Disentanglement Challenge: From Regularization to Reconstruction

Jie Qiao*  
Zijian Li *  
Ruichu Cai  
School of Computers, Guangdong University of Technology, China

Abstract

The challenge of learning disentangled representation has recently attracted much attention and boils down into a competition. Various methods based on variational auto-encoder have been proposed to solve this problem, by enforcing the independence between the representation and modify the regularization term in the variational lower bound. However recent work by Locatello et al. (2018) has demonstrated that the proposed methods are heavily influenced by randomness and the choice of the hyper-parameter. In this work, instead of designing a new regularization term, we found that the performance of the reconstruction, which is rarely explored, is also essential for learning a good disentangled representation. Therefore, in this competition, by increasing the capacity of the encoder and decoder network and the training step size, we achieve the first place in the leaderboard.

Keywords: Disentangled representation, unsupervised learning

1. Introduction

The great success of unsupervised learning heavily depends on the representation of the feature in real-world. It is widely believed that the real-world data is generated by a few explanatory factors which are distributed, invariant and disentangled (Bengio et al., 2013). The challenge of learning disentangled representation boils down into a competition to build the best disentangled model.

The key idea in disentangled representation is that the perfect representation should be a one to one mapping to the ground truth disentangled factor. Thus, if one factor changed and other factors fixed, then the representation of the fixed factor should be fixed accordingly while others representation changed. As a result, it is essential to find a representation that (i) are independent of each other, and (ii) align to the ground truth factor.

Recent line of works in disentanglement representation learning are commonly focusing on enforcing the independence of the representation by modifying the regulation term in the variational lower bound of Variational Autoencoders (VAE) (Kingma and Welling, 2013), methods that include \( \beta \)-VAE (Higgins et al., 2017), AnnealedVAE (Burgess et al., 2018), \( \beta \)-TCVAE (Chen et al., 2018), DIP-VAE (Kumar et al., 2018) and FactorVAE (Kim and Mnih, 2018). See Appendix A for more details of these model.

* equal contribution

To evaluate the disentangled, several metrics have been proposed, including the Factor-VAE metric (Kim and Mnih, 2018), Mutual Information Gap (MIG) (Chen et al., 2018), DCI metric (Eastwood and Williams, 2018), IRS metric (Suter et al., 2019), SAP score (Kumar et al., 2018).

However, one of our findings is that these methods are heavily influenced by randomness and the choice of the hyper-parameter. Such phenomenon is also discovered by Locatello et al. (2018). As illustrate in Fig. 1, the metrics have high variance across all our submissions with different hyper-parameters and model. Therefore, rather than to design a new regularization term, we try to enforce the performance of the reconstruction. We believe that, the better the performance, the better the alignment of the ground-truth factors.

We found that the alignment of the representation is heavily depend on the performance of the reconstruction. Therefore, the more capacity of the encoder and decoder network, the better result would be. Furthermore, after increase the capacity, we also try to increase the training step which also shown a significant improvement of evaluation metrics.

Overall, our contribution can be summarized as follow: (1) we found that the performance of the reconstruction is also essential for learning disentangled representation, and (2) we achieve the state-of-the-art performance in the competition.

2. Experiments Design

In this section, we introduce the experiment design and aim to explore the effectiveness of the disentangle learning model and the performance of the reconstruction for disentangle learning. Therefore, we first employ different kinds of variational autoencoder including BottleneckVAE, AnnealedVAE, DIPVAE, BetaTCVAE, BetaVAE with 30000 step size, and then choose the best performance VAE variation. Sequentially, we want to know whether the capacity play an important role in disentanglement. Since the larger the capacity, the better reconstruction can be obtained, which further reinforcement the disentanglement. We devise different capacity with different parameters. In detail, we control the number of latent variables and the size of filters in convolutional neural network (CNN).

After augmenting the capacity of the neural network, we should also augment the training step to adaptive the large capacity. Therefore, we further employ different training step to validate our assumption.

3. Experiments Results

In this section, we will introduce our experiment result. Firstly, we will present the performance of different kinds of VAEs, which is shown in Table 1.

According to the experiment result, we find that Factor VAE achieve the best result when training step is 30000. Considering the limited number of submission, in the following experiment, we choose Factor VAE as the base model. Therefore, in the second experiment,
Table 1: Variational variational autoencoder with 30000 training steps.

