Subset-Conditioned Generation Using Variational Autoencoder With A Learnable Tensor-Train Induced Prior

Maksim Kuznetsov
Insilico Medicine
Rockville, MD 20850
kuznetsov@insilico.com

Daniil Polykovskiy
Insilico Medicine
Rockville, MD 20850
daniil@insilico.com

Dmitry Vetrov
Higher School of Economics
Moscow, Russia
vetrovd@yandex.ru

Alexander Zhebrak
Insilico Medicine
Rockville, MD 20850
zhebrak@insilico.com

1 Introduction

Generative models appear in many applications from text [1, 2] and audio generation [3] to molecular design [4–10] Variational Autoencoders[11] are powerful generative models that find applications in many domains [12, 13]. In this paper, we explore the problem of *subset-conditioned generation* [14]—training a generative model that can sample objects conditioned on a subset of available properties. Unspecified properties may either be unknown (missing), or irrelevant for the given generation round. A straightforward approach to conditional generation with Conditional Variational Autoencoders (CVAE)[15] does not support omitted conditions, but can be turned into a subset-conditioned model by predicting missing variables.

In this paper, we parametrize a joint distribution on the conditions and the latent codes of VAE in a Tensor-Train format. The resulting model can efficiently sample from the conditional distributions for an arbitrary subset of conditions. The model also learns a flexible grid-structured prior distribution on the latent codes that can be used for down-stream tasks such as classification or clusterization.

2 Background

Tensor-Train (TT) decomposition [16] is a method for approximating high-dimensional tensors with a relatively small number of parameters. We use it to represent discrete distributions: consider a joint distribution $p(r_1, r_2, \dots r_n)$ of n discrete random variables r_k taking values from $\{0, 1, \dots N_k\}$. Let us denote this distribution as an n-dimensional tensor $P[r_1, r_2, \dots r_n] = p(r_1, r_2, \dots r_n)$. The number of elements in P grows exponentially with the number of dimensions. TT format reduces the number of parameters by approximating the tensor P using low-rank matrices—*cores*—$Q_k[r_k] \in \mathbb{R}^{m_k \times m_{k+1}}$:

$$P[r_1, r_2, \dots, r_n] \approx \mathbf{1}_{m_1}^T \cdot Q_1[r_1] \cdot Q_2[r_2] \cdot \dots \cdot Q_n[r_n] \cdot \mathbf{1}_{m_{n+1}}, \tag{1}$$

where $\mathbf{1}_m \in \mathbb{R}^{m \times 1}$ is a column-vector of ones. In this format, the number of parameters grows linearly with the number of dimensions. With larger cores, TT format can represent more complex distributions.

In TT format, we can compute marginal and conditional distributions and sample from them in polynomial time, without computing the whole tensor $P[r_1, r_2, ..., r_n]$. For example, to marginalize out the random variable r_k, we marginalize Q_k cores:

$$P[r_1, \dots, r_{k-1}, r_{k+1}, \dots r_n] = \mathbf{1}_{m_1}^T \cdot \prod_{j=1}^{k-1} Q_j[r_j] \cdot \left(\sum_{r_k=1}^{N_k} Q_k[r_k]\right) \cdot \prod_{j=k+1}^{n} Q_j[r_j] \cdot \mathbf{1}_{m_{n+1}} \tag{2}$$

We can compute a normalizing constant as a marginal over all variables and sample with the chain rule.

3 Variational Autoencoder with a Tensor-Train Induced Learnable Prior

In this section, we introduce Variational Autoencoder with a Tensor-Train Induced Learnable Prior (VAE-TTLP) and apply it to the subset-conditioned generation. As we described in Section 2, Tensor-Train approximation of a distribution can be used to efficiently compute conditionals and marginals. In VAE-TTLP model, we estimate a joint distribution $p(z, y)$ of VAE latent code z and conditions y in a Tensor-Train format. With this distribution, we can compute the likelihood of a partially labeled data by marginalizing out the unobserved conditions. During generation, we sample from conditional distribution over latent codes, given observed conditions.

3.1 Tensor-Train induced distribution with continuous variables

Tensor-Train decomposition generally works only with discrete distributions, while generative autoencoders mostly use continuous variables in the latent space. We assume that latent codes $z = [z_1, \dots z_d]$ are continuous, while conditions $y = [y_1, \dots y_n]$ are discrete. Our goal is to build a Tensor-Train representation for a joint distribution $p_\psi(z, y)$ that can model dependencies between z and y, as well as inner dependencies in z and y.

