 2.5 hours for linear eval on VICReg Lie versus 1.5 hours for VICReg).

B.1.1 ARCHITECTURES

Storing and re-using learned t in the context of Lie models During the Lie models supervised linear eval or finetuning, we access single images (and not pairs). The supervised classifier uses the Lie backbone to which the single image is fed. At this point we tried two options. Option 1 is to simply retrieve the representation vector $\mathbf{z}$ from the Lie encoder, as with any other baseline model. Option 2 is to use the learned Lie operators to generate “neighbors", that are, transformed versions of that image (represented in embedding space). The intuition behind this option is that we use the Lie generators to create artificial, transformed images (the neighbors of the initial sample) and average the losses over these to increase these neighbors images to improve invariance of the classifier with respect to pose changes.

For Option 2 during training, once the image is fed to the Lie encoder, we generate the representation vector $\mathbf{z}$ for that image but also “neighbors" by (i) sampling random $\mathbf{t}_1, ..., \mathbf{t}_k$ vectors (ii) computing the corresponding $\hat{g}_1, ..., \hat{g}_k$ as in Step 3 of Algorithm 1 (iii) and computing corresponding neighbors $\mathbf{z}^{\hat{g}_1}, ..., \mathbf{z}^{\hat{g}_k}$ as in Step 4 of 1. We consider these the “neighbors" of $\mathbf{z}$, which represent transformed version of the image, but in the embedding space. Then, if we’re in training stage, we back-propagate the loss of the classifier on all neighbors. If we’re in evaluation / testing stage, we perform the prediction on the image and don’t use the neighbors.

We explored both options in our initial experiments and found that they made little difference. Hence, we report MAE Lie and VICReg Lie using Option 2 with 1 neighbor.

Architecture for inferring t To infer t from the ground truth δ we use a multi-layer perceptron with two linear layers connected by a leaky ReLU. The inference network takes as inputs δ and the two embeddings. The network outputs a t matching the dimension of the Lie algebra * batch dimension.

B.1.2 LIE HYPERPARAMETER SWEEPS

MAE Lie sweep: For MAE Lie and its MAE baseline we use a similar procedure. For the self-supervised learning phase, models are trained with a batch size of 64 for 100 epochs. We sweep over three learning rates 5e-4,5e-5,5e-6 using the Adam optimizer along with every combination of each λ in $\{1, 5\}$. The baseline MAE model is trained with λ_{lie} and λ_{euc} set to zero. To cross validate, we use validation accuracy with the same diversity proportion as in training. For the supervised phase, we sweep over learning rates of 5e-3,5e-4,5e-5,5e-6 for finetuning and 5e-3,5e-4,5e-5 for linear evaluation. We show results for the best set of hyperparameters selected based on combined validation accuracy, which samples both diverse and typical poses for instance to match those seen during training. We report the average and standard errors of Top-1 accuracy across three seeds. We find the best hyperparameters via cross validation are $\lambda_{\text{lie}} = 5.0$, $\lambda_{\text{euc}} = 5.0$, and $\lambda_{\text{l2}} = 5.0$ with learning rate of $5e - 4$ for MAE Lie.

VICReg Lie sweep: The procedure for VICReg Lie is similar to MAE Lie, small difference is that we use every combination of each λ in $\{0, 1, 5\}$. The baseline VICReg model is trained with λ_{lie} and λ_{euc} set to zero. We find the best hyperparameters via cross validation are $\lambda_{\text{lie}} = 1.0$, $\lambda_{\text{euc}} = 0.0$, and $\lambda_{\text{l2}} = 1.0$ with a learning rate of $5e - 4$ for VICReg Lie.

B.1.3 TRANSFER EXPERIMENTS

For the experiments in Section 5, we finetune both the baseline MAE model and MAE Lie for 20k steps on ImageNet with a linear classification head. We sweep over the following learning rates $5e - 4, 5e - 5, 5e - 6$ with a batch size of 64 when finetuning each model. We select the best model based on the validation loss.

C Code and Data

We will release code to reproduce our experiments upon acceptance. We base our data generation code on https://github.com/panmari/stanford-shapenet-renderer. We use 52 classes for our analysis after omitting 3 classes which contain overlapping instances. We select 50 instance from each class and partition those into 10% for validation and 15% for testing. For instance show in diverse poses, we show all possible pose configurations in 2D and 3D with a step size of 4 degrees. For 3D rotation we set the angles of all three axes to the same value. The resulting dataset consists of 500,000 unique images. Figure A1 shows a sample of instance images in various poses across five classes.

