CaGeM: A Cluster Aware Deep Generative Model

Lars Maaløe **Marco Fraccaro** **Ole Winther**
Technical University of Denmark
{larsma,marfra,olwi}@dtu.dk

Abstract

Deep generative models trained with large amounts of unlabelled data have proven to be powerful within the domain of unsupervised learning. Many real life data sets contain a small amount of labelled data points, that are typically disregarded when training generative models. We propose the *Cluster-aware Generative Model*, that offers a way to couple information from a small fraction of labelled data with the information from unlabelled data to improve generative performance. This paradigm is denoted semi-supervised generation and is achieved by embedding cluster information into the higher latent stochastic units of a variational auto-encoder. With the information from only a small fraction of labelled data points, the generative performance of the model improves by more than 2 nats when comparing to similar models on permutation invariant MNIST, obtaining a log-likelihood of -79.38 nats. Even fully unsupervised, the model improves the log-likelihood performance. The model can also be applied to semi-supervised classification tasks on which it performs comparable to state-of-the-art.

1 Introduction

Variational Auto-Encoders (VAE) [17, 26] and Generative Adversarial Networks (GAN) [10] have shown promising generative performances on data from complex high-dimensional distributions. Both approaches have spawn numerous related deep generative models, not only to model data points like those in a large unlabelled training data set, but also for semi-supervised classification [15, 20, 30, 28]. In semi-supervised classification a few points in the training data are endowed with class labels, and the plethora of unlabelled data aids to improve a supervised classification model.

Could a few labelled training data points in turn improve a deep generative model? This reverse perspective, doing semi-supervised generation, is investigated in this work. Many of the real life data sets contain a small amount of labelled data, but incorporating this partial knowledge in the generative models is not straightforward, because of the risk of overfitting towards the labelled data. This overfitting can be avoided by finding a good scheme for updating the parameters, like the one introduced in the models for semi-supervised classification [15, 20]. However, there is a difference in optimizing the model towards optimal classification accuracy and generative performance. We introduce the *Cluster-aware Generative Model (CaGeM)*, an extension of a VAE, that improves the generative performances, by being able to model the natural clustering in the higher feature representations through a discrete variable [3]. The model can be trained fully unsupervised, but its performances can be further improved using labelled class information that helps in constructing well defined clusters.

The main contributions of the paper are:

i. CaGeM, a model that presents a novel approach to extend the VAE with a discrete variable enhancing clustering in the higher latent variables and enabling semi-supervised generation.

ii. Improving generative log-likelihood performance significantly within unsupervised and semi-supervised generation.

2 Variational Auto-encoders

A *Variational Auto-Encoder (VAE)* [17, 26] defines a deep generative model for data x that depends on latent variable z or a hierarchy of latent variables, e.g. $z = [z_1, z_2]$. The joint distribution of the two-level generative model is given by $p_\theta(x, z_1, z_2) = p_\theta(x|z_1)p_\theta(z_1|z_2)p(z_2)$, where

$$p_\theta(z_1|z_2) = \mathcal{N}(z_1; \mu_\theta^1(z_2), \sigma_\theta^1(z_2)); \quad p(z_2) = \mathcal{N}(z_2; 0, I)$$

are Gaussian distributions with a diagonal covariance matrix and $p_\theta(x|z_1)$ is typically a parameterized Gaussian (continuous data) or Bernoulli distribution (binary data). The probability distributions of the generative model of a VAE are parameterized using deep neural networks whose parameters are denoted by θ. Training is performed by optimizing the *Evidence Lower Bound (ELBO)*, a lower bound to the intractable log-likelihood $\log p_\theta(x)$ obtained using Jensen's inequality:

$$\log p_\theta(x) = \log \iint p_\theta(x, z_1, z_2)\mathrm{d}z_1\mathrm{d}z_2 \geq \mathbb{E}_{q_\phi(z_1,z_2|x)}\left[\log \frac{p_\theta(x, z_1, z_2)}{q_\phi(z_1, z_2|x)}\right] = \mathcal{F}(\theta, \phi) \ . \quad (1)$$

