RESCIENCE C

Replication / ML Reproducibility Challenge 2022

[Re] VAE Approximation Error: ELBO and Exponential Families

Volodymyr Kyrylov[1], Navdeep Singh Bedi[1], and Qianbo Zang[1]

[1]Università della Svizzera italiana, Lugano, Ticino, Switzerland

Edited by
Koustuv Sinha,
Maurits Bleeker,
Samarth Bhargav

Received
04 February 2023

Published
20 July 2023

DOI
10.5281/zenodo.8173745

Reproducibility Summary

Scope of Reproducibility — Exponential family variational autoencoders struggle with reconstruction when encoders output limited information.
We reproduce two experiments in which we first train the decoder and encoder separately. Then, we train both modules jointly using ELBO and observe the degradation of reconstruction.
We verify how the theoretical insight into the design of the approximate posterior and decoder distributions for a discrete VAE in a semantic hashing application influences the choice of input features to improve overall performance.

Methodology — We implement and train a synthetic experiment from scratch on a laptop. We use a mix of authors' code and publicly available code to reproduce a GAN reinterpreted as a VAE. We consult authors' code to reproduce semantic hashing experiment and make our own implementation. We train models the USI HPC cluster on machines with GTX 1080 Ti or A100 GPUs and 128 GiB of RAM. We spend under 100 GPU hours for all experiments.

Results — We observe expected qualitative behavior predicted by the theory on all experiments. We improve the best semantic hashing model's test performance by 5 nats by using a modern method for gradient estimation of discrete random variables.

What was easy — Following experiment recipes was easy once we worked out the theory.

What was difficult — The paper enables verification of exponential family distributions VAE designs of arbitrary complexity, which require probabilistic modeling skills. We contribute elaboration on the implementation details of the synthetic experiment and provide code.

Communication with original authors — We are extremely grateful to the authors for discussing the paper, encouraging us to implement experiments on our own, and suggesting directions for improving results over e-mail.

Copyright © 2023 V. Kyrylov, N.S. Bedi and Q. Zang, released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to Volodymyr Kyrylov (volodymyr.kyrylov@usi.ch)
The authors have declared that no competing interests exist.
Code is available at https://github.com/proger/vae – DOI https://doi.org/10.5281/zenodo.7951527. – SWH swh:1:dir:6328abd80f26a66924bea7e6b656ebfee9161f31.
Open peer review is available at https://openreview.net/forum?id=ozbAwipuZu.

1 Introduction

VAE [1] is a generative model of $p_\theta(x, z)$ where x is observed and z is a latent variable designed to generate novel samples of x given $z \sim p(z)$ using a *decoder* $p_\theta(x|z)$, jointly learned with an *encoder* $q_\phi(z|x)$ using an Evidence Lower Bound (ELBO) objective $\mathcal{L}$. The objective consists of the latent prior loss and reconstruction loss.
The gap between the true data likelihood and ELBO is the divergence between the approximate encoder and the true posterior:

$$\log p_\theta(x) - \mathcal{L}(\theta, \phi) = \mathbb{E}_{p_d}[D_{KL}(q_\phi(z|x)||p_\theta(z|x))] \tag{1}$$

Impressive reconstruction results with VAEs involve employing autoregression [2], normalizing flows [3], hierarchies of latent spaces [4] or explicitly balancing rate-distortion tradeoff [5] to make this gap low.

[6] focuses on studying q_ϕ and p_θ from the exponential family that includes continuous Gaussian and discrete Bernoulli distributions. The so-called *consistent* set of VAEs that can model the posterior exactly under this assumption is shown to be quite restricted: these models need to be *linear mappings of sufficient statistics* of observations and latents, regardless of how deep or complex encoder and decoder neural networks are.

When an encoder is factorized and cannot fully model the posterior, joint optimization using ELBO forces the decoder to adapt to the posterior at the expense of reconstruction quality. [6] enables better understanding of VAE designs.