We provide a PyTorch pseudo-code below for the Lie operator and Lie losses.

Listing 1: Lie operator and loss pseudo-code

```
import torch

def lie_operator(z, t):
"""Applies Lie operator to on given representation z"""
    # L is the set of basis Lie algebra generators
    action = torch.matrix_exp(t*L)
    return action * z


def lie_loss(z, z_hat):
    pred_z_hat = lie_operator(z)
    # standard infonce loss to encourage similarity
    return infonce(z_hat, pred_z_hat)
```

D Additional Results

D.1 Pose changes in 3D

We show additional results for MAE Lie and VICReg Lie for 3D pose changes in Table A1 and A2.

Table A1: **Linear evaluation performance for 3D pose changes as the proportion of diverse instances seen during training varies.** Table reports linear evaluation top-1 accuracy as **mean** ± standard error (+ absolute difference, x relative multiple) to the baseline model.

	known instance in new pose			unknown instance		
diverse proportion	0.05	0.25	0.50	0.05	0.25	0.50
MAE Lie	12.3 ± 0.3 (**+3.0**, **1.32x**)	12.1 ± 0.4 (**+3.1**, **1.35x**)	14.9 ± 0.3 (**+4.8**, **1.48x**)	11.8 ± 0.3 (**+3.1**, **1.35x**)	10.5 ± 0.4 (**+2.0**, **1.24x**)	12.1 ± 0.2 (**+2.6**, **1.28x**)
MAE	9.4 ± 0.1	8.9 ± 0.4	10.1 ± 0.3	8.7 ± 0.1	8.5 ± 0.2	9.4 ± 0.6
VICReg Lie	65.0 ± 0.1 (**+11.3**, **1.21x**)	68.5 ± 0.0 (**+4.3**, **1.07x**)	72.8 ± 0.1 (**+5.7**, **1.09x**)	51.3 ± 0.1 (**+4.7**, **1.10x**)	57.4 ± 0.1 (**+2.2**, **1.04x**)	59.6 ± 0.1 (**+0.9**, **1.02x**)
VICReg	53.7 ± 0.0	64.2 ± 0.1	67.1 ± 0.1	46.6 ± 0.1	55.2 ± 0.1	58.7 ± 0.2

Table A2: **Finetuning performance for 3D pose changes as the proportion of diverse instances seen during training varies.** Table reports linear evaluation top-1 accuracy as **mean** ± standard error (+ absolute difference, x relative multiple) to the baseline model.

	known instance in new pose			unknown instance		
diverse proportion	0.05	0.25	0.50	0.05	0.25	0.50
MAE Lie	43.2 ± 0.1 (**+8.6**, **1.25x**)	54.9 ± (**+12.8**, **1.31x**)	59.5 ± 0.4 (**+13.9**, **1.31x**)	26.3 ± 0.2 (**+7.6**, **1.40x**)	35.3 ± (**+11.4**, **1.48x**)	39.3 ± 1.0 (**+12.1**, **1.44x**)
MAE	34.6 ± 0.2	42.0 ± 0.3	45.6 ± 0.6	18.8 ± 0.2	23.9 ± 0.3	27.2 ± 0.4

E Applying the Lie operator to SimCLR (Chen et al., 2020)

We also experiment applying our Lie operator to the SimCLR architecture (Chen et al., 2020), where L_{ssl} is the regular SimCLR Loss. Since the Lie models receive frame pairs as input, we construct an additional baseline, SimCLR Frames, with access to the same pairs of frames. In addition to the

