MPCC: Matching Priors and Conditionals for Clustering

A Matching marginals and conditionals is equivalent to matching joints

In the following sections we demonstrate that for the Kullback–Leibler divergence matching the marginals and conditionals is equivalent to matching the joint distributions. In Section A.1 we demonstrate this equivalence for the two variable case using the forward KL [9], this can be trivially demonstrated for the reverse KL [10] interchangeably replacing the variables z and x, and the models p and q. In Section A.2 we demonstrate the equivalence for the three variable case using the reverse KL (MPCC). The demonstration for the three-variable forward KL is equivalent requiring only to replace the variables y and x, and the models p and q.

A.1 Matching marginals and conditionals, two variable case

In Section 2 of the paper we show that maximizing the VAE objective (ELBO) can be interpreted as matching the conditional distributions in observed space and the marginal distributions in latent space. Let $p(x, z)$ and $q(x, z)$ be the decoder (*generation*) and encoder (*inference*) joint distributions, respectively, where x represents the observed variable (data) and z represents the continuous latent variable. The dependencies in these distributions are expressed as $p(x, z) = p(z)p(x|z)$ and $q(x, z) = q(z|x)q(x)$. In what follows we prove, for the two variable case, that matching conditionals and marginals is equivalent to matching the joint distributions in terms of the Kullback-Leibler divergence. Starting from the KL divergence between the joint distributions we can show that

$$
\begin{aligned}
& D_{\mathrm{KL}}\left(q(x,z)||p(x,z)\right) \\
= & \int_x \int_z q(x,z) \log \frac{q(x,z)}{p(x,z)}\, dz\, dx \\
= & \int_x q(x) \int_z q(z|x) \log \frac{q(z|x)}{p(z|x)}\, dz\, dx + \int_x q(x) \log \frac{q(x)}{p(x)}\, dx \\
= & \mathbb{E}_{x\sim q(x)}\left[D_{\mathrm{KL}}\left(q(z|x)||p(z|x)\right)\right] + D_{\mathrm{KL}}\left(q(x)||p(x)\right), \qquad (1)
\end{aligned}
$$

i.e. the forward KL divergence between the joints is indeed equivalent to the forward KL between the marginals in data space plus the expected value under the data distribution of the forward KL between the conditionals in latent space.

A.2 Matching priors and conditionals for three variables *a.k.a* MPCC case

In Section 3 of the paper we define the MPCC model starting from a joint distribution matching perspective. Let $p(x, z, y)$ and $q(x, z, y)$ be the decoder (*generation*) and encoder (*inference*) joint distributions, respectively, where x represents the observed variable (data), z represents the continuous latent variable and y represents the cluster membership. The dependencies in these distributions are expressed as $p(x, z, y) = p(y)p(z|y)p(x|z, y)$ and $q(x, z, y) = q(y|z)q(z|x)q(x)$. The only assumption is that $q(y|x, z) = q(y|z)$. In what follows we prove, for the three variable case, that matching conditionals and marginals is equivalent to matching the joint distributions in terms of the Kullback-Leibler divergence.

Starting from the (reverse) KL divergence between the joint distributions we show that

$$
\begin{aligned}
& D_{\mathrm{KL}}\left(p(x,z,y)||q(x,z,y)\right) \\
= & \int_x \int_y \int_z p(x,z,y) \log \frac{p(x,z,y)}{q(x,z,y)}\, dx\, dy\, dz \\
= & \int_y \int_z p(z,y) \int_x p(x|z,y) \log \frac{p(x|z,y)}{q(x|z,y)}\, dx\, dy\, dz \\
+ & \int_y p(y) \int_z p(z|y) \log \frac{p(z|y)}{q(z|y)}\, dz\, dy + \int_y p(y) \log \frac{p(y)}{q(y)}\, dy \\
= & \mathbb{E}_{z,y\sim p(z,y)}\left[D_{\mathrm{KL}}\left(p(x|z,y)||q(x|z,y)\right)\right] \\
& + \mathbb{E}_{y\sim p(y)}\left[D_{\mathrm{KL}}\left(p(z|y)||q(z|y)\right)\right] + D_{\mathrm{KL}}\left(p(y)||q(y)\right), \qquad (2)
\end{aligned}
$$

i.e. the KL between the joints is the sum of the KL divergences for $x|(z, y)$, $z|y$ and y, respectively. The KL divergence is non-negative so if we match the priors and conditionals then the joints have to match too.