2 Scope of reproducibility

We support the following claims with our experiments:

- setting up the encoder with conditionally independent bit outputs causes approximation errors in a perfect decoder, as shown in Section 3 with a synthetic experiment;
- same behavior is observed on image data: conditionally independent Gaussian encoder causes approximation errors in a decoder initialized by GAN pretraining, as shown in Section 4;
- word counts are better than word frequencies in a Bernoulli VAE in a semantic hashing task, as shown in Section 5.

3 Gaussian mixtures with factorized encoder

3.1 Methodology

We study the choice of encoder output distribution on a toy problem. [6] predicts achieving zero approximation error when the approximate posterior can exactly represent the true posterior. We induce a factorization in the posterior and expect approximation errors to arise. We implement this experiment from scratch using PyTorch.

Model description — Following [6], we define a ground truth decoder $p^*(x, z) = p^*(z)p^*(x|z)$ to be a mixture of four 2D Gaussians, where $x \in \mathbb{R}^2, z \in \{1, 2, 3, 4\}$, $p^*(z) = 0.25$, and $p^*(x|z) = \mathcal{N}(x|\mu_\theta(z), \sigma^2 I)$.

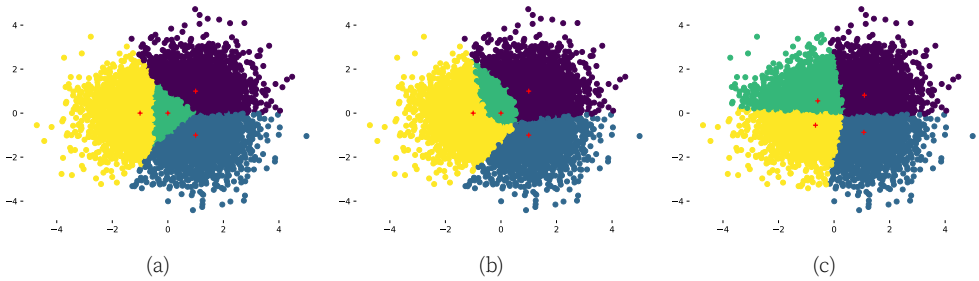

Figure 1. Cluster assignments according to the posterior distributions:
(a) ground truth $p(z|x)$ computed using Bayes rule
(b) approximate q_ϕ trained using ELBO with frozen p_θ
(c) approximate q_ϕ trained using ELBO jointly with p_θ

Our decoder parameters are a single linear layer $\theta : \mathbb{R}^4 \to \mathbb{R}^2$. Given a latent z represented using one hot encoding θ computes a mean parameter: $\mu_\theta(z) = \theta z$. Columns of θ are set to $(-1, 0), (0, 0), (1, 1), (1, -1)$. We guess these values by looking at cluster centers of Figure 2 in [6]. Computed mean and fixed identity covariance scaled by $\sigma = 0.01$ determine the normal distribution that is used to evaluate log probability of x.

Our encoder is defined as $q_\phi(z|x) \propto \exp\langle g_\phi(x), z_b\rangle$, where $\langle\cdot,\cdot\rangle$ is a dot product. We implement g_ϕ as a network with three randomly initialized linear layers with input dimension 2, hidden dimensions of 64, and output dimension of 2. Every layer except the output uses ReLU activation functions. We rely on default PyTorch initialization.

$z_b \in \{0, 1\}^2$ is a random variable that can be viewed as a string of two bits. To compute $q_\phi(z|x)$, we evaluate $g_\phi(x)$ once and compute inner products of its output with each possible value of $z_b = [0, 0], [0, 1], [1, 0], [1, 1]$ and apply the softmax function computes exp and normalizes, making the output a proper probability distribution. In this computation each bit of z_b is considered independently, making the approximate posterior distribution *factorized*. The factorization is chosen to demonstrate how preventing the encoder from exactly modeling the posterior causes approximation errors in the decoder.

Dataset — We make our training set x by sampling 2000 points from the ground truth decoder g_ϕ after setting the random seed to 0. To sample we choose a value of z and then choose a point from a univariate normal distribution given z.

Experimental setup — We train the encoder by minimizing negative ELBO keeping the decoder frozen for 10000 steps. We use Adam with a learning rate of 1e-3. Finally, we unfreeze the decoder and train both modules jointly for 10000 steps. We perform full batch training.