<table>
<thead>
<tr>
<th>VAE variation</th>
<th>FactorVAE</th>
<th>sap</th>
<th>score</th>
<th>dci</th>
<th>irs</th>
<th>mig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottleneck VAE</td>
<td>0.453</td>
<td>0.0395</td>
<td>0.107</td>
<td>0.547</td>
<td>0.0589</td>
<td></td>
</tr>
<tr>
<td>Anneled VAE</td>
<td>0.3586</td>
<td>0.0069</td>
<td>0.1153</td>
<td>0.5122</td>
<td>0.0237</td>
<td></td>
</tr>
<tr>
<td>DIPVAE</td>
<td>0.265</td>
<td>0.005</td>
<td>0.021</td>
<td>0.265</td>
<td>0.490</td>
<td></td>
</tr>
<tr>
<td>BetaTCVAE</td>
<td>0.342</td>
<td>0.026</td>
<td>0.093</td>
<td>0.342</td>
<td>0.981</td>
<td></td>
</tr>
<tr>
<td>Beta VAE</td>
<td>0.3586</td>
<td>0.0069</td>
<td>0.1153</td>
<td>0.5122</td>
<td>0.0237</td>
<td></td>
</tr>
<tr>
<td>Factor VAE</td>
<td><strong>0.449</strong></td>
<td><strong>0.0596</strong></td>
<td><strong>0.1385</strong></td>
<td><strong>0.5976</strong></td>
<td><strong>0.0589</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: FactorVAE with different latent variables size.

<table>
<thead>
<tr>
<th>VAE variation (latent variables, layer parameters types)</th>
<th>FactorVAE</th>
<th>sap</th>
<th>score</th>
<th>dci</th>
<th>irs</th>
<th>mig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor VAE (10, A)</td>
<td>0.449</td>
<td>0.0596</td>
<td>0.1385</td>
<td>0.5976</td>
<td>0.0589</td>
<td></td>
</tr>
<tr>
<td>Factor VAE (128, A)</td>
<td>0.3024</td>
<td>0.0051</td>
<td>0.2661</td>
<td>0.6045</td>
<td>0.0739</td>
<td></td>
</tr>
<tr>
<td>Factor VAE (128, B)</td>
<td>0.359</td>
<td>0.0873</td>
<td>0.4709</td>
<td>0.6946</td>
<td>0.1619</td>
<td></td>
</tr>
<tr>
<td>Factor VAE (256, A)</td>
<td>0.4282</td>
<td>0.0596</td>
<td>0.4126</td>
<td>0.6939</td>
<td>0.1796</td>
<td></td>
</tr>
<tr>
<td>Factor VAE (256, B)</td>
<td>0.4988</td>
<td>0.1371</td>
<td>0.5349</td>
<td>0.5435</td>
<td>0.3438</td>
<td></td>
</tr>
</tbody>
</table>

to verify the effectiveness of capacity, we first change the latent variables size and the layer size, i.e., the filters of CNN layers and the neurons of full-connection layers while keep the other hyper-parameters fixed. The changed parameters is given in Table 5.

As the experiment result shown in Table 2, we find that the size of latent variables is propitious to the disentanglement performance. The experiment in this part may be not so sufficiency, but it’s still illustrate the fact that larger the capacity is, better the disentanglement performance become. Since we largen the capacity of the model, it’s reasonable to largen the training step at the same time. This is because the model with more parameters should be trained with more steps.

Table 3: Factor VAE with different training step

<table>
<thead>
<tr>
<th>VAE variation (training step)</th>
<th>FactorVAE</th>
<th>sap</th>
<th>score</th>
<th>dci</th>
<th>irs</th>
<th>mig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor VAE (30k)</td>
<td>0.449</td>
<td>0.0596</td>
<td>0.1385</td>
<td>0.5976</td>
<td>0.0589</td>
<td></td>
</tr>
<tr>
<td>Factor VAE (500k)</td>
<td>0.432</td>
<td>0.0743</td>
<td>0.4395</td>
<td>0.6041</td>
<td>0.0739</td>
<td></td>
</tr>
<tr>
<td>Factor VAE (1000k)</td>
<td>0.4282</td>
<td>0.0596</td>
<td>0.4126</td>
<td>0.6939</td>
<td>0.1796</td>
<td></td>
</tr>
<tr>
<td>Factor VAE (1200k)</td>
<td><strong>0.4888</strong></td>
<td><strong>0.156</strong></td>
<td><strong>0.534</strong></td>
<td><strong>0.6744</strong></td>
<td><strong>0.3805</strong></td>
<td></td>
</tr>
</tbody>
</table>

As shown in Table 3, after we choose the Factor VAE as the base model with the type B layer parameters and 1200k training steps, the best result was achieved.