B Variational Deep Embedding (VaDE)

B.1 Variational Deep Embedding matches conditionals and marginals in data space

Here we show that Variational Deep Embedding (VaDE) [7] is in fact matching the joint distributions of the encoder and decoder by matching posteriors and marginal in the space of the observed variable x (data). We start by expanding the divergence between the joint distribution of the encoder and decoder as:

$$
\begin{aligned}
& D_{\mathrm{KL}}\left(q(x,z,y)||p(x,z,y)\right) \\
= & \int_x \int_y \int_z q(x,z,y) \log \frac{q(x,z,y)}{p(x,z,y)}\, dx\, dy\, dz \\
= & \mathbb{E}_{z,x\sim q(z,x)}\left[D_{\mathrm{KL}}\left(q(y|z,x)||p(y|z,x)\right)\right] \\
& + \mathbb{E}_{x\sim q(x)}\left[D_{\mathrm{KL}}\left(q(z|x)||p(z|x)\right)\right] + D_{\mathrm{KL}}\left(q(x)||p(x)\right). \qquad (3)
\end{aligned}
$$

The first divergence in the right hand side of Eq. (3) is

$$D_{\mathrm{KL}}\left(q(y|z,x)||p(y|z,x)\right) = \int q(y|x)\log\frac{q(y|x)p(z)}{p(z|y)p(y)}\,dy$$
$$= \mathbb{E}_{y\sim q(y|x)}\left[\log\frac{q(y|x)}{p(z|y)p(y)}\right] + \log p(z), \qquad (4)$$

where we used the replacements $q(y|z,x) = q(y|x)$ and $p(y|z,x) = \frac{p(x|z)p(z|y)p(y)}{p(x|z)p(z)}$, which come from the graphical model assumptions considered in [7].

The second divergence in the right hand side of Eq. (3) is

$$D_{\mathrm{KL}}\left(q(z|x)||p(z|x)\right) = \int q(z|x)\log\frac{q(z|x)p(x)}{p(x|z)p(z)}\,dz$$
$$= \mathbb{E}_{z\sim q(z|x)}\left[\log\frac{q(z|x)}{p(x|z)} - \log p(z)\right] + \log p(x), \qquad (5)$$

and the third divergence in the right hand side of Eq. (3) is

$$D_{\mathrm{KL}}\left(q(x)||p(x)\right) = \mathbb{E}_{x\sim q(x)}\left[\log q(x) - \log p(x)\right]. \qquad (6)$$

If we add the expectation over $q(z,x) = q(z|x)q(x)$ of Eq. (4) with the expectation over $q(x)$ of Eq. (5) and Eq. (6) we obtain:

$$\begin{aligned}
&\mathbb{E}_{z,x\sim q(z,x)}\left[D_{\mathrm{KL}}\left(q(y|z,x)||p(y|z,x)\right)\right] + \mathbb{E}_{x\sim q(x)}\left[D_{\mathrm{KL}}\left(q(z|x)||p(z|x)\right)\right] \\
&+ D_{\mathrm{KL}}\left(q(x)||p(x)\right) \\
&= \mathbb{E}_{q(x)}[\log q(x) - \mathbb{E}_{z,y\sim q(z,y|x)}[\log p(x|z) - \log q(z|x) - \log q(y|x) \\
&+ \log p(z|y) + \log p(y)]] \\
&= \mathbb{E}_{q(x)}\left[\log q(x) - \mathcal{L}_{\mathrm{VaDE}}(x)\right],
\end{aligned}$$

where $\mathcal{L}_{\mathrm{VaDE}}(x)$ corresponds to Eq. (9) in [7]. This means that by maximizing VaDE's loss function one is matching the conditionals and marginals between encoder and decoder in data space. Note that the entropy of the data distribution $\mathbb{E}_{q(x)}\left[\log q(x)\right]$ is constant during optimization.