To calculate the reconstruction loss part of ELBO, we evaluate expectation

$$\mathbb{E}_{q_\phi} \log p_\theta(x|z) = \sum_i q_\phi(z_i|x) \log p_\theta(x|z = z_i)$$

exactly, running the decoder forward once for every possible z.

It takes under 10 minutes to run all experiments on a laptop CPU.

3.2 Results

After training the encoder alone, we achieve ELBO of -3.37. After training encoder and decoder jointly we observed a large improvement of ELBO to -3.31, however the clusters' geometric structure as shown in Figure 1 (c) is completely destroyed. In the original paper the ELBO value after first frozen decoder training is -3.74 and improves to -3.70. The relative behavior of the values is the same, however the overall likelihood is impacted by different choice of ground truth cluster center locations: perturbing their relative positions causes small changes in likelihoods. Changing σ significantly impacts the scale of the values.

The factorization assumption in the encoder prevents our VAE from being consistent.

Restoring consistency of this VAE — A consistent VAE [6] for exponential family distributions allows the encoder to model the posterior exactly. We get rid of the factorization to output all four sufficient statistics of z. We simplify the encoder network to have a single linear layer with input dimension 2 and output dimension 4 and use the softmax activation function in the output.

4 DCGAN turned Gaussian VAE on CelebA

4.1 Methodology

In this experiment, we train a ground truth decoder as a GAN [7]. Then, we pretrain the encoder using conditional likelihood. Next, we reinterpret GAN generator as the decoder and then train both components jointly using ELBO. We compare FID score [8] before and after joint training to detect approximation errors.
We use the code of [9] to implement DCGAN [10]. We use the authors' code for conditional likelihood training of the encoder and joint fine-tuning.

Model description — We designate a trained DCGAN $g_\theta : \mathbb{R}^{100} \to \mathbb{R}^{64\times64\times3}$ to be ground truth decoder, making it probabilistic by adding Gaussian noise: $z \sim \mathcal{N}(0 \in \mathbb{R}^{100}, I), x \sim \mathcal{N}(g_\theta(z), \sigma^2 I)$, $\sigma = 0.05$. We use an architecture similar to DCGAN as the encoder: transposed convolutions are replaced with convolutions, and ReLU activation is replaced with leaky ReLU with slope 0.2.

Dataset — We use an aligned and cropped version of the CelebA dataset. It contains 202,599 face images of 10,177 celebrity identities [11].

Experimental setup — To train a ground truth decoder as a DCGAN, we follow hyperparameters from [9], training with an epoch limit of 100 epochs. We select a checkpoint from one epoch before mode collapse, inspired by [12].

Next, we train the encoder: we sample z, generate x given z, use the encoder to predict parameters μ and σ given x and maximize $\mathcal{N}(z|\mu, \sigma)$ using Adam optimizer with a learning rate of 2e-4 for 1e6 iterations. After training we compute FID between 2e5 images generated from the ground truth decoder and re-decoded images after encoding them.

Next, we jointly update both modules with ELBO using Adam with a learning rate of 2e-4 for another 1e6 iterations. After joint training we compute FID between same ground truth and re-decoded images from new models.

Computational requirements — Pretraining of the decoder as a GAN takes about 5 hours. Training the encoder takes about 1 hour. Joint training takes about 3 hours. All experiments are performed on A100 40GiB GPUs.

4.2 Results

We report results in Table 1: we see ELBO improve after joint training. However this happens at the cost of introducing approximation errors: we observe significant degradation of FID.

Why does *ELBO go up, but FID goes down*? Optimization of ELBO harms a good image decoder to accommodate the parameters of approximate posterior. [6] argue that a Gaussian latent variable model is not expressive enough to generate realistic images. [6] additionally performs experiments where σ is a learned parameter which we choose to leave out for simplicity.

Table 1. CelebA results

Experiment	ELBO↑	FID↓
trained encoder, frozen decoder	-19072.746	1.311
joint training	5137.171	71.852

5 Semantic hashing on 20 Newsgroups

5.1 Methodology

We investigate a Bernoulli VAE for the semantic hashing task. In this task, we aim to learn an encoder that maps our document to binary codes comparable using Hamming distance. We compare the performance of term frequencies as features in [13] to term counts suggested in [6].
We first become acquainted with the experiment by running the code provided by the author. One experiment for one set of hyperparameters runs for about 30 minutes on one GPU. The whole set of experiments takes about 16 GPU hours in total. Next, we study the author's code by reimplementing it and improving results as described in Section 5.1.3.

