Improved Disentanglement through Learned Aggregation of Convolutional Feature Maps

Maximilian Seitzer CONTACT@MAX-SEITZER.DE
Andreas Foltyn ANDREAS.FOLTYN@IIS.FRAUNHOFER.DE
Felix P. Kemeth FELIX.KEMETH@IIS.FRAUNHOFER.DE
Fraunhofer Institute for Integrated Circuits IIS, Erlangen, Germany

Abstract

We present and discuss a simple image preprocessing method for learning disentangled latent factors. In particular, we utilize the implicit inductive bias contained in features from networks pretrained on the ImageNet database. We enhance this bias by explicitly fine-tuning such pretrained networks on tasks useful for the NeurIPS2019 disentanglement challenge, such as angle and position estimation or color classification. Furthermore, we train a VAE on regionally aggregate feature maps, and discuss its disentanglement performance using metrics proposed in recent literature.

1. Introduction

Fully unsupervised methods, that is, without any human supervision, are doomed to fail for tasks such as learning disentangled representations (Locatello et al., 2018). In this contribution, we utilize the implicit inductive bias contained in models pretrained on the ImageNet database (Russakovsky et al., 2014), and enhance it by finetuning such models on challenge-relevant tasks such as angle and position estimation or color classification. In particular, our submission for challenge stage 2 builds on our submission from stage 1[1], in which we employed pretrained CNNs to extract convolutional feature maps as a preprocessing step before training a VAE (Kingma and Welling, 2013). Although this approach already results in partial disentanglement, we identified two issues with the feature vectors extracted this way. Firstly, the feature extraction network is trained on ImageNet, which is rather dissimilar to the *MPI3d* dataset used in the challenge. Secondly, the feature aggregation mechanism was chosen ad-hoc and likely does not retain all information needed for disentanglement. We attempt to fix these issues by finetuning the feature extraction network as well as learning the aggregation of feature maps from data by using the labels of the simulation datasets *MPI3d-toy* and *MPI3d-realistic*.

2. Method

Our method consists of the following three steps: (1) supervised finetuning of the feature extraction CNN (section 2.1), (2) extracting a feature vector from each image in the dataset using the finetuned network (section 2.2), (3) training a VAE to reconstruct the feature vectors and disentangle the latent factors of variation (section 2.3).

1. Accessible at https://openreview.net/forum?id=ryxOvH86SH

2.1. Finetuning the Feature Extraction Network

In this step, we finetune the feature extraction network offline (before submission to the evaluation server). The goal is to adapt the network such that it produces aggregated feature vectors that capture the latent variables well. In particular, the network is finetuned by learning to predict the value of each latent factor from the aggregated feature vector of an image. To this end, we use the simulation datasets *MPI3d-toy* and *MPI3d-realistic*[2], namely the images as inputs and the labels as supervised classification targets.

For the feature extraction network, we use the VGG19-BN architecture (Simonyan and Zisserman, 2014) of the `torchvision` package. The input images are standardized using mean and variance across each channel computed from the ImageNet dataset. We use the output feature maps of the last layer before the final average pooling (dimensionality $512 \times 2 \times 2$) as the input to a feature aggregation module which reduces the feature map to a 512-dimensional vector[3]. This aggregation module consists of three convolution layers with $1024, 2048, 512$ feature maps and kernel sizes $1, 2, 1$ respectively. Each layer is followed by batch normalization and ReLU activation. We also employ layerwise dropout with rate 0.1 before each convolution layer. Finally, the aggregated feature vector is $\ell 2$-normalized, which was empirically found to be important for the resulting disentanglement performance. Then, for each latent factor, we add a linear classification layer computing the logits of each class from the aggregated feature vector. These linear layers are discarded after this step.

We use both *MPI3d-toy* and *MPI3d-realistic* for training to push the network to learn features that identify the latent factors in a robust way, regardless of details such as reflections or specific textures. In particular, we use a random split of 80% of each dataset as the training set, and the remaining samples as a validation set. VGG19-BN is initialized with a set of weights resulting from ImageNet training[4], and the aggregation module and linear layers were randomly initialized using uniform He initialization (He et al., 2015). The network is trained for 5 epochs using the RAdam optimizer (Liu et al., 2019) with learning rate 0.001, $\beta_0 = 0.999$, $\beta_1 = 0.9$, a batch size of 512 and a weight decay of 0.01. We use a multi-task classification loss consisting of the sum of cross entropies between the prediction and the ground truth of each latent factor. After training, the classification accuracy on the validation set is around 98% for the two degrees of freedom of the robot arm, and around 99.9% for the remaining latent factors.

2.2. Feature Map Extraction and Aggregation

In this step, we use the finetuned feature extraction network to produce a set of aggregated feature vectors. We simply run the network detailed in the previous step on each image of the dataset and store the aggregated 512-dimensional vectors in memory. Again, inputs to the feature extractor are standardized such that mean and variance across each channel correspond to the respective ones from the ImageNet dataset.

2. Pretraining using *any* data was explicitly stated to be allowed by the challenge organizers.
3. Using aggregated feature vectors instead of feature maps is necessitated by the memory requirements of the challenge.
4. https://download.pytorch.org/models/vgg19_bn-c79401a0.pth

2.3. VAE Training

Finally, we train a standard β-VAE (Higgins et al., 2017) on the set of aggregated feature vectors resulting from the previous step. The encoder network consists of 4 fully-connected layers with 1024 neurons each, followed by two fully-connected layers parametrizing mean and log variance of the factorized Gaussian approximate posterior $q(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ with $C = 16$ latent factors. The number of latent factors was experimentally determined. The decoder network consists of 4 fully-connected layers with 1024 neurons each, followed by a fully-connected layer parametrizing the mean of the factorized Gaussian conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{N}(\hat{\boldsymbol{\mu}}, \boldsymbol{I})$. The mean is constrained to range $(0, 1)$ using the sigmoid activation. All fully-connected layers but the final ones use batch normalization and are followed by ReLU activation functions. We use orthogonal initialization Saxe et al. (2013) for all layers and assume a factorized Gaussian prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ on the latent variables.

