# Flow Factorized Representation Learning –Supplementary Material–

**Yue Song**[1,2], **Andy Keller**[2], **Nicu Sebe**[1], and **Max Welling**[2]

[1]Department of Information Engineering and Computer Science, University of Trento, Italy
[2]Amsterdam Machine Learning Lab, University of Amsterdam, the Netherlands
yue.song@unitn.it

## A Supplementary Material

### A.1 Pseudo codes

```
import torch

#Randomly sample a transformation at each iteration
index = torch.randint(0, potential_number)
x_bar = sequence_generation(index)

#Generating index according to the supervision setting
if training_mode = "supervised":
  index_potential = index
elif training_mode = "weakly-supervised":
  index_potential = q_k(x_bar)

#initial element of the sequence
z, rho_z = flow_vae(x_bar[0])

#Future elements of the sequence obtained by latent flow
for t in range(0,T)
   PDE_loss, delta_z, delat_rho_z = HJ_PDE(index_potential,z,t)

   #Updates in the sample and probability space
   z = z + delta_z
   rho_z = rho_z + delat_rho_z

   #Inference at every intermediate step
   hat_xt = flow_vae.inference(z)

   #Loss: PDE loss + reconstrutction loss + KL div
   loss += PDE_loss + CE(hat_xt,x_bar[t]) + KL(rho_z, prior_rho_z)

#KL div for index prediction (weakly-supervised setting)
if training_mode = "weakly-supervised":
   loss += KL(index_potential,index)

loss.backward()
optimizer.step()
```

Figure 1: Pytorch-like pseudo codes for training our flow-factorized VAE.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Fig. 1 displays the Pytorch implementation for training our flow-factorized VAE under different supervision settings. Here we omit the computation of HJ PDEs for concisity.

### A.2 Implementation details

**Common settings.** During the training stage, we randomly sample one single transformation at each iteration. The batch size is set to 128 for both datasets. We use Adam optimizer and the learning rate is set as $1e{-}4$ for all the parameters. The encoder consists of four stacked convolution layers with the activation function ReLU, while the decoder is comprised of four stacked transposed convolution layers. For the prior evolution, the diffusion coefficient $D_k$ is initialized with 0 and we set it as a learnable parameter for distinct $k$. For `MLPs` that parameterize the potential $u(\boldsymbol{z}, t)$ and the force $f(\boldsymbol{z}, t)$, we use the sinusoidal positional embeddings [12] to embed the timestep $t$, and use linear layers for embedding the latent code $\boldsymbol{z}$. `Tanh` gates are applied as the activation functions of the `MLPs`. All the experiments are run on a single NVIDIA Quadro RTX 6000 GPU.

**MNIST.** The input images are of the size $28{\times}28$. The sequence of each transformation contains 9 states of variations. The scaling transformation scales the image from 1.0 up to 1.8 times. The rotation transformation rotates the object by maximally 80 degrees, and the coloring transformation adjusts the image hue from 0 to 340 degrees. The model is trained for $90,000$ iterations.

**Shapes3D.** The input images are resized to $64{\times}64$. Each transformation sequence consists of 8 images. The model is also trained for $90,000$ iterations.

**Falcol3D and Isaac3D.** The input images are in the resolution of $128{\times}128$. We use the self-contained transformations of the datasets, which mainly comprise variations of lighting conditions and viewpoints in indoor 3D scenes for Falcolr3D, and different robot arm movements in dynamic environments for Isaac3D.

**Weakly-supervised setting.** For the `Gumbel-Softmax` trick, we re-parameterize $q_\gamma(k|\bar{\boldsymbol{x}})$ by

$$y_i = \frac{e^{\frac{x_i+g_i}{\tau}}}{\sum_i e^{\frac{x_i+g_i}{\tau}}} \tag{1}$$

where $x_i$ is the category prediction, $g_i$ is the sample drawn from Gumbel distributions, and $\tau$ is the small temperature to make `softmax` behave like `argmax`. We take the 'hard' binary prediction in the forward pass and use the straight-through gradient estimator [2] during backpropagation. The temperature $\tau$ is initialized with 1 and is gradually reduced to 0.05 with the annealing rate $3e{-}5$.