Dataset — We use a preprocessed 20 Newsgroups dataset [14]. The vocabulary is restricted to 10000 most common words. Each document is represented as a vector of smoothed word counts. The training set contains 11269 documents, and a test set contains 7505 documents. We do not use text categories.

Model description — The encoder takes a vector of 10000 word counts as input and passes them through one linear layer with output dimension b, representing logits of b Bernoulli variables. The decoder takes b binary digits as input and passes them through d hidden layers with hidden dimension 512 and output dimension matching vocabulary size. The final activation of the decoder is log of softmax.

Gradient estimation for Bernoulli variables — Evaluation of ELBO involves estimating the decoder likelihood over the approximate posterior distribution. In discrete settings, implementations resort to Monte Carlo sampling. A commonly used method for gradient estimation of discrete variables is the REINFORCE algorithm [15]. This estimator is unbiased and has high variance. Reducing variance involves taking many samples, trading variance for bias [16], and sample augmentation.

Table 2. Train Negative ELBO (lower is better)

	d=0			d=1			d=2		
b	e1	e2	e3	e1	e2	e3	e1	e2	e3
8	**406**	419	432	**325**	382	407	**325**	363	415
16	**372**	383	418	**191**	298	397	**169**	287	406
32	**332**	343	409	**153**	181	391	**118**	170	405
64	**288**	304	405	**151**	174	391	**122**	165	401

Table 3. Test Negative ELBO (lower is better). All tests use DisARM gradient estimator. For the best overall configuration (no hidden decoder layers, word counts as features, no hidden encoder layers, 64 latent outputs) we repeat the column from the original paper for comparison. The column is labeled "ARM" for the used gradient estimator.

	d=0				d=1			d=2		
b	e1	ARM	e2	e3	e1	e2	e3	e1	e2	e3
8	**423**	**423**	444	439	**413**	453	427	**419**	431	431
16	**409**	**409**	451	430	**403**	466	431	**408**	448	441
32	**397**	**396**	439	433	**396**	454	455	**402**	466	473
64	**387**	392	460	467	**393**	486	481	**401**	484	478

ARM [17] re-parametrizes Bernoulli variables using a continuous Logistic distribution and evaluates them in antithetic pairs $(\epsilon, -\epsilon)$, benefiting from augmentation. DisARM [18] improves upon ARM by observing that Logistic reparametrization comes at the cost of increasing variance on its own and integrates continuous variables out using a conditioning technique. We implement DisARM method in our experiments.

Experimental Setup — We use a seed of 1, Adam with a learning rate of 1e-3 and a batch size of 256. We train the model for 1000 epochs estimating ELBO using an antithetic gradient estimator. We're additionally computing ELBO without augmentations to estimate training and test losses. Next, we train the model for 500 more epochs, averaging the loss over 10 backward passes for further gradient variance reduction.

We sweep over the following hyperparameters: varying latent dimension $b = 8, 16, 32, 64$, varying decoder hidden layer count $d = 0, 1, 2$. We use three following encoder designs. We refer to models with word counts as features as configuration `e1`. We convert word counts to word frequencies and add an extra hidden layer of dimension 512 to the encoder in configuration `e2`. We use word frequencies without extra hidden layers in configuration `e3`.

5.2 Results

Configurations with word counts as features consistently achieve lower negative ELBO, as shown in Tables 2 and 3.

We also note that best test results are achieved very early in training and low negative ELBO values in Table 2 correspond to final epochs: there is a significant gap in ELBO values between test and train sets. In accordance with the experiment protocol of [6], we keep track of all test ELBOs over time and choose the minimum to report, so it would be fair to consider this test set as development.

In our implementation, we use DisARM to estimate gradients and see that its variance reduction additionally improves the best configuration performance by 5 nats over re-

sults reported in [6], as shown in Table 3. We report only one column for brevity and refer readers to the original paper for side-by-side comparison.