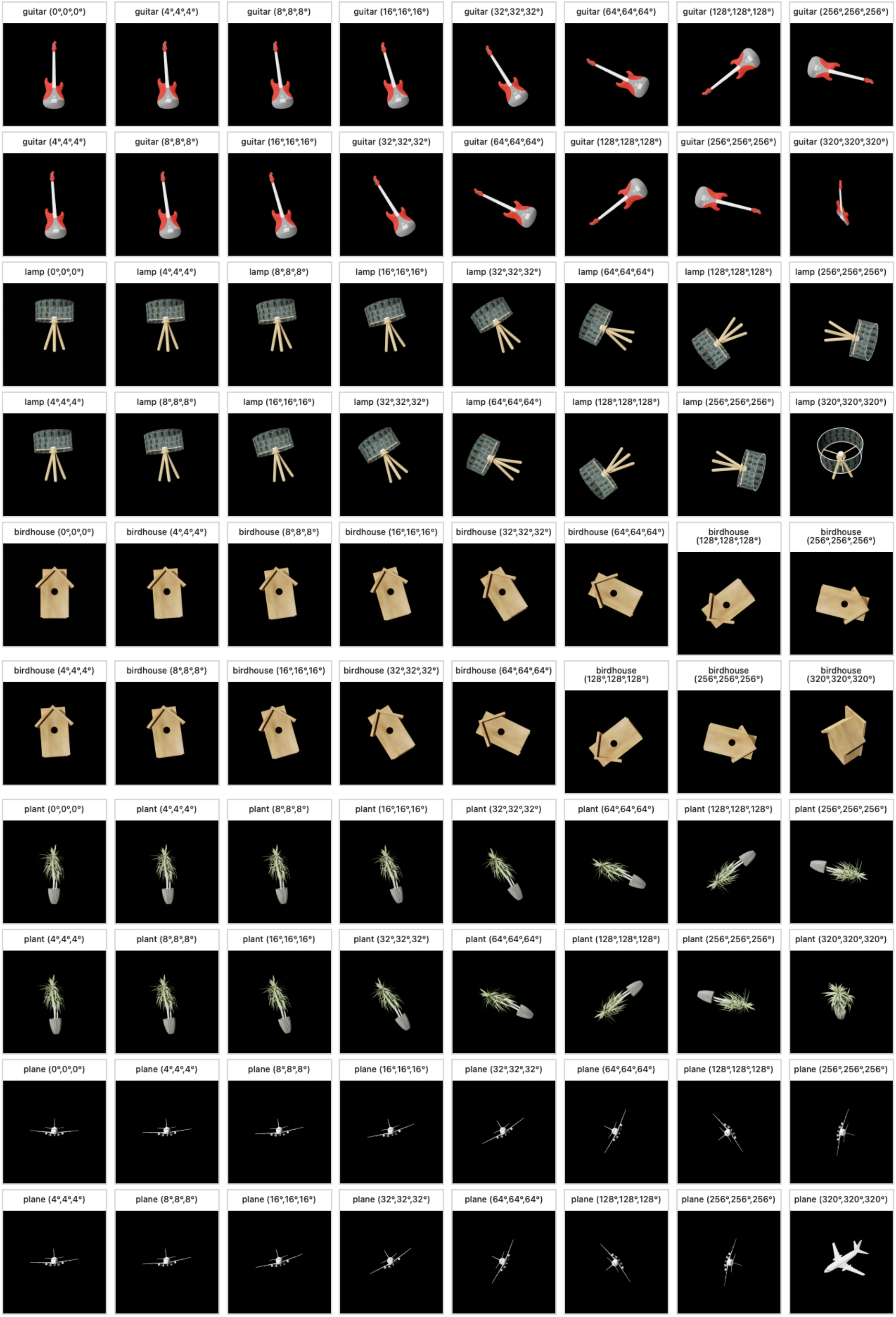

Figure A1: **Sample images of the shapes dataset in various 2D and 3D poses**. From the upper left onward is class guitar, lamp, birdhouse, plant, and plane. The first image for each class is the canonical pose followed by 4, 8, 16, 32, 64, 128, 256 planar and 3D rotations. The last image for each class is (320, 320, 320) degree 3D pose. In our training and evaluation, we use full span of 2D and 3D pose change with a step size of 4 degrees.