We parameterize continuous distribution as a mixture model and define a joint distribution by introducing auxiliary categorical random variables $s_k \in \{1, \dots N_k\}$ as the component indices:

$$p_\psi(z, y) = \sum_{s_1, \dots, s_d} p_\psi(z, s, y) = \sum_{s_1, \dots, s_d} p_\psi(s, y) p_\psi(z|s) \tag{3}$$

We parameterize $p_\psi(z|s)$ as a fully factorized Gaussian. The distribution $p_\psi(s, y)$ is a distribution over discrete variables and can be seen as a tensor $p_\psi(s, y) = W[s_1, \dots, s_d, y_1, \dots, y_n]$. We represent W in a Tensor-Train format. The resulting prior distribution becomes:

$$p_\psi(z, y) = \sum_{s_1, \dots, s_d} \underbrace{W[s_1, \dots, s_d, y_1, \dots, y_n]}_{p_\psi(s,y)} \underbrace{\prod_{k=1}^{d} \mathcal{N}(z_k \mid \mu_{k,s_k}, \sigma^2_{k,s_k})}_{p_\psi(z|s,y)} \tag{4}$$

3.2 VAE-TTLP training

Consider a training example (x, y_{ob}), where x is an object and y_{ob} are observed conditions. The lower bound for $\log p(x, y_{\text{ob}})$ in the VAE model with a learnable prior is:

$$\begin{aligned} \overline{\mathcal{L}}_{\text{TTLP}}(\theta, \phi, \psi) =& \mathbb{E}_{q_\phi(z|x,y_{\text{ob}})} \log p_\theta(x \mid z, y_{\text{ob}}) \\ &- \mathcal{KL}\big(q_\phi(z \mid x, y_{\text{ob}}) \mid\mid p_\psi(z \mid y_{\text{ob}})\big) \\ &+ \log p_\psi(y_{\text{ob}}) \end{aligned} \tag{5}$$

We make two assumptions: first, that all information about y is contained in the object x itself. For example, a hand-written digit (x) already contains the information about its label (y). Under this assumption, $q_\phi(z \mid x, y_{\text{ob}}) = q_\phi(z \mid x)$. Second, we assume that $p_\theta(x \mid z, y_{\text{ob}}) = p_\theta(x \mid z)$. In other words, an object is fully defined by its latent code. This results in a final ELBO:

$$\begin{aligned}\mathcal{L}_{\text{TTLP}}(\theta, \phi, \psi) =& \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x \mid z) - \mathcal{KL}\big(q_\phi(z \mid x) \mid\mid p_\psi(z \mid y_{\text{ob}})\big) + \log p_\psi(y_{\text{ob}}) \\ \approx& \frac{1}{l}\sum_{i=1}^{l}\left[\log p_\theta(x \mid z_i) - \log \frac{q_\phi(z \mid x)}{p_\psi(z \mid y_{\text{ob}})}\right] + \log p_\psi(y_{\text{ob}}).\end{aligned} \tag{6}$$

where $z_i \sim q_\phi(z \mid x)$. Since the joint distribution $p_\psi(z, y)$ is parameterized in TT format, we can compute posterior distribution on the latent code given the observed conditions $p_\psi(z \mid y_{\text{ob}})$ analytically. This model can also be used to fill in missing conditions by sampling $p_\psi(y_{\text{un}} \mid z)$.

3.3 VAE-TTLP sampling

We produce samples from $p(x \mid y_{\text{ob}})$ using the chain rule:

$$p(z \mid y_{\text{ob}}) = \prod_{k=1}^{d} p(z_k \mid y_{\text{ob}}, z_{<k}) \tag{7}$$

We then pass the latent code $z \sim p_\psi(z \mid y_{\text{ob}})$ through the decoder: $x \sim p_\theta(x \mid z)$. Note that the conditional distribution on z_k is a Gaussian mixture with unchanged centers μ_{k,s_k} and variances σ^2_{k,s_k}, but with different components weights:

$$\begin{aligned}p(z_k \mid y_{\text{ob}}, z_1, \ldots, z_{k-1}) &= \frac{p(y_{\text{ob}}, z_1, \ldots, z_{k-1}, z_k)}{p(y_{\text{ob}}, z_1, \ldots, z_{k-1})} \\ &= \sum_{s, y_{\text{un}}}\left[\frac{W[s, y_{\text{ob}}, y_{\text{un}}]}{\sum_{s', y_{\text{un}}} W[s', y_{\text{ob}}, y_{\text{un}}]}\right]\mathcal{N}(z_k \mid \mu_{k,s_k}, \sigma^2_{k,s_k})\end{aligned} \tag{8}$$

Weights of the components can be efficiently computed, since we represent W in a Tensor-Train format. The overall architecture is schematically shown in Figure 1.