For optimization, we use the RAdam optimizer (Liu et al., 2019) with a learning rate of 0.001, $\beta_0 = 0.999$, $\beta_1 = 0.9$ and a batch size of $B = 256$. The VAE is trained for $N = 100$ epochs by minimizing

$$\frac{1}{B} \sum_{i=1}^{512} (\hat{\mu}_i - x_i)^2 - 0.5 \frac{\beta}{BC} \sum_{j=1}^{C} 1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2$$

where β is a hyperparameter to balance the losses of the MSE reconstruction and the KLD penalty terms. As the scale of the KLD term depends on the numbers of latent factors C, we normalize it by C such that β can be varied independently of C. It can be harmful to start training with too much weight on the KLD term (Bowman et al., 2015). Therefore, we use the following cosine schedule to smoothly anneal β from $\beta_{\text{start}} = 0.001$ to $\beta_{\text{end}} = 0.4$ over the course of training:

$$\beta(t) = \begin{cases} \beta_{\text{start}} & \text{for } t < t_{\text{start}} \\ \beta_{\text{end}} - \frac{1}{2}(\beta_{\text{end}} - \beta_{\text{start}}) \left(1 + \cos \pi \frac{t - t_{\text{start}}}{t_{\text{end}} - t_{\text{start}}}\right) & \text{for } t_{\text{start}} \leqslant t \leqslant t_{\text{end}} \\ \beta_{\text{end}} & \text{for } t > t_{\text{end}} \end{cases}$$

where $\beta(t)$ is the value for β in training episode $t \in \{0, \ldots, N-1\}$, and annealing runs from epoch $t_{\text{start}} = 10$ to epoch $t_{\text{end}} = 49$. This schedule lets the model initially learn to reconstruct the data well and only then puts pressure on the latent variables to be factorized which we found to considerably improve performance.

3. Discussion

On the public leaderboard (i.e. on *MPI3D-real*), our best submission achieves the first rank on the FactorVAE (Kim and Mnih, 2018), and DCI (Eastwood and Williams, 2018) metrics, with a large gap to the second-placed entry. See appendix A for a discussion of the results.

Unsurprisingly, introducing prior knowledge simplifies the disentanglement task considerably, reflected in improved scores. To do so, our approach makes use of task-specific supervision obtained from simulation, which restricts its applicability. Nevertheless, it constitutes a demonstration that this type of supervision can transfer to better disentanglement on real world data, which was one of the goals of the challenge.

References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. In *CoNLL*, 2015.

Tian Qi Chen, Xuechen Li, Roger Baker Grosse, and David Kristjanson Duvenaud. Isolating Sources of Disentanglement in Variational Autoencoders. In *ICLR*, 2018.

Cian Eastwood and Christopher K. I. Williams. A Framework for the Quantitative Evaluation of Disentangled Representations. In *ICLR*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *ICLR*, 2017.

Hyunjik Kim and Andriy Mnih. Disentangling by Factorising. In *ICML*, 2018.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *ArXiv*, abs/1312.6114, 2013.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational Inference of Disentangled Latent Concepts from Unlabeled Observations. *ArXiv*, abs/1711.00848, 2017.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the Variance of the Adaptive Learning Rate and Beyond. *ArXiv*, abs/1908.03265, 2019.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. In *RML@ICLR*, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115:211–252, 2014.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *ArXiv*, abs/1312.6120, 2013.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR*, 2014.

Raphael Suter, ore Miladinovic, Bernhard Schölkopf, and Stefan Bauer. Robustly Disentangled Causal Mechanisms: Validating Deep Representations for Interventional Robustness. In *ICML*, 2019.

Table 1: Summary of scores and ranks of our best submission on the public leaderboard at the end of stage 2. For comparison, we also include the private leaderboard scores of our phase 1 submission. The normalized score is provided to show the performance relative to the best submission and is calculated by dividing the score by the highest achieved score on each metric.

	FactorVAE	**DCI**	**SAP**	**IRS**	**MIG**	
Score	0.992	0.809	0.223	0.547	0.297	$\sum$ 2.868
Normalized Score	1	1	0.949	0.627	0.786	$\sum$ 4.362
Rank	1	1	2	11	3	$\varnothing$ 3.6
Score Phase 1	0.792	0.527	0.166	0.623	0.292	$\sum$ 2.400

Appendix A. Discussion of Results on the Public Leaderboard

We summarize the results of our best submission on the public leaderboard in table 1. Our method achieves the first rank on FactorVAE (Kim and Mnih, 2018) and DCI (Eastwood and Williams, 2018). For both metrics, there is a large absolute difference to the second ranked entry, namely 0.37 for FactorVAE and 0.26 for DCI. For SAP (Kumar et al., 2017), our method is almost tied with the first ranking entry, with 0.01 absolute difference. For MIG (Chen et al., 2018) and IRS (Suter et al., 2019), our method falls behind the best method, with an absolute distance of 0.08 and 0.13 respectively.

Compared to our phase 1 submission which does not use supervised finetuning, metrics for which our approach was already good (FactorVAE and DCI), became even better, while other metrics for which our approach performed subpar, stayed the same (MIG) or even became worse (IRS). It seems that adding supervised finetuning to our pretrained features approach enhances the already existing strengths and weaknesses. A more detailed investigation why each metric behaves the way it does for our method is needed.