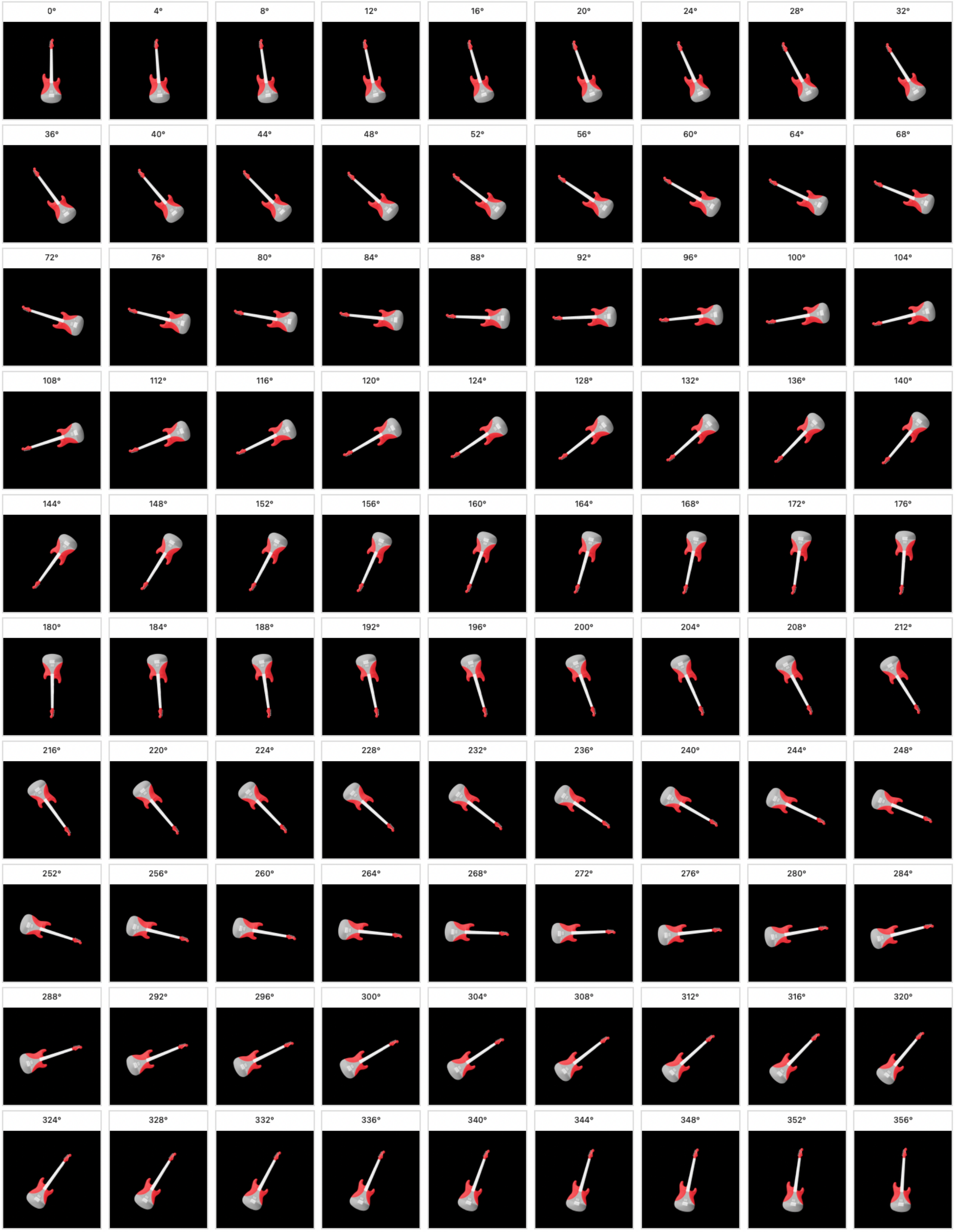

Figure A2: **Guitar object instance in all 2D pose configurations.** The poses vary by a step size of 4 degrees with the pose angle of each image shown above.

