Appendix of Commutative Lie Group VAE for Disentanglement Learning

Xinqi Zhu  Chang Xu  Dacheng Tao

1. Proof of Proposition 1

Proposition 1. If $A_iA_j = A_jA_i, \forall i, j$, then
\[
\exp(t_1A_1 + t_2A_2 + ... + t_mA_m) = \exp(t_1A_1)\exp(t_2A_2)...\exp(t_mA_m).
\]

Proof. By (Hall, 2013) Proposition 2.3, we have that if $XY = YX$, $\exp(X + Y) = \exp(X)\exp(Y) = \exp(Y)\exp(X)$.

We also have:
\[
t_1A_1(t_2A_2 + t_3A_3 + ... + t_mA_m)
- (t_2A_2 + t_3A_3 + ... + t_mA_m)t_1A_1
= (t_1A_1t_2A_2 - t_2A_2t_1A_1) + ... + t_1t_mA_mA_m - t_mA_mt_1A_1
= t_1t_2(A_1A_2 - A_2A_1) + ... + t_1t_mA_mA_m - A_mA_1
= 0,
\]
so
\[
t_1A_1(t_2A_2 + t_3A_3 + ... + t_mA_m)
= (t_2A_2 + t_3A_3 + ... + t_mA_m)t_1A_1
\]
Then we have:
\[
\exp(t_1A_1 + t_2A_2 + ... + t_mA_m)
= \exp(t_1A_1)\exp(t_2A_2 + ... + t_mA_m)
\]
and
\[
\exp(t_1A_1)\exp(t_2A_2 + ... + t_mA_m) = \exp(t_1A_1)\exp(t_2A_2)...\exp(t_mA_m).
\]

Apply this to all terms (index > 1), we have:
\[
\exp(t_1A_1 + t_2A_2 + ... + t_mA_m)
= \exp(t_1A_1)\exp(t_2A_2)...\exp(t_mA_m)
\]
\[
= \prod_{\text{perm}(i)} \exp(t_iA_i).
\]

2. Proof of Proposition 2

Proposition 2. If $A_iA_j = 0, \forall i, j$, then
\[
H_{ij} = \frac{\partial^2 g(t)}{\partial t_i\partial t_j} = 0,
\]
where $g(t) = \exp(t_1A_1 + t_2A_2 + ... + t_mA_m)$.

Proof. By (Rossmann, 2002) Theorem 5, we have:
\[
\frac{d}{dt}\exp(X) = \exp(X)\frac{1 - \exp(-\text{ad}_X) dt}{\text{ad}_X},
\]
where $X = X(t)$ is any matrix-valued differentiable function of a scalar variable $t$, and
\[
\frac{1 - \exp(-\text{ad}_X)}{\text{ad}_X} = \sum_{k=0}^{\infty} (-1)^k/k!(\text{ad}_X)^k,
\]
and $\text{ad}_XY = [X, Y] = XY - YX$ is the adjoint action on Lie algebra (note that $(\text{ad}_X)^kY = [X, [X, ..., [X, Y]]])$. By denoting $A = t_1A_1 + t_2A_2 + ... + t_mA_m$, we then have:
\[
\frac{\partial g(t)}{\partial t_i} = g(t)(-1)^{i-1}((\text{ad}_A)^0 + (-1)^{i-1}(\text{ad}_A)^1 + ... + (-1)^{2j-1}(\text{ad}_A)^2 + ...)]\frac{\partial(t_1A_1 + ... + t_mA_m)}{\partial t_i}
= g(t)[A_i + (-\frac{1}{2})[A, A_i] + \frac{1}{6}[A, [A, A_i]] + ...].
\]
Since $A_iA_j = 0, \forall i, j$, we have:
\[
[Y, A_i] = t_1[A_1, A_i] + ... + t_mA_mA_i
= t_1(A_1A_i - A_iA_1) + ... + t_m(A_mA_i - A_iA_m)
= 0,
\]
so
\[
\frac{\partial g(t)}{\partial t_i} = g(t)A_i,
\]
\[
H_{ij} = \frac{\partial^2 g(t)}{\partial t_i\partial t_j} = \frac{\partial}{\partial t_j} \left( \frac{\partial g(t)}{\partial t_i} \right)
= \frac{\partial g(t)A_i}{\partial t_j} = g(t)A_jA_i = 0.
\]

3. Introduction of Variational Autoencoder

The variational autoencoder (VAE) is a latent variable model to maximize the evidence lower bound (ELBO) of the intractable marginal log-likelihood of the training data:
\[
\log p(x) \geq \mathcal{L}(x, z) = \mathbb{E}_{q(z|x)} \log p(x|z) - KL(q(z|x)||p(z)),
\]
where $p(x)$ is the data distribution and $q(z|x)$ is the posterior distribution of the latent variable $z$, and $KL$ is the Kullback-Leibler divergence.
where an inference network \( q(z|x) \) is introduced to estimate the posterior distribution \( p(z|x) \). The \( q(z|x) \) and \( p(x|z) \) are constructed as two networks combined to form an stochastic autoencoder, where the \( q(z|x) \) is usually modeled as a Gaussian distribution with means and standard deviations being the outputs of a neural network. The prior distribution \( p(z) \) is usually fixed to be a standard Gaussian distribution. The first term in Eq. 24 is modeled as a reconstruction loss.

4. Proof of Proposition 3

**Proposition 3.** Suppose two latent variables \( z \) and \( t \) are used to model the log-likelihood of data \( x \), then we have:

\[
\log p(x) \geq \mathcal{L}_{\text{bottleneck}}(x, z, t) = E_{q(z|x)} E_{q(t|x, z)} \log p(x, z, t) - E_{q(z|x)}KL(q(t|x, z)||p(t)) - E_{q(z|x)} \log q(z|x)
\]

where Eq. 26 holds because we assume Markov property: \( q(t|z) = q(t|x, z), p(x|z, t) = p(x|z) \).