B.2 Why extending VaDE to any multi-modal distribution is harder than MPCC?

In MPCC the latent space can be naturally extended to any mixture of distributions, the only requirement being that the entropy of each distribution component $p(z|y)$ should have a closed-form or at least a bound. In general any model decomposed by the reverse KL enjoy this property.

Forward KL decompositions, such as the case of VAE and VaDE, need a closed-form solution for the divergence between the posterior and the prior. In VaDE this term corresponds to

$$\mathbb{E}_{q(x)}\mathbb{E}_{q(z,y|x)}[\log q(z|x) - \log p(z|y)] = \mathbb{E}_{q(x)}\mathbb{E}_{q(y|x)}[D_{KL}(q(z|x)||p(z|y))], \quad (7)$$

which has a closed-form since $q(z|x)$ and $p(z|y)$ are Gaussians. Other distributions can be used however they need to be from the exponential family and to have the same distribution [14], although some exceptions exists [19], [2]. In addition to the exponential family requirement, a reparameterization trick is needed for the posterior distribution further limiting the distributions that can be used and requiring other forms of reparameterization [4], [17], [6].

Alternatively, adversarial training can be used to match the marginal posterior with more flexible priors. However it has been observed [16] that this kind of optimization [11], [12] underestimates its Kullback-Leibler divergence and also worsen the likelihood of the decoder likely affecting its clustering capabilities.

C Details on neural network architectures

In MPCC we use the BigGAN model techniques [1] as a base for all our experiments. This architecture employs ResNet [5] and Spectral Normalization [13]. The residual block components of the generator and discriminator/encoder are shown in Fig. 1 (a) and (b), respectively. All the 3×3 Conv use a padding equal to one while 1×1 Conv have no padding. The upsampling operation of the generator is done using bilinear interpolation. A general scheme of the generator, discriminator and encoder architecture is shown in Fig. 2. The first residual block of the discriminator/encoder inverts the order of the 1×1 Conv and the average pooling and omits the first ReLU activation. Residual blocks with an asterisk correspond to the ones that do not perform average pooling and as a consequence they do not use 1×1 Conv.

Fig. 2 (a) shows the generator used for the CIFAR10/20 datasets. Fig 2 (b) shows the unconditional architecture of the discriminator. In the case of the conditional discriminator a term $\text{Embed}(y) \cdot h$ is added where h is the output of the global sum pooling (see Table 2). We can write the architectures for all datasets in a general way as in Fig. 2 or more specific as in Tables 1, 2 and 3, where C, J and D change between datasets.

As we observed in the paper (Table 3) we found an improvement in terms of sampling quality and reconstruction error when parameters between the discriminator and the encoder are shared. We experimented on the number of residual blocks shared and found that the best performance was obtained when sharing the first three residual blocks.

We use the Adam optimizer [8] with its default parameters $\beta_1 = 0$ and $\beta_2 = 0.999$. We use exponential moving average (EMA) with a decay rate of 0.9999 for the Generator for both sampling and reconstruction task. EMA is applied after the 1000th iteration. All generator, discriminator and encoder parameters use Spectral Normalization and are initialized with $\mathcal{N}(0, 0.02I)$, while an orthogonal initialization is used for prior parameters [18]. With the exception of the prior parameters we use a learning rate of $2e-4$ for all networks and experiments. For evaluation we use standing statistics [1] *i.e.* in evaluation mode we run many times (in our case 16) the forward propagation of the generator model

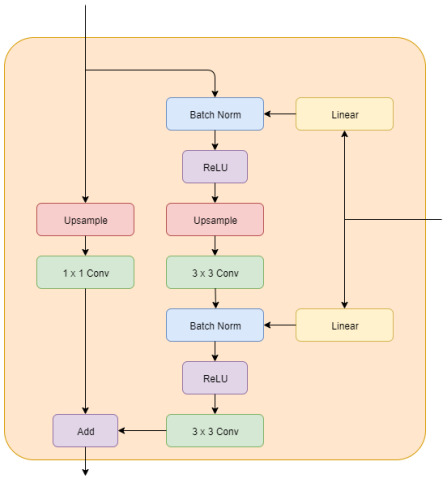
$\tilde{x} \sim p(x, z, y)$ storing the means and variances aggregated across all forward passes.