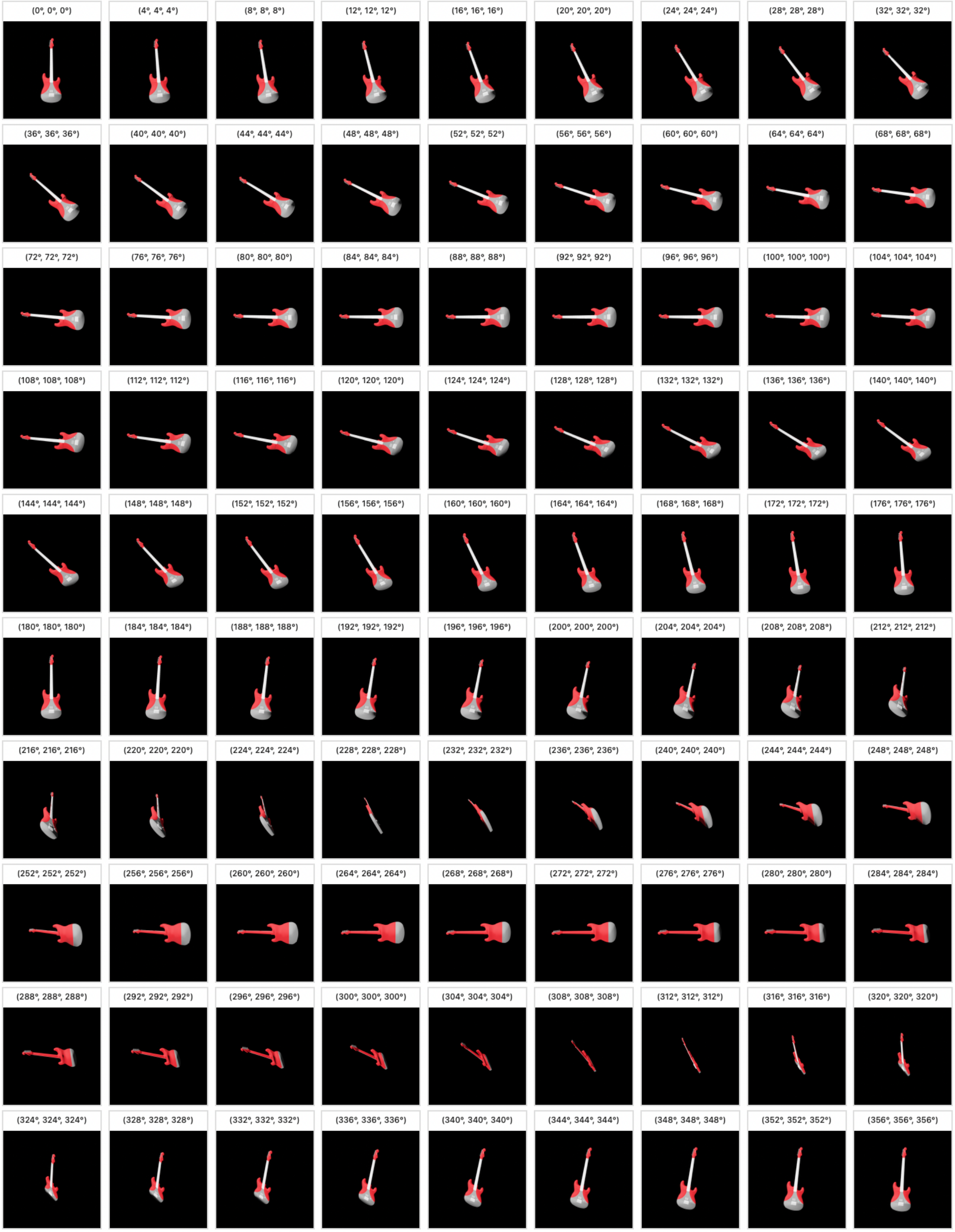

Figure A3: **Guitar object instance in all 3D pose configurations.** The poses vary by a step size of 4 degrees with the pose angle of each image shown above.

Table A3: **SimCLR Lie versus baselines: Linear evaluation robustness to pose changes as the proportion of diverse instances seen during training varies.** Table reports linear evaluation top-1 accuracy as **mean** ± standard error (+ absolute difference, x relative multiple) to the baseline model.

	known instance in new pose			unknown instance		
diverse proportion top-1 accuracy (%)	5%	25%	50%	5%	25%	50%
SimCLR Lie	**56.0** ± 0.2 (**+5.8**, **1.12x**)	**65.4** ± 0.1 (**+3.4**, **1.05x**)	**70.0** ± 0.1 (**+4.0**, **1.06x**)	**45.7** ± 0.1 (**+2.6**, **1.06x**)	**56.0** ± 0.1 (**+1.7**, **1.03x**)	**59.0** ± 0.2 (**+0.8**, **1.01x**)
SimCLR	50.2 ± 0.3	62.0 ± 0.1	66.0 ± 0.1	43.0 ± 0.3	54.3 ± 0.0	58.3 ± 0.2
SimCLR Frames	49.1 ± 0.2	60.6 ± 0.1	65.2 ± 0.0	42.2 ± 0.2	53.3 ± 0.1	58.0 ± 0.0

standard InfoNCE loss for SimCLR we add an additional InfoNCE loss to encourage similarity in the embeddings of the two frames. This baseline accounts for any performance gain simply due to the extra organization found by pairing frames. We report results of SimCLR, SimCLR Frames, and SimCLR Lie in Tables A3, A4, A5 and A6. Across all evaluation settings, we find our Lie operator extension of SimCLR matches or outperforms SimCLR Frames and SimCLR baselines, with best gains in the linear evaluation setting.