4 Experiments

We provide experiments on two image datasets: MNIST and CelebA [17]. Both datasets have attributes which can be used for conditional learning and generation. MNIST has a categorical class label feature, while for CelebA we selected 10 binary attributes, including gender, hair color, smile, eyeglasses. Details on the experimental setup can be found in Appendix (Section 6.1).

Table 1: Numerical comparison of generated images from different models.

Model	MNIST		CelebA	
	FID	Accuracy, %	FID	Accuracy, %
CVAE	**39.10**	86.34	220.53	82.89
VAE-TTLP	40.80	**89.94**	165.33	**88.79**
VAE-TTLP (pretrained VAE)	47.53	75.39	**162.73**	87.5

In the experiments (Section 6.2), we show that VAE-TTLP model trained on MNIST dataset separates latent space into sub-manifolds with objects from one class.

We also compare VAE-TTLP with CVAE for conditional generation problem (Sec. 6.3) (without any missing variables in conditions) and show that the proposed model outperforms CVAE in image quality and satisfaction of conditions (Table 1).

In Section 6.5, we also evaluate the performance for various percentages of missing conditions and show that VAE-TTLP produces images consistent with conditions even when many of the conditions are missing. We also assess the joint distribution $p_\psi(z, y)$ by imputing missing variables $y_{\text{un}} \sim p_\psi(y_{\text{un}} \mid z, y_{\text{ob}})$.

5 Conclusion

We introduced a Variational Autoencoder with a Tensor-Train Induced Learnable Prior (VAE-TTLP) and applied it to the problem of subset-conditioned generation. Tensor-Train format allows our model to capture underlying dependencies between latent codes and labels. The model provides diverse samples satisfying specified conditions. VAE-TTLP can be extended to any auto-encoding model as a parameterization of the latent space.

References

[1] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3881–3890, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/yang17d.html`.

[2] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. In *International Conference on Learning Representations*, 2018.

[3] Adam Roberts, Jesse Engel, and Douglas Eck, editors. *Hierarchical Variational Autoencoders for Music*, 2017. URL `https://nips2017creativity.github.io/doc/Hierarchical_Variational_Autoencoders_for_Music.pdf`.

[4] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7805–7814. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/8005-constrained-graph-variational-autoencoders-for-molecule-design.pdf`.

[5] E. Putin, A. Asadulaev, Y. Ivanenkov, V. Aladinskiy, B. Sanchez-Lengeling, A. Aspuru-Guzik, and A. Zhavoronkov. Reinforced Adversarial Neural Computer for de Novo Molecular Design. *J Chem Inf Model*, 58(6):1194–1204, Jun 2018.

[6] Xiang Ren Will Hamilton Jiaxuan You, Rex (Zhitao) Ying and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. *ICML*, 2018.

[7] E. Putin, A. Asadulaev, Q. Vanhaelen, Y. Ivanenkov, A. V. Aladinskaya, A. Aliper, and A. Zhavoronkov. Adversarial Threshold Neural Computer for Molecular de Novo Design. *Mol. Pharm.*, Mar 2018.

[8] Regina Barzilay Wengong Jin and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *ICML*, 2018.

[9] Daniil Polykovskiy, Alexander Zhebrak, Dmitry Vetrov, Yan Ivanenkov, Vladimir Aladinskiy, Marine Bozdaganyan, Polina Mamoshina, Alex Aliper, Alex Zhavoronkov, and Artur Kadurin. Entangled conditional adversarial autoencoder for de-novo drug discovery. *Molecular Pharmaceutics*, sep 2018. doi: 10.1021/acs.molpharmaceut.8b00839. URL `https://doi.org/10.1021/acs.molpharmaceut.8b00839`.

[10] Artur Kadurin, Sergey Nikolenko, Kuzma Khrabrov, Alex Aliper, and Alex Zhavoronkov. druGAN: An advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular Pharmaceutics*, 14(9): 3098–3104, aug 2017.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2013.

[12] Rafael Gómez-Bombarelli, David K. Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *CoRR*, abs/1610.02415, 2016.

[13] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1878–1889. Curran Associates, Inc., 2017.

[14] Ramakrishna Vedantam, Ian Fischer, Jonathan Huang, and Kevin Murphy. Generative models of visually grounded imagination. In *International Conference on Learning Representations*, 2018.