**Baselines.** For the disentanglement methods, we largely enrich the original MNIST dataset by adding the transformed images of the whole sequence. This makes it possible for both $\beta$-VAE and FactorVAE to learn the given transformations in an unsupervised manner. For tuning the interpolation range, we start from the initial value $z_i$ and traverse till the appropriate bound which is selected from the range $[-5, 5]$ with the interval of 0.1.

### A.3 Disentanglement metrics

There are many traditional disentanglement metrics [7, 5, 4], but they are designed for single-dimension traversal methods. These metrics assume and require that each latent dimension is responsible for one semantic and manipulating single dimensions of the latent variable would involve distinct output transformations. However, for the recent disentanglement methods including ours [8, 11, 9], there emerges a more realistic disengagement setting: all the latent dimensions are perturbed by vectors for meaningful output variations. When it comes to these vector-based disentanglement methods, their scores of disentanglement metrics would drop considerably and cannot be compared with those single-dimension baselines.

Table 1: VP Scores (%) on MNIST.

| Training Set Split | Ours | PoFlow | TVAE | FactorVAE | $\beta$-VAE |
|---|---|---|---|---|---|
| 10% | **95.69** | 93.05 | 89.91 | 85.92 | 87.31 |
| 1% | **92.71** | 91.27 | 88.15 | 84.46 | 85.25 |

Table 2: VP Scores (%) on Shapes3D.

| Training Set Split | Ours | PoFlow | TVAE | FactorVAE | $\beta$-VAE |
|---|---|---|---|---|---|
| 10% | **95.92** | 91.48 | 88.27 | 84.49 | 85.91 |
| 1% | **77.03** | 72.32 | 68.39 | 63.83 | 65.78 |

Nonetheless, certain disentanglement metrics such as VP scores [13] can be leveraged as they do not pose any assumptions on the latent space but only require image pairs $[\boldsymbol{x}_0, \boldsymbol{x}_T]$ of different

transformations for evaluation. The VP metric adopts the few-shot learning setting (using $1\%$ or $10\%$ of the dataset as the training set) and takes a lightweight neural network for learning to classify image pairs $[\boldsymbol{x}_0, \boldsymbol{x}_T]$ of different attributes. The generalization ability ( *i.e.,* validation accuracy) can be thus regarded as a reasonable surrogate for the disentanglement ability. Table 1 and 2 present the VP scores of all the baseline methods on MNIST and Shapes3D. To ensure a fair comparison, for FactorVAE and $\beta$-VAE, we choose the dimensions with the lowest equivariance errors to generate image pairs of different transformations. Our method outperforms the previous disentanglement baselines and achieves superior performance on the VP scores. This indicates that our flow-factorized VAE has better disentanglement ability.

### A.4 Ablation studies

Table 3: Equivariance error of different priors.

| Prior | Scaling | Rotation | Coloring |
|---|---|---|---|
| **SG** | 190.24±2.18 | 158.93±3.25 | 164.18±2.77 |
| **MoG** | 188.23±2.45 | 157.79±2.86 | 161.49±2.62 |
| **VAMP** | 192.81±3.67 | 161.47±4.12 | 162.97±3.89 |
| **Diffusion** | **185.42±2.35** | **153.54±3.10** | **158.57±2.95** |

Table 4: Equivariance error of different PDEs.

| Prior | Scaling | Rotation | Coloring |
|---|---|---|---|
| **Heat** | 223.95±3.38 | 212.47±3.85 | 207.66±2.91 |
| **FP** | 211.54±3.17 | 188.59±3.92 | 194.73±3.09 |
| **OHJ** | 190.43±2.48 | 163.87±3.03 | 162.38±2.86 |
| **GHJ** | **185.42±2.35** | **153.54±3.10** | **158.57±2.95** |

**Impact of different priors.** We use diffusion equations to model the prior evolution as random particle movement. It would also be intriguing to choose other priors commonly used in the VAE literature, such as Standard Gaussian (SG) priors $\mathcal{N}(0, 1)$, mixture of Gaussian (MoG) priors $\sum w_i \mathcal{N}(\mu_i, \sigma_i^2)$, and VAMP priors [10] which average aggregated posterior of $N$ pseudo-inputs as $1/N \sum_n q(\boldsymbol{z}_n)$. Table 3 presents the equivariance error of different priors on MNIST. Among these priors, our diffusion equations achieve the best performance. This meets our assumption that modeling the prior evolution as a diffusion process suits more the random motion. Nonetheless, we see that the performance gap between each baseline is narrow, which somehow implies that the impact of different priors is limited.