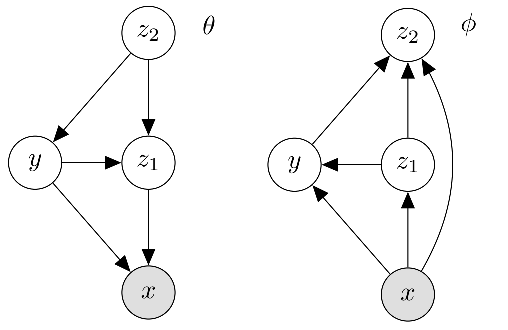
The introduced variational distribution $q_\phi(z_1, z_2|x)$ is an approximation to the model's posterior distribution $p_\theta(z_1, z_2|x)$, defined with a bottom-up dependency structure where each variable of the model depends on the variable below in the hierarchy, so that $q_\phi(z_1, z_2|x) = q_\phi(z_1|x)q_\phi(z_2|z_1)$ and

$$q_\phi(z_1|x) = \mathcal{N}(z_1; \mu_\phi^1(x), \sigma_\phi^1(x)); \quad q_\phi(z_2|z_1) = \mathcal{N}(z_2; \mu_\phi^2(z_1), \sigma_\phi^2(z_1)) \ .$$

Similar to the generative model, the mean and diagonal covariance of both Gaussian distributions defining the inference network q_ϕ are parameterized with deep neural networks that depend on parameters ϕ.

3 Cluster-aware Generative Models

Hierarchical models parameterized by deep neural networks have the ability to represent very flexible distributions. In VAEs however, the addition of more stochastic layers is often accompanied with a built-in pruning effect so that the higher layers become inactive and therefore not exploited by the model [6, 29], see Appendix A for a more detailed discussion on the topic. As we will see, in CaGeM the possibility of learning a representation in the higher stochastic layers that can model the natural clustering of the data [3] drastically reduces this issue. This results in a model that is able to disentangle some of the factors of variation in the data and that extracts a hierarchy of features beneficial during the generation phase.



(a) CaGeM Generative p_θ (b) CaGeM Inference q_ϕ

Figure 1: Generative model and inference model of CaGeM with two stochastic layers.

We favor the flow of higher-level global information through z_2 by extending the generative model of a VAE with a discrete variable y representing the choice of one out of K different clusters in the data. The joint distribution $p_\theta(x, z_1, z_2)$ is computed by marginalizing over y:

$$p_\theta(x, z_1, z_2) = \sum_y p_\theta(x, y, z_1, z_2) = \sum_y p_\theta(x|y, z_1)p_\theta(z_1|y, z_2)p_\theta(y|z_2)p(z_2) \ .$$

We call this model *Cluster-aware Generative Model (CaGeM)*, see Figure 1 for a graphical representation. The introduced categorical distribution $p_\theta(y|z_2) = \mathrm{Cat}(y; \pi_\theta(z_2))$ (π_θ represents the class distribution) depends solely on z_2, that needs therefore to stay active for the model to be able to represent clusters in the data. We further add the dependence of z_1 and x on y, so that they can now both also represent cluster-dependent information.

3.1 Inference

We define the variational approximation $q_\phi(y, z_1, z_2|x)$ over the latent variables of the model as

$$q_\phi(y, z_1, z_2|x) = q_\phi(z_2|x, y, z_1)q_\phi(y|z_1, x)q_\phi(z_1|x) \ ,$$

where $q_\phi(z_1|x) = \mathcal{N}(z_1; \mu^1_\phi(x), \sigma^1_\phi(x))$, $q_\phi(y|z_1, x) = \text{Cat}(y; \pi_\phi(z_1, x))$ and $q_\phi(z_2|x, y, z_1) = \mathcal{N}(z_2; \mu^2_\phi(y, z_1), \sigma^2_\phi(y, z_1))$. In the inference network we then reverse all the dependencies among random variables in the generative model (the arrows in the graphical model in Figure 1). This results in a *bottom-up* inference network that performs a feature extraction that is fundamental for learning a good representation of the data. Starting from the data x we construct higher levels of abstraction, first through the variables z_1 and y, and finally through the variable z_2, that includes the global information used in the generative model. In order to make the higher representation more expressive we add a skip-connection from x to z_2, that is however not fundamental to improve the performances of the model[1].