We use three techniques depending on the dataset to deliver the latent information z and y into the decoder distribution $p(x|z, y)$. The first two correspond to a hierarchical latent space architecture [1] which concatenate Embed(y) with a subset of z, and then a linear transformation to estimate the statistics of the batch norm layers is applied (see Fig. 1). The first method is the one observed in Fig. 2 which splits the latent variable z into equal chunks, delivering each one to a different part of the network. In this case we have four chunks (1 entry + 3 residual blocks). The second method is similar to first, the only difference being that all z is shared and no split operation is done. The schematic of this generator is equivalent to Fig. 2 (a) except that the purple box performs a copy instead of split operation. The third method passes all the latent z as usual [15] and uses conditional batch normalization [3]. This method learns embeddings conditioned on y which are different for each layer, *i.e.* the linear transformation in the yellow boxes of Fig. 1 correspond to an embedding, and the shared embedding should be ignored.

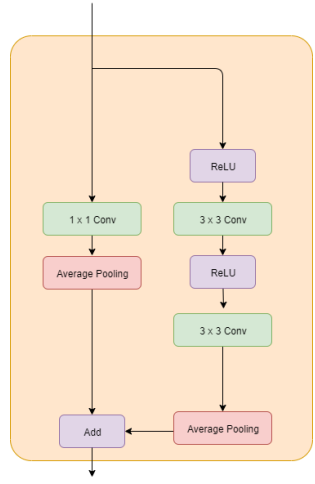
- For CIFAR10 and CIFAR20 we use the first method since this is the default architecture used in BigGAN. For simplicity we kept this configuration for all ablation and clustering experiments with these datasets. We found that mode collapse problems would appear if the third method is used in these datasets. The configuration of the parameters for these datasets is $C = 96$, $D = 3$ (RGB), $J = 128$ and $\eta_p = 6 \cdot 10^{-4}$.
- For datasets with simpler distributions such as MNIST and Omniglot, the third method is more stable and yields the best results. We found that if we use hierarchical latent space architectures poor results were obtained. In particular we observed that the chunks in the first method are decorrelated, which is particularly bad for simpler datasets such as MNIST and Omniglot because the network gains lot of capacity ignoring the embedding y and learning the full real distribution in all the clusters. The configuration of the parameters for these datasets is $D = 1$ (grayscale), $J = 24$ and $\eta_p = 1.6 \cdot 10^{-3}$. For MNIST $C = 12$ and for Omniglot $C = 16$.
- For FMNIST we observed that a poor performance was obtained with both the first and third method. The best results for this dataset were obtained using the second method. The configuration of the parameters for this dataset is $C = 24$, $D = 1$ (gray), $J = 16$ and $\eta_p = 1.6 \cdot 10^{-3}$.

We develop a Pytorch implementation for MPCC based on the implementation of BigGAN[1]. The IS and FID scores are calculated using the official implementations[2]. We run each model in a GeForce RTX 2080 Ti, the amount of time that MPCC iterates depends on the dataset but it is within the range of 12-24 hours.

[1] `https://github.com/ajbrock/BigGAN-PyTorch`
[2] `https://github.com/bioinf-jku/TTUR`