**Impact of different PDEs.** We apply the generalized HJ (GHJ) equation as the PINN constraint in order to achieve dynamic OT. It would be also interesting to try other commonly used PDEs. We compare our GHJ with the ordinary HJ (OHJ) equation, the Fokker Planck (FP) equation, and the heat equation. Table 4 compares the equivariance error of PDEs on MNIST. Our GHJ and OHJ equations achieve the best performance as they both satisfy the condition of dynamic OT. This empirical evidence indicates that the OT theory can indeed model better latent flow paths. Moreover, our GHJ outperforms the OHJ by a slight margin. We attribute this advantage to the external driving force $f(\boldsymbol{z}, t)$ which gives us more flexibility and dynamics in modeling the velocity fields $\nabla u^k$.

Table 5: Equivariance error on MNIST of a different number of transformations ($K$).

| $K$ | Scaling | Rotation | Coloring |
|---|---|---|---|
| 1 | 185.27±2.59 | – | – |
| 2 | 185.78±2.21 | 154.29±2.87 | – |
| 3 | 185.42±2.45 | 153.54±3.10 | 158.57±2.95 |

Table 6: Equivariance error on MNIST of different sequence lengths (T).

| Sequnce Length ($T$) | Scaling | Rotation |
|---|---|---|
| 9 | 185.42±2.35 | 153.54±3.10 |
| 12 | 214.47±2.59 | 198.72±2.89 |

**Impact of different $K$.** We conduct an ablation study on the impact of the number of transformations on MNIST and present the evaluation results in Table 5. As indicated above, in general, the performance is not affected by the number of transformations being applied. The fluctuation of the results when $K$ varies can be sufficiently negligible. We expect that this is because the transformations are learned by distinct potentials (which are implemented as $K$ different MLPs). Each flow evolves along with the gradient field $\nabla u$ on the potential landscape $u$; *having multiple latent flows defined on different potential landscapes therefore does not interfere with each other.*

**Impact of different sequence lengths.** Regarding the impact of sequence lengths, if the sequence is longer and has larger variations, generally the equivariance error would be worse. To better illustrate this point, we perform an ablation study on MNIST and present the results in Table 6. Specifically, we change the sequence length from 9 to 12, which increases the extent of scaling from maximally $1.8$ times to maximally $2.1$ times, and increases the rotation angle from maximally $80$ degrees to maximally $110$ degrees. As can be observed, the equivariance error gets larger when the sequence

becomes longer and the variations are larger. Notice that even for the longer sequence, our method still outperforms other baselines with shorter sequences.

### A.5 HJ equations as dynamic optimal transport

We now turn to introduce why HJ equations could minimize the Wasserstein distance. As stated in [1], the $L_2$ Wasserstein distance can be re-formulated in the fluid mechanical interpretation as

$$W^2 = \inf \int_D \int_0^1 \frac{1}{2} \rho(x,t) v(x,t)^2 \, dx \, dt \tag{2}$$

where the density satisfies the continuity equation ($\partial_t \rho = -\nabla \cdot (\rho(x,t)v(x,t))$. If we introduce the momentum $m(x,t) = \rho(x,t)v(x,t)$ and two Lagrange multipliers $u$ and $\lambda$, the Lagrangian function of the Wasserstein distance would be:

$$L(\rho, m, \phi) = \int_D \int_0^1 \frac{||m||^2}{2\rho} + u(\partial_t \rho + \nabla \cdot m) - \lambda(\rho - s^2) \tag{3}$$

where the second term is the equality constraint, and the third term is an equality constraint with a slack variable $s$. Using integration by parts formula, the above equation can be re-written as

$$L(\rho, m, \phi) = \int_D \int_0^1 \frac{||m||^2}{2\rho} + \int_D u\rho|_0^1 - \int_D \int_0^1 (\partial_t u\rho + \nabla u \cdot m) - \lambda(\rho - s^2) \tag{4}$$

