A PyTorch-Based Approach for Variational Learning with Disentanglement

Abstract

This paper presents the Disentanglement-PyTorch library, which has been developed to assist in the research, application, and assessment of novel variational algorithms. This modular library allows for independent and reliable experimentation across diverse variational methodologies, through the decoupling of neural architectures, the dimensionality of the latent space, and training algorithms. Furthermore, the library manages training schedules, logging, and the visualization of reconstructions and traversals in the latent space. It also provides evaluation of the encodings using various disentanglement metrics. Currently, the library includes implementations of the following unsupervised algorithms: VAE, β-VAE, Factor-VAE, DIP-I-VAE, DIP-II-VAE, Info-VAE, and β-TCVAE. Additionally, conditional approaches such as CVAE and IFCVAE are also supported. This library was utilized in some Disentanglement Challenge, where it achieved a 3rd rank in both the first and second phases of the competition.

1 Introduction

In the field of representation learning, two primary paths can be identified. One path concentrates on learning transformations that are specific to a given task, often optimized for particular domains and applications. The other path involves learning the inherent factors of variation, in a manner that is both disentangled and task-invariant. The task of unsupervised disentanglement of latent factors, where changes in a single factor shift the latent encoding in a single direction, represents an unresolved problem in representation learning. Disentangled representations offer significant advantages across various domains of machine learning including few-shot learning, reinforcement learning, transfer learning, and semi-supervised learning. This work introduces a library developed using the functionalities of the PyTorch framework. This library has been designed to facilitate the research, implementation, and evaluation of new variational algorithms, with a specific emphasis on representation learning and disentanglement. This library was created in conjunction with the Disentanglement Challenge of NeurIPS 2019. The Disentanglement-PyTorch library is publicly available under the GNU General Public License.

2 Library Features

2.1 Supported Algorithms and Objective Functions

2.1.1 Unsupervised Objectives

The library currently offers implementations of the following unsupervised variational algorithms: VAE, β-VAE, β-TCVAE, Factor-VAE, Info-VAE, DIP-I-VAE, and DIP-II-VAE. The algorithms are incorporated as plug-ins to the variational Bayesian framework. They are specified by their respective loss terms. Consequently, if the loss terms from two learning algorithms (e.g., A and B) are compatible, they can be integrated into the objective function by setting the appropriate flag. This allows researchers to combine loss terms that optimize for related objectives.

.

2.1.2 Conditional and Attribute-variant Objectives

The library provides support for conditional methods such as CVAE, where extra known attributes (i.e., labels) are utilized in both the encoding and decoding procedures. It also offers support for IFCVAE. This is a method that enforces certain latent factors to encode known attributes through a set of positive and negative discriminators in a supervised manner. The library's modular construction allows the use of any of the previously mentioned unsupervised loss terms in conjunction with conditional and information factorization techniques. This allows for the encouragement of disentanglement across attribute-invariant latents.

2.2 Neural Architectures

The neural architectures and the dimensionality of the data and latent spaces can be configured and are independent from the training algorithm. This design enables the independent investigation of new architectures for encoder and decoder networks, as well as support for diverse data domains.

2.3 Evaluation of Disentanglement

To evaluate the quality of the learned representations, we use an existing implementation of disentanglement metrics. Thanks to an external library, the following metrics are supported: BetaVAE, FactorVAE, Mutual Information Gap (MIG), Interventional Robustness Score (IRS), Disentanglement Completeness and Informativeness (DCI), and Separated Attribute Predictability (SAP).

2.4 Miscellaneous Features

2.4.1 Controlled Capacity Increase

It has been demonstrated that gradually relaxing the information bottleneck during training improves disentanglement without compromising reconstruction accuracy. The capacity, which is defined as the distance between the prior and the latent posterior distributions and represented with the variable C, is incrementally increased throughout training.

2.4.2 Reconstruction Weight Scheduler

To prevent convergence at points with high reconstruction loss, training can be initialized with a greater focus on reconstruction. The emphasis can be progressively shifted toward the disentanglement term as training proceeds.

2.4.3 Dynamic Learning Rate Scheduling

The library supports all types of learning rate schedulers. Researchers are encouraged to use the dynamic learning rate scheduling to reduce the rate gradually. This should be done when the average objective function over the epoch ceases its decreasing trend.

2.4.4 Logging and Visualization

The library utilizes a tool to log the training process and visualizations. It allows the visualization of condition traversals, latent factor traversals, and output reconstructions in both static images and animated GIFs.

3 Experiments and Results

The β-TCVAE algorithm yielded the most effective disentanglement outcomes on the mpi3d real dataset during the second phase of the disentanglement challenge. Given the limited 8-hour timeframe allocated for training, the model was pre-trained on the mpi3d toy dataset. The model was trained using the Adam optimizer for a total of 90,000 iterations, with a batch size of 64. The β value for the β-TCVAE objective function was set at 2. The learning rate was initially set to 0.001. It was reduced by a factor of 0.95 when the objective function reached a plateau. The capacity parameter, C, was increased gradually from 0 to 25. The dimensionality of the z-space was set to 20.

The encoder comprised 5 convolutional layers. The number of kernels increased gradually from 32 to 256. The encoder concluded with a dense linear layer. This layer was used to estimate the posterior latent distribution as a parametric Gaussian. The decoder network included one convolutional layer. This was followed by 6 deconvolutional (transposed convolutional) layers. The number of kernels gradually decreased from 256 down to the number of channels in the image space. ReLU activations were used for all layers, except for the final layers of both the encoder and decoder networks.

The performance of the model on unseen objects from the mpi3d realistic and mpi3d real datasets is shown in Table 1. The model consistently performed better on the mpi3d realistic and mpi3d real datasets. This is despite the fact that the model was only pre-trained using the mpi3d toy dataset.

Table 1: Results of the best configurations of β-TCVAE on DCI, FactorVAE, SAP, MIG, and IRS metrics.

Method	Dataset	DCI	FactorVAE	SAP	MIG	IRS
β-TCVAE	mpi3d realistic	0.3989	0.3614	0.1443	0.2067	0.6315
β-TCVAE	mpi3d real	0.4044	0.5226	0.1592	0.2367	0.6423

4 Conclusion

The Disentanglement-PyTorch library offers a modular platform for studying, implementing, and assessing algorithms for disentanglement learning. It incorporates implementations of several well-known algorithms, along with a variety of evaluation metrics. This makes it a valuable resource for the research community.

Appendix A. Latent Factor Traversal

[width=0.8]latent$_traversal_figure$

Figure 1: Latent factor traversal of the trained β-TCVAE model on a random sample of the mpi3d realistic dataset. The disentanglement is not complete as some features are encoded in the same latent factor. A latent space of size 20 was used, however, changes in the other 13 latent factors had no effect on the reconstruction; thus, these feature-invariant factors were not included for brevity.