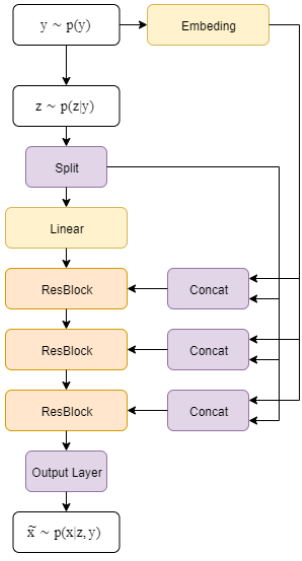


(a) Generator block

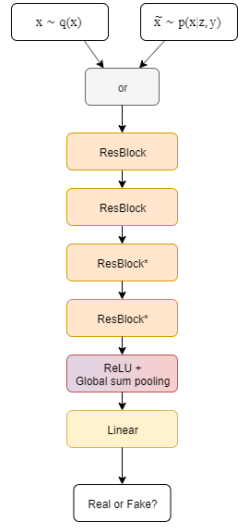


(b) Discriminator/Encoder block

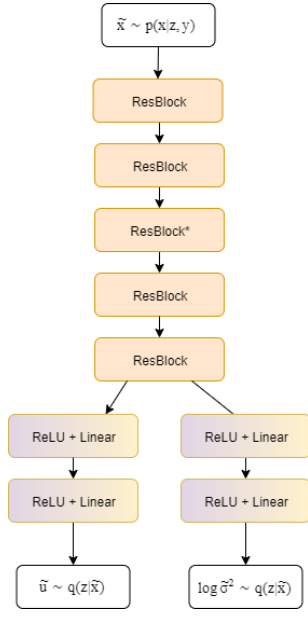
Fig. 1. Residual blocks used for MPCC generator, discriminator and encoder networks.



(a) Generator



(b) Discriminator



(c) Encoder

Fig. 2. Architectures of MPCC generator, discriminator and encoder networks, respectively

$y_i \in \{0, \ldots, K-1\} \sim Cat(\phi)$ $z_i \in \mathbb{R}^J \sim \mathcal{N}(\mu_{y_i}, \sigma^2_{y_i})$ Share Embed$(y) \in \mathbb{R}^J$
Linear$(J) \to 4 \times 4 \times 4C$
Resblock up $4C \to 4C$
Resblock up $4C \to 4C$
Resblock up $4C \to 4C$
Output Layer: BN, ReLU, 3×3 Conv $C \to 3$ Tanh

Table 1. Generator

$x \in \mathbb{R}^{32 \times 32 \times D}$
Resblock down $4C \to 4C$
Resblock down $4C \to 4C$
Resblock $4C \to 4C$
Resblock $4C \to 4C$
ReLU, Global sum pooling
(linear $\to 1$) if conditional :+ Embed$(y) \cdot h$

Table 2. Discriminator

$x \in \mathbb{R}^{32 \times 32 \times D}$
Resblock down $4C \to 4C$
Resblock down $4C \to 4C$
Resblock $4C \to 4C$
Resblock down $4C \to 4C$
Resblock down $4C \to 4C$
Flatten
$\times 2$: Linear $(32//2^4 \times 4C) \to (32//2^4 \times 4C)//2$
$\times 2$: Linear $((32//2^4 \times 4C)//2) \to J$

Table 3. Encoder

D Optimization problems

We observed two types of errors which restrict the architecture and the optimization techniques. Theses difficulties are particularly relevant for the CIFAR10 and CIFAR20 datasets which present the more complex distributions. We used the default parameters of the CIFAR10 architecture unless otherwise stated.

The first problem is associated with the batch size. We found that we can't optimize MPCC with a big batch size while using a large learning rate of the prior parameter η_p. Note that the latter is necessary to obtain good accuracy performance as it was shown in the paper (Table 2 in the paper). The batch size is relevant to increase the IS and FID scores [1]. Artifacts or saturation

problems would appear when doing a small modification in the optimization. The examples shown in Fig. 3 (a) use a batch size slightly larger than the one used in the paper (50). We observe that using a slightly larger batch size (64) with a prior learning rate of $\eta_p = 8 \cdot 10^4$ the results change drastically and the generated images show notable saturation.

Mode collapse is an important topic in GANs research and is the second problem that we observed in MPCC. Usually it is associated with the limitations in generation quality caused by the model, which memorize only a small part of the real distribution affecting the performance of the GAN. In MPCC the mode collapse problem can make an entire cluster collapse. Setting $D_{step} = 4$ solves this problem partially for a large amount of models and is sufficient to obtain good performance. In Fig. 3 (b) we show samples from a model trained with $bs = 64$ and $\eta_p = 2 \cdot 10^{-4}$ where we can see how a mode collapse problem looks in MPCC. We observed that when using a large prior learning rate $\eta_p = 6 \cdot 10^{-4}$ this problem would regularly appear after 150,000 iterations. This doesn't occur for the best configuration of MPCC (the one reported in the paper) and setting $\eta_p = 2 \cdot 10^{-4}$ even after a large number of iterations, however we would like to increase η_p further as we observed that it correlates with better clustering accuracy (Table 2 in the paper).