[15] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc., 2014.

[16] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5): 2295–2317, 2011.

[17] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6629–6640, 2017.

6 Appendix

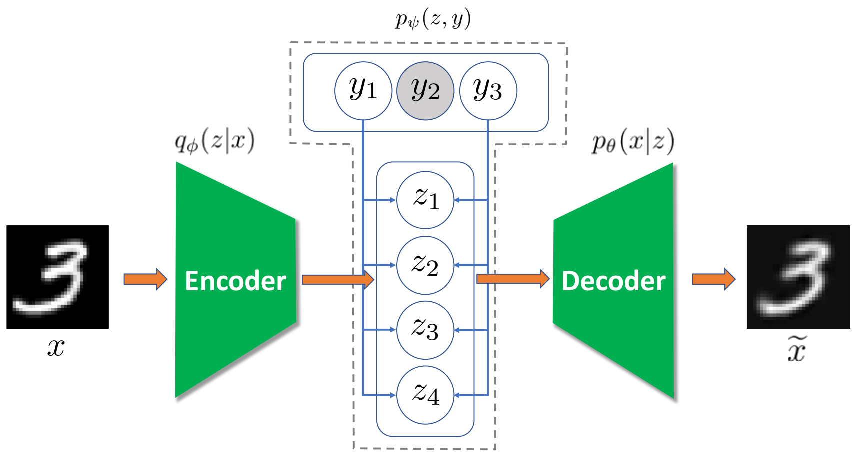


Figure 1: VAE-TTLP model. Autoencoder is trained to map object x to the latent code z. Joint distribution on conditions y and the latent code z is trained in a Tensor-Train format. This model can be used to generate samples from a subset of all possible conditions. For example, we can condition samples on properties y_1 and y_3 and omit y_2.

6.1 Experimental setup

We used 8-layer convolutional neural network (6 convolutional layers followed by 2 fully-connected layers) for the encoder and a symmetric architecture with deconvolutions for the decoder. MNIST images are 28x28 gray-scale images. In CelebA, we worked with images in 64x64 resolution.

6.2 Latent space structure

For the first experiment, we visualize the latent space of VAE-TTLP model trained on MNIST data. In Figure 2, we can see that the model assigned a separate cluster in the latent space for each label.

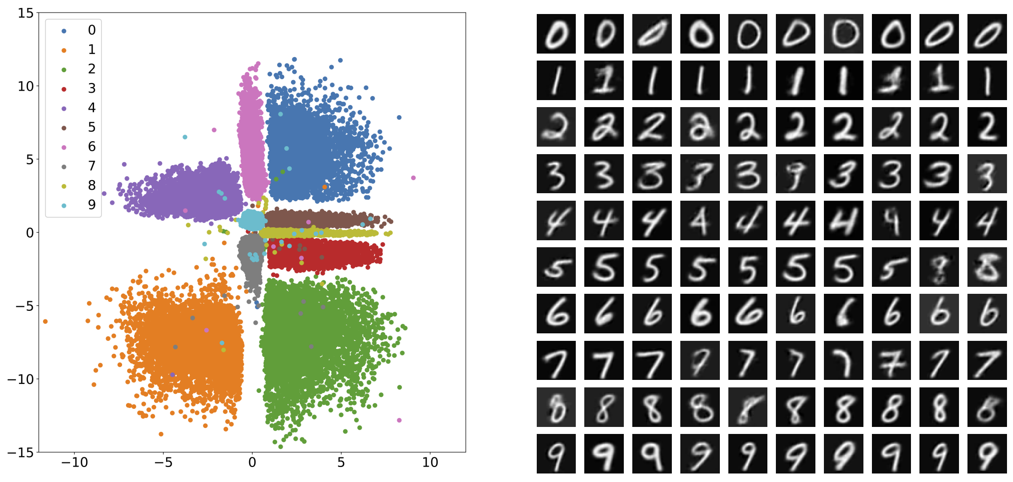


Figure 2: Samples from VAE-TTLP model trained on the MNIST dataset. **Left:** Learned latent space, **right:** Samples from the model

6.3 Generated images

In this experiment, we use CelebA dataset [17] to visually and numerically compare the quality of images generated with three models: CVAE, VAE-TTLP, and VAE-TTLP with a pretrained VAE. To estimate the visual quality of samples we calculate Fréchet Inception Distance (FID) [18] that was shown to correlate with assessor's opinion. To estimate how well a generated image matches specified condition, we predict image attributes with a separately trained predictor. Results are shown in Table 1 along with samples for visual analysis in Figure 3. Experiments suggest that VAE-TTLP outperforms or gives comparable results to CVAE in both visual quality and condition matching. Also, pretrained VAE model performs reasonably well, indicating that the model can be pretrained on unlabeled datasets.