6 Conclusion

We focused on the most fundamental examples of the exponential family models: continuous Gaussian and a discrete Bernoulli and observed the importance . The original paper provides a general framework to analyze more complex distributions that fall into exponential family, like Markov Random Fields, which we would like to explore in the future.

7 Acknowledgements

We would like to thank paper authors Alexander Shekhovtsov for discussing the paper and suggesting to try DisARM, and Dmitrij Schlesinger for feedback and encouraging us to implement experiments on our own.
We would like to thank Cesare Alippi, Andrea Cini and Ivan Marisca for enabling this work as part of Advanced Topics in Machine Learning course in USI. We want to thank Anthony Bugatto for the discussions on the paper.
We would like to thank anonymous reviewers for providing invaluable feedback on the paper.

References

1. D. P. Kingma and M. Welling. **Auto-Encoding Variational Bayes**. 2013. DOI: 10.48550/ARXIV.1312.6114. URL: https://arxiv.org/abs/1312.6114.
2. D. P. Kingma, T. Salimans, and M. Welling. "Improved Variational Inference with Inverse Autoregressive Flow." In: **ArXiv** abs/1606.04934 (2016).
3. D. J. Rezende and S. Mohamed. "Variational Inference with Normalizing Flows." In: **International Conference on Machine Learning**. 2015.
4. A. Vahdat and J. Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder." In: **ArXiv** abs/2007.03898 (2020).
5. J. Bae, M. R. Zhang, M. Ruan, E. Wang, S. Hasegawa, J. Ba, and R. B. Grosse. "Multi-Rate VAE: Train Once, Get the Full Rate-Distortion Curve." In: **The Eleventh International Conference on Learning Representations**. 2023. URL: https://openreview.net/forum?id=OJ8aSjCaMNK.
6. A. Shekhovtsov, D. Schlesinger, and B. Flach. "VAE Approximation Error: ELBO and Exponential Families." In: **International Conference on Learning Representations**. 2022. URL: https://openreview.net/forum?id=OIs3SxU5Ynl.
7. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets." In: **Advances in Neural Information Processing Systems**. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.
8. M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." In: **Advances in Neural Information Processing Systems**. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.
9. N. Inkawhich. **DCGAN Tutorial**. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html. Accessed: 2022-12-18.
10. A. Radford, L. Metz, and S. Chintala. **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks**. 2015. DOI: 10.48550/ARXIV.1511.06434. URL: https://arxiv.org/abs/1511.06434.
11. Z. Liu, P. Luo, X. Wang, and X. Tang. "Deep Learning Face Attributes in the Wild." In: **Proceedings of International Conference on Computer Vision (ICCV)**. Dec. 2015.
12. A. Brock, J. Donahue, and K. Simonyan. **Large Scale GAN Training for High Fidelity Natural Image Synthesis**. 2019. arXiv:1809.11096 [cs.LG].

13. S. Chaidaroon and Y. Fang. "Variational Deep Semantic Hashing for Text Documents." In: **Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval** (2017).
14. J. Rennie and K. Lang. "20 Newsgroups." In: **Proceedings of the Twelfth International Conference on Machine Learning**. 1995, pp. 331–339.
15. R. J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." In: **Mach. Learn.** 8.3–4 (May 1992), pp. 229–256. DOI: 10.1007/BF00992696. URL: https://doi.org/10.1007/BF00992696.
16. A. Shekhovtsov. "Bias-Variance Tradeoffs in Single-Sample Binary Gradient Estimators." In: **ArXiv** abs/2110.03549 (2021).
17. M. Yin and M. Zhou. **ARM: Augment-REINFORCE-Merge Gradient for Stochastic Binary Networks**. 2018. DOI: 10.48550/ARXIV.1807.11143. URL: https://arxiv.org/abs/1807.11143.
18. Z. Dong, A. Mnih, and G. Tucker. "DisARM: An Antithetic Gradient Estimator for Binary Latent Variables." In: (2020). DOI: 10.48550/ARXIV.2006.10680. URL: https://arxiv.org/abs/2006.10680.