(a) Saturation problems

(b) Mode collapse problems

Fig. 3. Generated images with bad optimization setting at iteration 50000. Sub-figure (a) shows images associated with saturation problems and (b) with mode collapse problems. Each row represents a different cluster.

E Additional qualitative results

In this section we provide additional reconstructions and samples for the CIFAR-10 dataset in Figures 4 and 5, and for the MNIST dataset in 6 and 7. To give more insight about MPCC's capacity we also include samples for datasets with a high number of classes, CIFAR-20 and Omniglot in Figures 8 and 9 respectively.

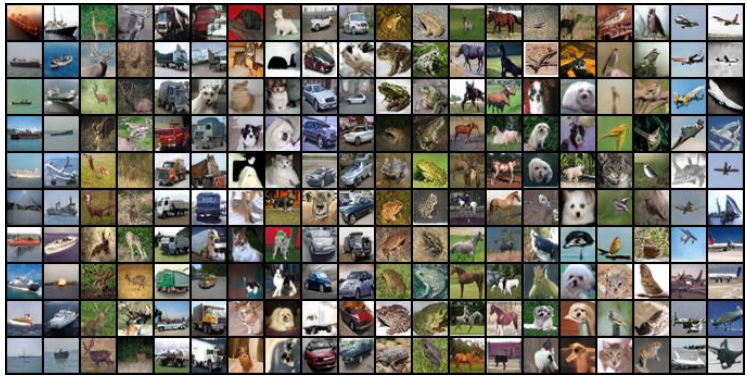


Fig. 4. Generated images for the CIFAR-10 dataset. Every two columns we set a different value for the categorical latent variable y. *i.e.* the samples shown correspond to a different conditional latent space $z \sim p(z|y)$.

Fig. 5. Reconstructions for rhe CIFAR-10 dataset. Odd columns represent real data and even columns correspond to their reconstructions. The real label is used to sort the column pairs.

Fig. 6. Generated images for the MNIST dataset. Every two columns we set a different value for the categorical latent variable y. *i.e.* the samples shown correspond to a different conditional latent space $z \sim p(z|y)$.

Fig. 7. Reconstructions for the MNIST dataset. Odd columns represent real data and even columns correspond to their reconstructions. The real label is used to sort the column pairs.

Fig. 8. Generated images for CIFAR-20 dataset. In every row we set a different value for the categorical latent variable y, *i.e.* the samples shown correspond to a different conditional latent space $z \sim p(z|y)$.