Based on the set of Karush–Kuhn–Tucker (KKT) conditions ($\partial_m L = 0$, $\partial_u L = 0$, $\partial_\rho L = 0$, and $\lambda \geq 0$), we would have:

$$\begin{cases} \partial_m L = \frac{m}{\rho} - \nabla u = v - \nabla u = 0 \\ \partial_u L = \partial_t \rho + \nabla \cdot m = 0 \\ \partial_\rho L = -\frac{||m||^2}{2\rho^2} - \partial_t u - \lambda = -\frac{1}{2}||v||^2 - \partial_t u - \lambda = 0 \end{cases} \tag{5}$$

where the first condition indicates that the gradient $\nabla u$ acts as the velocity field, and the third condition implies the optimal solution is given by the generalized HJ equation:

$$\partial_t u + \frac{1}{2}||\nabla u||^2 = -\lambda \leq 0 \tag{6}$$

We thus apply the generalized HJ equation (*i.e.,* $\partial_t u + \frac{1}{2}||\nabla u||^2 \leq 0$) as the constraints. We further use an extra negative force because this would give more dynamics for modeling the posterior flow.

### A.6 More visualizations

Fig. 2, 3, and 4 display more visualization results of the latent evolution on MNIST, Shapes3D, Falcol3D and Isaac3D, respectively. Across all the datasets, our method presents precise control of the given transformations. Fig. 5 and 6 show more latent evolution results of switching transformations (top) and combining transformations (bottom) on MNIST and Shapes3D, respectively. Fig. 7 also visualizes a few examples of superposing and switching transformation on Falcol3D and Isaac3D. Our latent flows learn to compose or switch different transformations precisely and flexibly.

## References


[1] Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 2000.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[3] Chris Burgess and Hyunjik Kim. 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

[4] Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *NeurIPS*, 2018.

[5] Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *ICLR*, 2018.

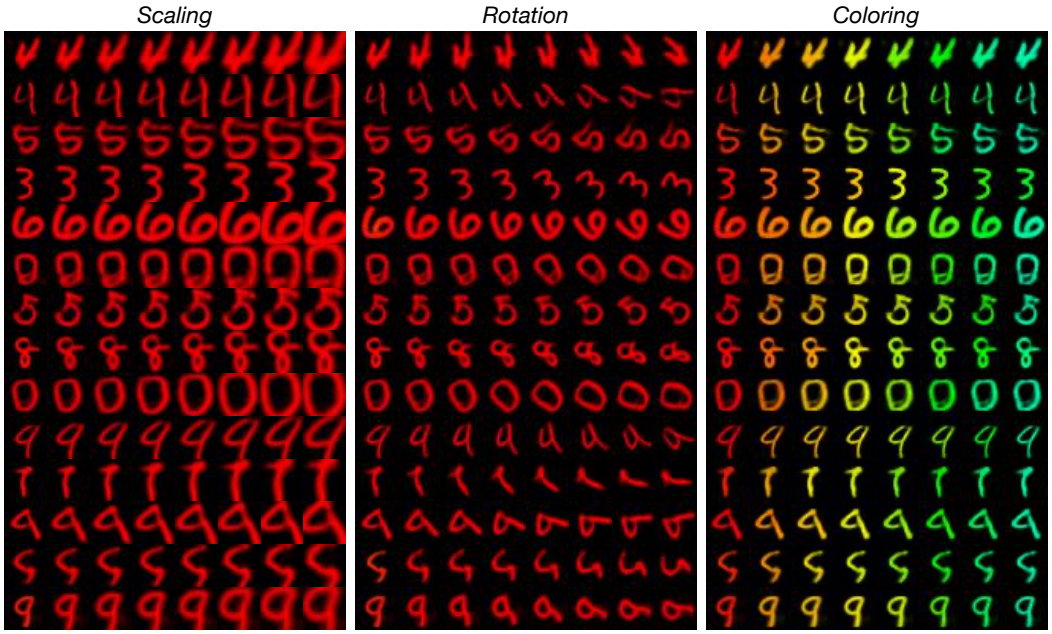


Figure 2: More visualizations of the learned latent flows on MNIST [6].