4. Conclusion

In this work, we performed an empirical study on disentangled learning. We first shows the randomness of the disentangle learning in various methods, and second by consider to improve the performance of the reconstruction, we increase the capacity of the model, and third we further increase the step size, and finally, we achieve a competitive result.
References


Appendix A. Related works

In this section, we are going to summarize the state-of-the-art unsupervised disentanglement learning methods. Most of works are developed based on the Variational Auto-encoder (VAE) (Kingma and Welling, 2013), a generative model that maximize the following evidence lower bound to approximate the intractable distribution $p_\theta(x|z)$ using $q_\phi(z|x)$,

$$
\max_{\phi, \theta} \mathbb{E}_{p(x)} \left[ \mathbb{E}_{q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] \right] - D_{\text{KL}}(q_\phi(z|x) \| p(z)),
$$

(1)
where $q_\phi(z|x)$ denote Encoder with parameter $\phi$ and $p_\theta(x|z)$ denote Decoder with parameter $\theta$.

As shown in Table 4, all the lower bound of variant VAEs can be described as Reconstruction Loss + Regularization where all the Regularization term and the hyper-parameters are given in this table.

Table 4: Summary of variant unsupervised disentanglement learning methods

<table>
<thead>
<tr>
<th>Model</th>
<th>Regularization</th>
<th>Hyper-Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$-VAE</td>
<td>$\beta \mathbb{D}<em>{KL}(q</em>\phi(z</td>
<td>x)</td>
</tr>
<tr>
<td>AnnealedVAE</td>
<td>$\gamma \mathbb{D}<em>{KL}(q</em>\phi(z</td>
<td>x)</td>
</tr>
<tr>
<td>FactorVAE</td>
<td>$D_{KL} (q_\phi(z</td>
<td>x)</td>
</tr>
<tr>
<td>DIP-VAE-I</td>
<td>$D_{KL} (q_\phi(z</td>
<td>x)</td>
</tr>
<tr>
<td>DIP-VAE-II</td>
<td>$D_{KL} (q_\phi(z</td>
<td>x)</td>
</tr>
</tbody>
</table>

Appendix B. Experiments

Table 5: Two types of layer parameters

<table>
<thead>
<tr>
<th>layer name</th>
<th>Type A</th>
<th>Type B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv_encoder_1,2</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>Conv_encoder_3,4</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Conv_decoder_1,2</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>Conv_decoder_3,4</td>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>

Appendix C. Model Architecture
e1 = tf.layers.conv2d(
    inputs=input_tensor,
    filters=64,
    kernel_size=5,
    strides=2,
    activation=tf.nn.leaky_relu,
    padding="same",
    name="e1",
)

e1 = tf.contrib.layers.dropout(e1, is_training=is_training)

e2 = tf.layers.conv2d(
    inputs=e1,
    filters=64,
    kernel_size=5,
    strides=2,
    activation=tf.nn.leaky_relu,
    padding="same",
    name="e2",
)

e2 = tf.contrib.layers.dropout(e2, is_training=is_training)

e3 = tf.layers.conv2d(
    inputs=e2,
    filters=64,
    kernel_size=3,
    strides=2,
    activation=tf.nn.leaky_relu,
    padding="same",
    name="e3",
)

e3 = tf.contrib.layers.dropout(e3, is_training=is_training)

e4 = tf.layers.conv2d(
    inputs=e3,
    filters=64,
    kernel_size=3,
    strides=2,
    activation=tf.nn.leaky_relu,
    padding="same",
    name="e4",
)

flat_e4 = tf.layers.flatten(e4)
e5 = tf.layers.dense(flat_e4, 256, activation=tf.nn.leaky_relu, name="e5")
means = tf.layers.dense(e5, num_latent, activation=None, name="means")
log_var = tf.layers.dense(e5, num_latent, activation=None, name="log_var")
return means, log_var

Figure 2: This is the code of the encoder, which contains the detail parameters.
Figure 3: This is the code of the decoder, which contains the detail parameters.