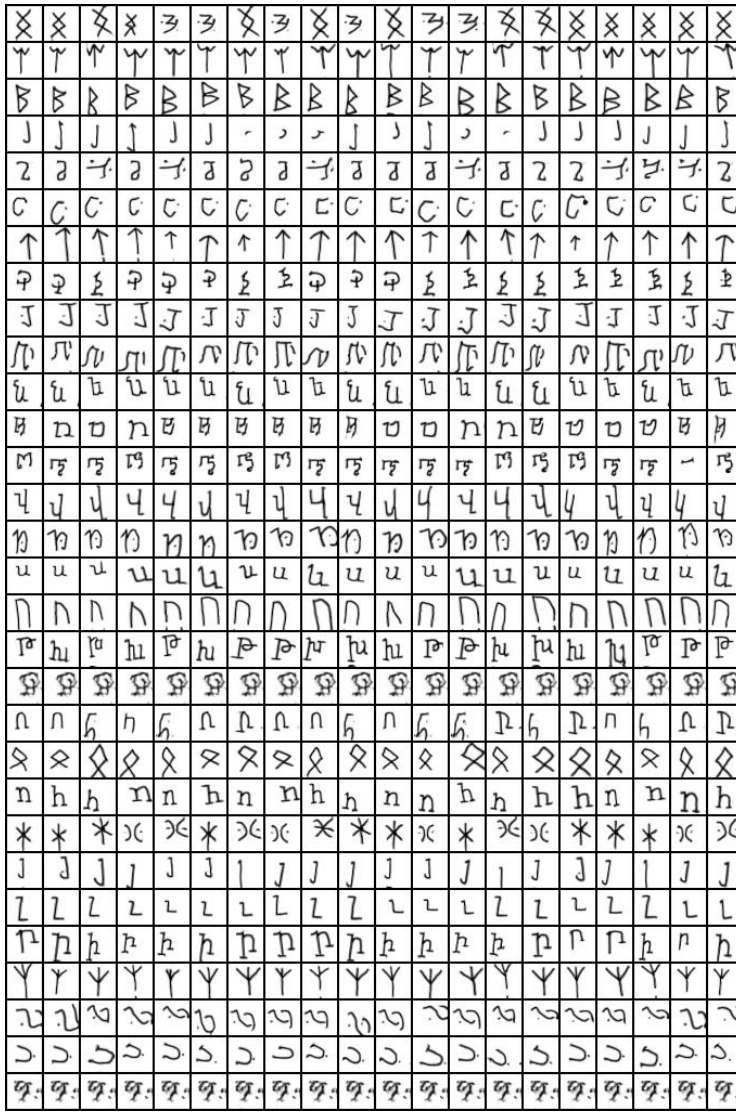


Fig. 9. Generated images for Omniglot dataset. In every row we set a different value for the categorical latent variable y, *i.e.* the samples shown correspond to a different conditional latent space $z \sim p(z|y)$. 30 cluster were randomly chosen.

References

1. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=B1xsqj09Fm
2. Chyzak, F., Nielsen, F.: A closed-form formula for the Kullback-Leibler divergence between Cauchy distributions (Dec 2019), https://hal.inria.fr/hal-02420591, 8 pages
3. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. ICLR (2017), https://arxiv.org/abs/1610.07629
4. Figurnov, M., Mohamed, S., Mnih, A.: Implicit reparameterization gradients. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 441–452. Curran Associates, Inc. (2018), http://papers.nips.cc/paper/7326-implicit-reparameterization-gradients.pdf
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (June 2016). https://doi.org/10.1109/CVPR.2016.90
6. Jankowiak, M., Obermeyer, F.: Pathwise derivatives beyond the reparameterization trick. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 2235–2244. PMLR, Stockholmsmässan, Stockholm Sweden (10–15 Jul 2018), http://proceedings.mlr.press/v80/jankowiak18a.html
7. Jiang, Z., Zheng, Y., Tan, H., Tang, B., Zhou, H.: Variational deep embedding: an unsupervised and generative approach to clustering. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 1965–1972. AAAI Press (2017)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014), http://arxiv.org/abs/1412.6980, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015
9. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014), http://arxiv.org/abs/1312.6114
10. Li, H., Wang, Y., Chen, C., Gao, J.: AIM: Adversarial inference by matching priors and conditionals (2019), https://openreview.net/forum?id=rJx_b3RqY7
11. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I.: Adversarial autoencoders. In: International Conference on Learning Representations (2016), http://arxiv.org/abs/1511.05644
12. Mescheder, L., Nowozin, S., Geiger, A.: Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 2391–2400. PMLR, International Convention Centre, Sydney, Australia (06–11 Aug 2017)
13. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=B1QRgziT-
14. Nielsen, F., Nock, R.: Entropies and cross-entropies of exponential families. In: 2010 IEEE International Conference on Image Processing. pp. 3621–3624 (Sep 2010). https://doi.org/10.1109/ICIP.2010.5652054

15. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), http://arxiv.org/abs/1511.06434
16. Rosca, M., Lakshminarayanan, B., Mohamed, S.: Distribution matching in variational inference. CoRR **abs/1802.06847** (2018)
17. Ruiz, F.R., Titsias RC AUEB, M., Blei, D.: The generalized reparameterization gradient. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 460–468. Curran Associates, Inc. (2016), http://papers.nips.cc/paper/6328-the-generalized-reparameterization-gradient.pdf
18. Saxe, A.M., Mcclelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In: In International Conference on Learning Representations (2014)
19. Soch, J., Allefeld, C.: Kullback-leibler divergence for the normal-gamma distribution (2016)