**Proof.**

\[
\log p(x) = \log \int_z \int_t p(x, z, t)
\]

\[
= \log \int_z \int_t p(x, z, t) \frac{q(t|x, z)q(z|x)}{q(t|x, z)q(z|x)}
\]

\[
\geq \int_z q(z|x) \log \int_t p(x, z, t) \frac{q(t|x, z)}{q(t|x, z)q(z|x)}
\]

\[
= \int_z q(z|x) \log \int_t p(x, z, t) \frac{q(t|x, z)}{q(t|x, z)q(z|x)} - E_{q(z|x)} \log q(z|x)
\]

**Table 1.** Encoder architecture on Dsprites, 3DSHAPEs, CelebA and 3DCAMs datasets.

<table>
<thead>
<tr>
<th>Encoder-64</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 × 64 × n_channel</td>
</tr>
<tr>
<td>4 × 4 Conv. 32, ReLU, Stride 2</td>
</tr>
<tr>
<td>4 × 4 Conv. 32, ReLU, Stride 2</td>
</tr>
<tr>
<td>4 × 4 Conv. 64, ReLU, Stride 2</td>
</tr>
<tr>
<td>4 × 4 Conv. 64, ReLU, Stride 2</td>
</tr>
<tr>
<td>Flatten + FC. 256, ReLU</td>
</tr>
<tr>
<td>FC. group_size</td>
</tr>
<tr>
<td>FC. group_size × 4, ReLU</td>
</tr>
<tr>
<td>FC. 2 × n latent</td>
</tr>
</tbody>
</table>

**Table 2.** Decoder architecture on Dsprites and 3DSHAPEs, CelebA and 3DCAMs datasets.

<table>
<thead>
<tr>
<th>Decoder-64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latent code ( \in \mathbb{R}^{d_{\text{latent}}} )</td>
</tr>
<tr>
<td>Multiply Lie Algebra Basis + Matrix Exponential</td>
</tr>
<tr>
<td>Flatten + FC. 256, ReLU</td>
</tr>
<tr>
<td>FC. 4 × 4 × 64, ReLU</td>
</tr>
<tr>
<td>4 × 4 Deconv. 64, ReLU, Stride 2</td>
</tr>
<tr>
<td>4 × 4 Deconv. 32, ReLU, Stride 2</td>
</tr>
<tr>
<td>4 × 4 Deconv. 32, ReLU, Stride 2</td>
</tr>
<tr>
<td>4 × 4 Deconv. n_channel, Sigmoid, Stride 2</td>
</tr>
</tbody>
</table>

\[
= E_{q(z|x)} E_{q(t|x, z)} \log p(x, z|t)
\]

where Eq. 29 and 31 are due to Jensen’s inequality, and Eq. 35 holds because we assume Markov property: \( q(t|z) = q(t|x, z) \).

5. Implementation Details

The Lie Group VAE Encoder and Decoder architectures used on all datasets are shown in Table 1, 2, 3, and 4. These architectures all inherit from the architectures proposed in FactorVAE (Kim & Mnih, 2018) with small modifications to support the integration of Lie Group constraints proposed in this paper. Specifically, two FC layers are added in the encoder to obtain the inferred group representation and latent code, while Lie Algebra multiplication and matrix exponential mapping are added in the decoder. The convolution layers are kept the same as in (Kim & Mnih, 2018). The \( loss_{\text{rec-group}} \) is scaled by 0.1 on all datasets. On 3DSHAPEs and CelebA the group size is set to 400, while on other datasets we use 100.
Appendix of Commutative Lie Group VAE for Disentanglement Learning

<table>
<thead>
<tr>
<th>Encoder-32</th>
</tr>
</thead>
<tbody>
<tr>
<td>$32 \times 32 \times n_{\text{channel}}$</td>
</tr>
<tr>
<td>$4 \times 4$ Conv. 32, ReLU, Stride 2</td>
</tr>
<tr>
<td>$4 \times 4$ Conv. 32, ReLU, Stride 2</td>
</tr>
<tr>
<td>$4 \times 4$ Conv. 64, ReLU, Stride 2</td>
</tr>
<tr>
<td>Flatten + FC. 256, ReLU</td>
</tr>
<tr>
<td>FC. $\text{group_size}$</td>
</tr>
<tr>
<td>FC. $\text{group_size} \times 4$, ReLU</td>
</tr>
<tr>
<td>FC. $2 \times n_{\text{latent}}$</td>
</tr>
</tbody>
</table>

**Table 3.** Encoder architecture on Mnist dataset.

<table>
<thead>
<tr>
<th>Decoder-32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latent code $\in \mathbb{R}^{n_{\text{latent}}}$</td>
</tr>
<tr>
<td>Multiply Lie Algebra Basis + Matrix Exponential</td>
</tr>
<tr>
<td>Flatten + FC. 256, ReLU</td>
</tr>
<tr>
<td>FC. $4 \times 4 \times 64$, ReLU</td>
</tr>
<tr>
<td>$4 \times 4$ Deconv. 32, ReLU, Stride 2</td>
</tr>
<tr>
<td>$4 \times 4$ Deconv. 32, ReLU, Stride 2</td>
</tr>
<tr>
<td>$4 \times 4$ Deconv. $n_{\text{channel}}$, Sigmoid, Stride 2</td>
</tr>
</tbody>
</table>

**Table 4.** Decoder architecture on Mnist dataset.

**References**