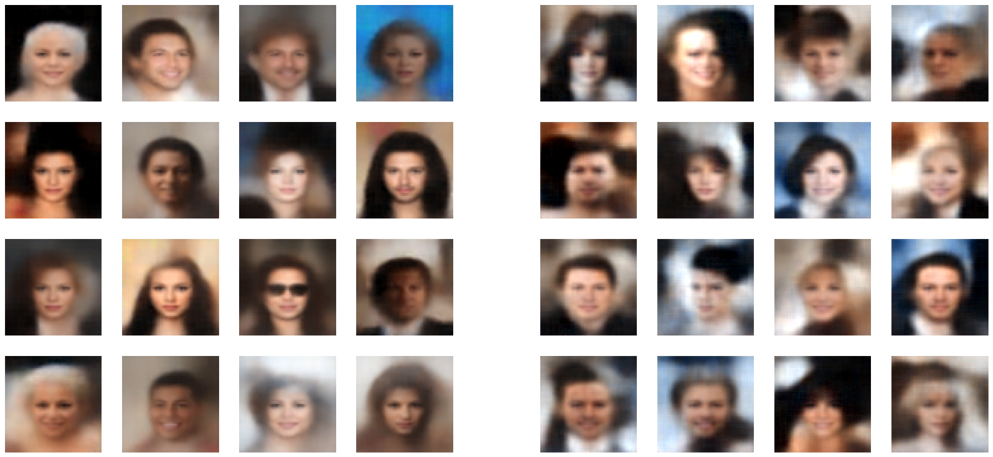


Figure 3: Samples from models trained on the CelebA dataset. **Left:** VAE-TTLP, **right:** CVAE

6.4 Training on partially labeled data

In this section, we estimate the performance of the model for different percentages of missing labels: fully labeled data, data with 30% and 70% randomly missing attributes. During the generation procedure we conditioned the model on all of the attributes. Numerical results are reported in Table 2. As seen in the results, the model is quite stable even when the dataset is sparsely labeled.

Table 2: Performance of VAE-TTLP on dataset with different percentage of missing attributes.

Missing attributes, %	MNIST		CelebA	
	FID	Accuracy, %	FID	Accuracy, %
0	40.80	89.94	165.33	88.7
30	41.33	89.84	178.32	84.89
70	41.86	88.97	169.10	87.08

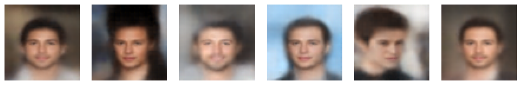
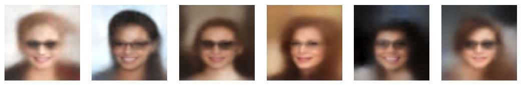
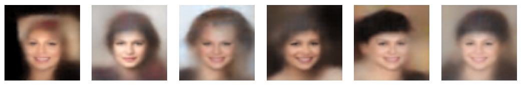
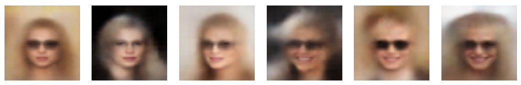
6.5 Imputing missing conditions with VAE-TTLP

As discussed in Section 3.2, VAE-TTLP model can be used for imputing missing conditions by sampling from distribution $p_\psi(y \mid x)$. On MNIST dataset we got 95.4% accuracy and 89.21% accuracy on CelebA. While state of the art models predict MNIST digits with over 99% accuracy and facial attributes with more than 93%, our model still performed reasonably well even though the predictive model was obtained as a by-product of VAE-TTLP.

6.6 Generated images conditioned on the subset of features

Finally, we generate images for a specified subset of conditions to estimate diversity of generated images. For example, if we specify an image to generate 'Young man', we would expect different images to have different hair colors, presence and absence of glasses and hat. Generated images shown in Figure 3 indicate that the model learned to produce diverse images with multiple varying attributes.

Table 3: Generated images for different attributes. Notice high diversity across rows.

Young man	
Smiling woman in eyeglasses	
Smiling woman wearing hat	
Blond haired woman wearing eyeglasses	
Attractive bearded man wearing hat