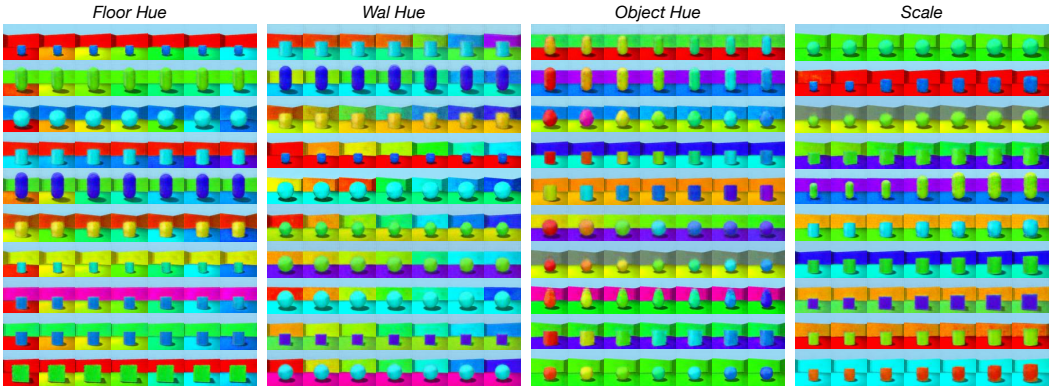


Figure 3: More visualizations of the learned latent flows on Shapes3D [3].

[6] Yann LeCun. The mnist database of handwritten digits. 1998. URL `http://yann.lecun.com/exdb/mnist/`.

[7] Karl Ridgeway and Michael C Mozer. Learning deep disentangled embeddings with the f-statistic loss. *NeurIPS*, 2018.

[8] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *CVPR*, 2021.

[9] Yue Song, Andy Keller, Nicu Sebe, and Max Welling. Latent traversals in generative models as potential flows. In *ICML*. PMLR, 2023.

[10] Jakub Tomczak and Max Welling. Vae with a vampprior. In *AISTATS*. PMLR, 2018.

[11] Christos Tzelepis, Georgios Tzimiropoulos, and Ioannis Patras. WarpedGANSpace: Finding non-linear rbf paths in GAN latent space. In *ICCV*, 2021.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[13] Xinqi Zhu, Chang Xu, and Dacheng Tao. Learning disentangled representations with latent variation predictability. In *ECCV*, 2020.