Table A4: **SimCLR Lie versus baselines: Finetuning robustness to pose changes as the proportion of diverse instances seen during training varies.** Table reports finetuning Top-1 Accuracy as **mean** ± standard error (+ absolute difference, x relative multiple) to the baseline model.

	known instance in new pose			unknown instance		
diverse proportion top-1 accuracy (%)	5%	25%	50%	5%	25%	50%
SimCLR Lie	**55.1** ± 0.4 (**+0.9**, **1.02x**)	**65.9** ± 0.6 (**+1.2**, **1.02x**)	**70.9** ± 0.4 (**+0.7**, **1.01x**)	**45.9** ± 0.1 (**+0.5**, **1.01x**)	**55.9** ± 0.2 (-0.0, 1.00x)	**61.8** ± 0.8 (**+0.7**, **1.01x**)
SimCLR	54.2 ± 0.8	64.7 ± 0.2	70.2 ± 0.2	45.4 ± 0.3	56.0 ± 0.2	61.1 ± 0.4
SimCLR Frames	53.9 ± 0.2	64.8 ± 0.6	70.7 ± 0.2	45.2 ± 0.2	55.8 ± 0.1	61.8 ± 0.5

SimCLR Training details We developed SimCLR Lie early in our experimental study, and explain here the procedure for training. In the case of SimCLR Lie for self-supervised learning phase, we cross-validated using top 1 online validation combined accuracy but we filter models such that the validation L_{Lie} loss, evaluated on the 2D rotated validation set, at the last epoch, is 60% smaller or equal to its first epoch's value. We found that online validation combined accuracy was not enough as a metric to shed light on the behavior of our Lie model on the representational space and do this filtering to ensure that the Lie operator bring the transformed sample close to its target. Also, note that in linear evaluation and finetuning, we report SimCLR Lie using Option 1 (regular supervised training of the classifier). For self-supervised learning phase:

- We cross-validate with $\lambda_{\text{lie}} \in \{1, 5, 10\}, \lambda_{\text{ssl}} \in \{1, 5, 10\}, \lambda_{\text{euc}} \in \{1, 2\}$. Best hyper-parameters chosen with the cross-validation procedure explained is $\lambda_{\text{lie}} = 10, \lambda_{\text{ssl}} = 10, \lambda_{\text{euc}} = 1$.
- We cross-validate with learning rates in $5e-5, 5e-6$.

For supervised linear evaluation:

- We cross-validate with learning rates in $5e-3, 5e-4, 5e-5$.

For supervised finetuning:

- We cross-validate with learning rates in $5e-3, 5e-4, 5e-5, 5e-6$.

F TRAINING CURVES

Figure A4 and A5 reports the Lie loss during self-supervised finetuning phase (for diverse sets, as it is by definition 0 for the canonical sets, where the two samples of the pair are in the same canonical pose) and top 1 accuracy during classifier finetuning respectively, for the reported MAE Lie model.

Table A5: **SimCLR Lie versus baselines: Linear evaluation performance for 3D pose changes as the proportion of diverse instances seen during training varies.** Table reports linear evaluation top-1 accuracy as **mean** ± standard error (+ absolute difference, x relative multiple) to the baseline model.

		known instance in new pose			unknown instance	
diverse proportion	0.05	0.25	0.50	0.05	0.25	0.50
SimCLR Lie	50.9 ± 0.1 (**+4.5**, **1.10x**)	61.0 ± 0.1 (**+2.4**, **1.04x**)	64.7 ± 0.1 (**+2.7**, **1.04x**)	43.2 ± 0.1 (**+2.4**, **1.06x**)	53.6 ± 0.1 (**+0.9**, **1.02x**)	56.2 ± 0.1 (**+0.4**, **1.01x**)
SimCLR	46.4 ± 0.2	58.6 ± 0.1	61.9 ± 0.2	40.8 ± 0.3	52.8 ± 0.1	55.8 ± 0.2
SimCLR Frames	45.6 ± 0.2	57.4 ± 0.1	61.0 ± 0.1	40.0 ± 0.2	51.2 ± 0.1	55.4 ± 0.1

Table A6: **SimCLR Lie versus baselines: Finetuning performance for 3D pose changes as the proportion of diverse instances seen during training varies.** Table reports linear evaluation top-1 accuracy as **mean** ± standard error (+ absolute difference, x relative multiple) to the baseline model.