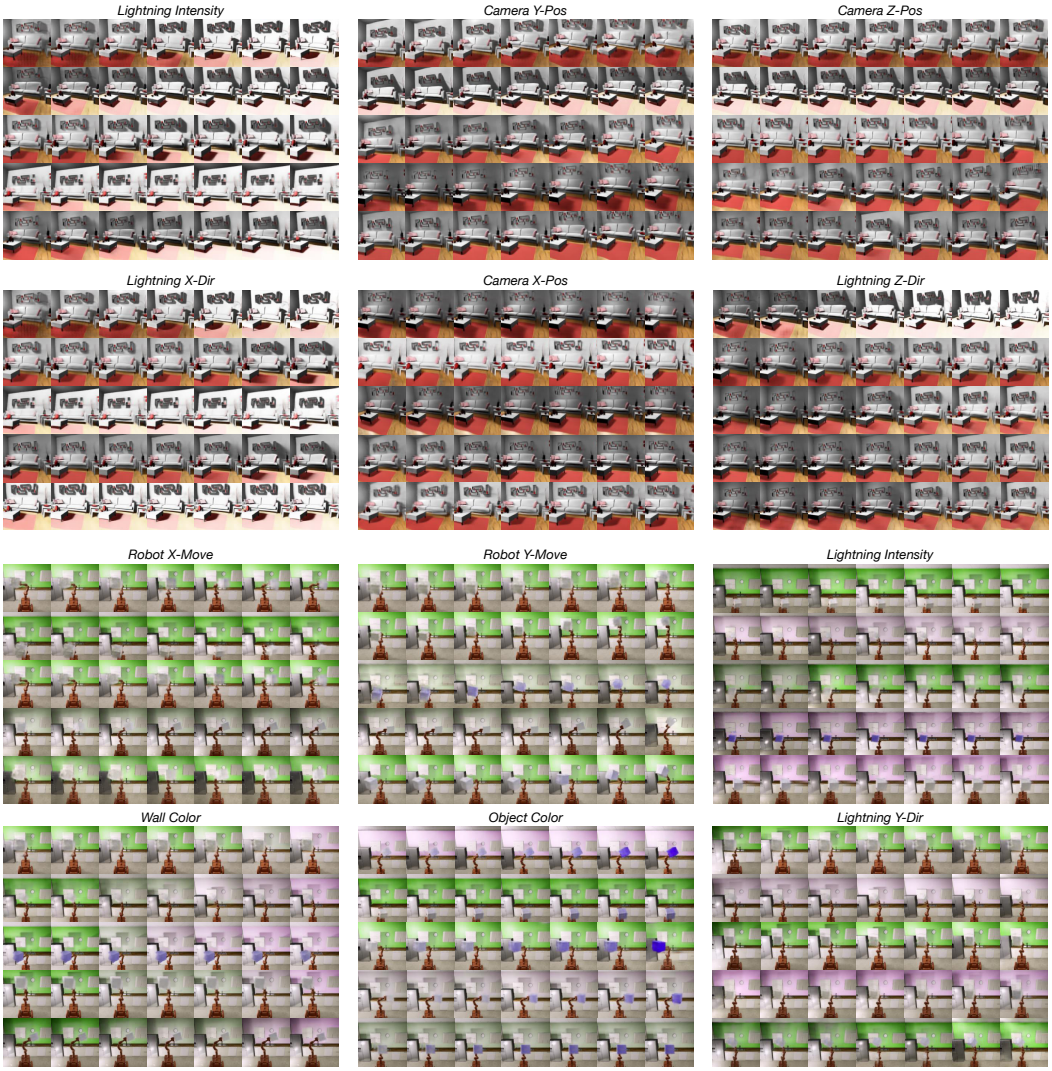


Figure 4: Exemplary visualizations of learned latent flows on Falcol3D (*top*) and Isaac3D (*bottom*).

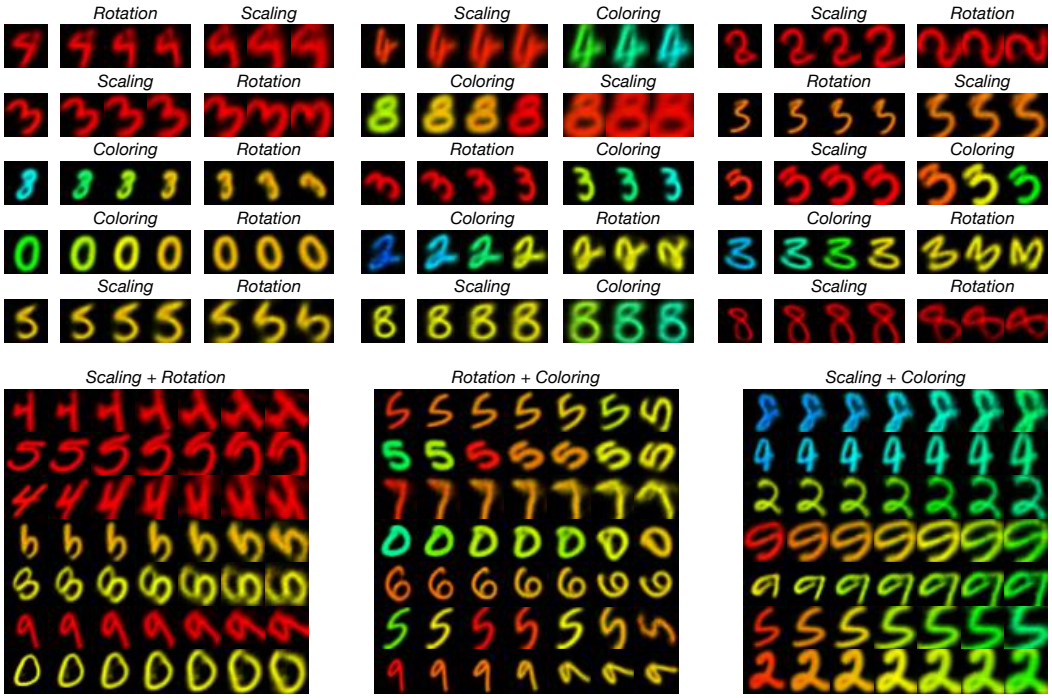


Figure 5: More visualizations of switching and superposing transformations on MNIST [6].