		known instance in new pose			unknown instance	
diverse proportion	0.05	0.25	0.50	0.05	0.25	0.50
SimCLR Lie	49.6 ± 0.5 (**+1.0**, **1.02x**)	62.3 ± 0.2 (**+0.4**, **1.01x**)	66.9 ± 0.3 (**+0.1**, **1.00x**)	42.0 ± 0.3 (**+0.8**, **1.02x**)	54.2 ± 0.2 (**-0.7**, **0.99x**)	59.8 ± 0.5 (**-0.2**, **1.00x**)
SimCLR	48.6 ± 0.5	61.9 ± 0.4	66.8 ± 0.2	41.3 ± 0.6	54.9 ± 0.1	60.1 ± 0.3
SimCLR Frames	48.7 ± 0.4	62.2 ± 0.5	67.3 ± 0.1	41.5 ± 0.2	55.1 ± 0.5	59.7 ± 0.5

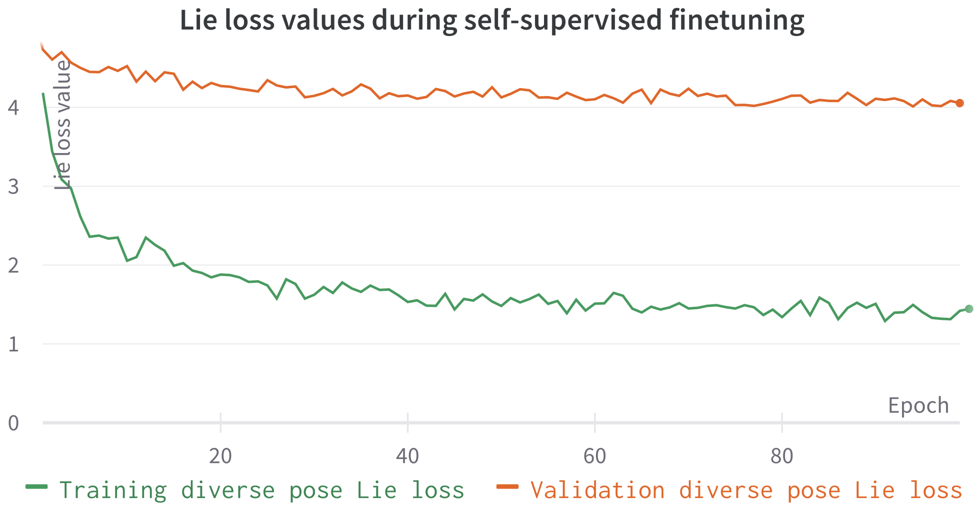

Figure A4: Lie loss during self-supervised finetuning phase for the MAE Lie model.

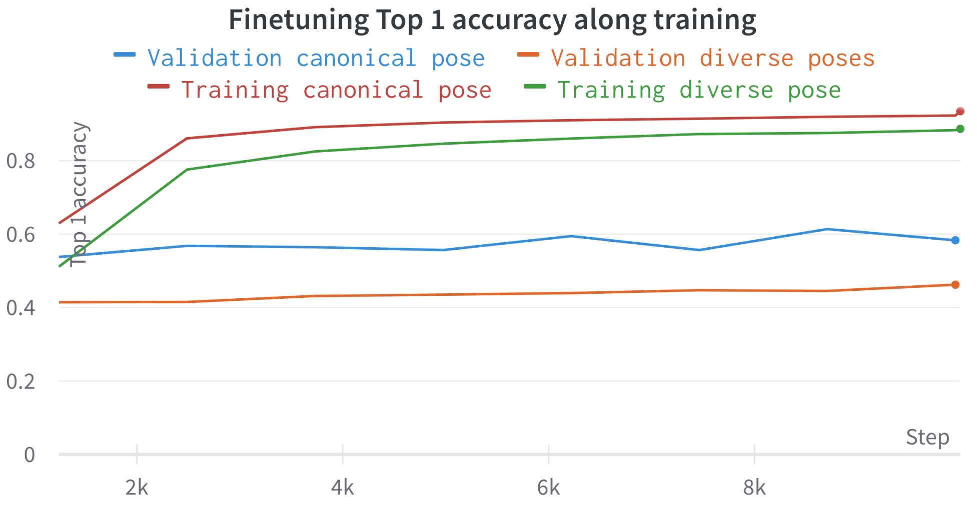

Figure A5: Top 1 accuracy during classifier finetuning for the MAE Lie model.