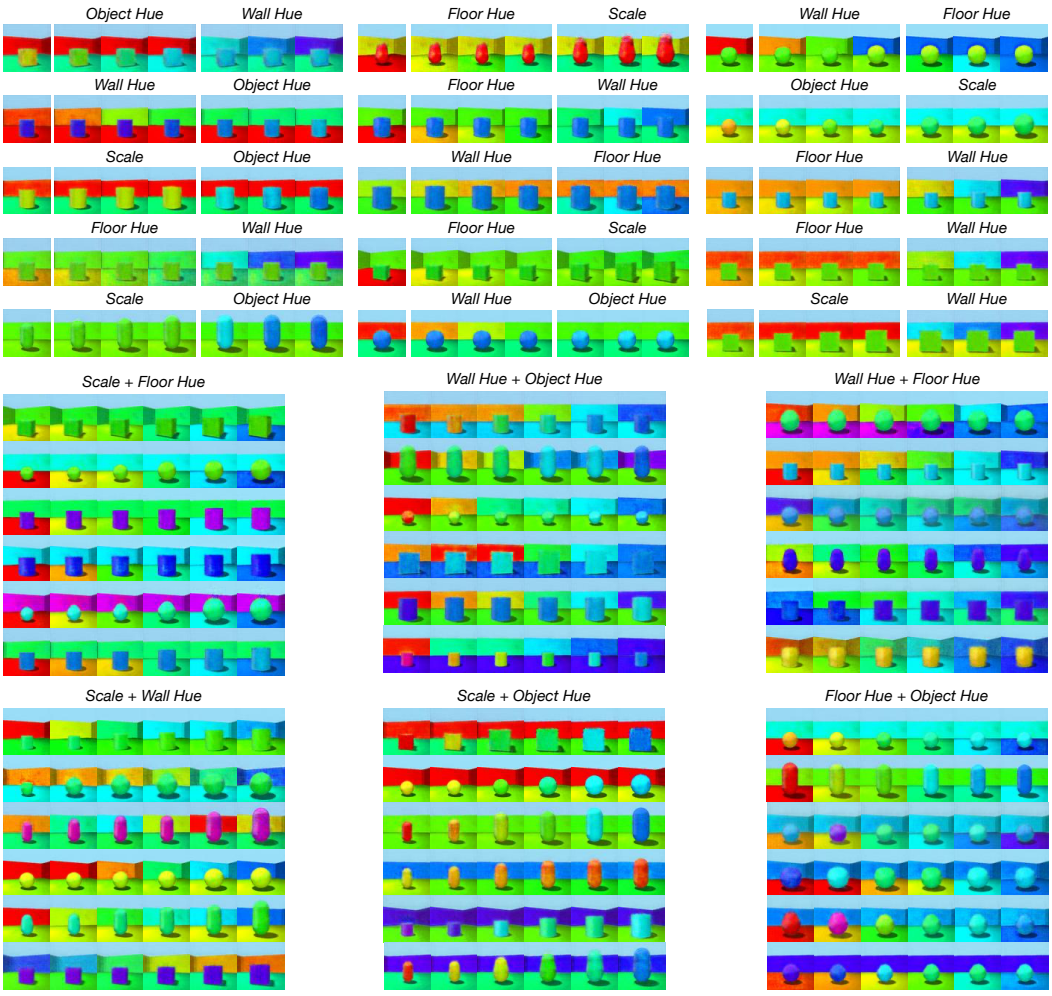


Figure 6: More visualizations of switching and superposing transformations on Shapes3D [3].

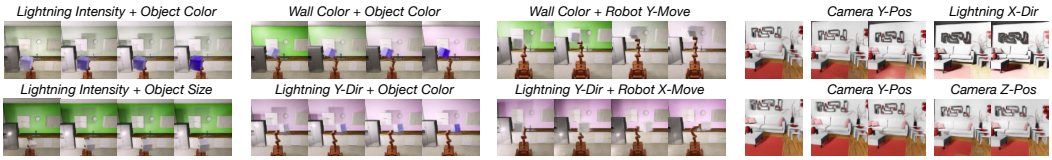


Figure 7: Exemplary visualization results of superposing transformations on Isaac3D (*left*) and switching transformations on Falcol3D (*right*).