With this factorization of the variational distribution $q_\phi(y, z_1, z_2|x)$, the ELBO can be written as

$$\mathcal{F}(\theta, \phi) = \mathbb{E}_{q_\phi(z_1|x)}\left[\sum_y q_\phi(y|z_1, x)\mathbb{E}_{q_\phi(z_2|x,y,z_1)}\left[\log \frac{p_\theta(x, y, z_1, z_2)}{q_\phi(y, z_1, z_2|x)}\right]\right] .$$

We maximize $\mathcal{F}(\theta, \phi)$ by jointly updating, with stochastic gradient ascent, the parameters θ of the generative model and ϕ of the variational approximation. When computing the gradients, the summation over y is performed analytically, whereas the intractable expectations over z_1 and z_2 are approximated by sampling. We use the reparameterization trick to reduce the variance of the stochastic gradients [17, 26].

4 Semi-Supervised Generation with CaGeM

In some applications we may have class label information for some of the data points in the training set. In the following we will show that CaGeM provides a natural way to exploit additional labelled data to improve the performance of the generative model. Notice that this *semi-supervised generation* approach differs from the more traditional *semi-supervised classification* task that uses unlabelled data to improve classification accuracies [15, 20, 28]. In our case in fact, it is the labelled data that supports the generative task. Nevertheless, we will see in our experiment that CaGeM also leads to competitive semi-supervised classification performances.

To exploit the class information, we first set the number of clusters K equal to the number of classes C. We can now define two classifiers in CaGeM:

1. In the inference network we can compute the class probabilities given the data, i.e. $q_\phi(y|x)$, by integrating out the stochastic variables z_1 from $q_\phi(y, z_1|x) = q_\phi(y|z_1, x)q_\phi(z_1|x)$
2. Another set of class-probabilities can be computed using the generative model. Given the posterior distribution $p_\theta(z_2|x)$ we have in fact $p_\theta(y|x) = \int p_\theta(y|z_2)p_\theta(z_2|x)\mathrm{d}z_2$. The posterior over z_2 is intractable, but we can approximate it using the variational approximation $q_\phi(z_2|x)$, that is obtained by marginalizing out y and the variable z_1 in the joint distribution $q_\phi(y, z_1, z_2|x) = q_\phi(z_2|x, y, z_1)q_\phi(y|z_1, x)q_\phi(z_1|x)$. We can see this as a *cascade* of classifiers, as the class probabilities of the $p_\theta(y|x)$ classifier will depend on the probabilities of the classifier $q_\phi(y|z_1, x)$ in the inference model.

These two classifiers are used to define a new objective function for CaGeM:

$$\mathcal{I} = \sum_{\{x_u\}} \mathcal{F}(\theta, \phi) - \alpha \left(\sum_{\{x_l, y_l\}} (\mathcal{H}_p(\theta, \phi) + \mathcal{H}_q(\phi)) \right)$$

where $\{x_u\}$ is the set of unlabelled training points, $\{x_l, y_l\}$ is the set of labelled ones, and $\mathcal{H}_p$ and $\mathcal{H}_q$ are the standard categorical cross-entropies for the $p_\theta(y|x)$ and $q_\phi(y|x)$ classifiers respectively, that will only be optimized using labelled data points (see Appendix B for additional training details).

5 Results

We evaluate CaGeM by computing the generative log-likelihood performance on MNIST and OMNIGLOT [19] datasets.

[1]We witnessed a slight increase of 0.18 nats from using the skip connection.

The ELBO is evaluated by taking 5000 importance-weighted (IW) samples, denoted $\mathcal{F}_{5000}$. We evaluate the performance of CaGeM with different numbers of labelled samples referred to as CaGeM-#`labels`. When used, the labelled data is randomly sampled evenly across the class distribution. All experiments across datasets are run with the same architecture. Experimental details can be found in Appendix C.

	$\leq \log p(x)$
ADGM-100, L=2, IW=1 [20]	-86.06
IWAE, L=2, IW=1 [6]	-85.33
AVAE, L=2, IW=1 [20]	-82.97
IWAE, L=2, IW=50 [6]	-82.90
LVAE, L=5, IW=1 [29]	-82.12
LVAE, L=5, IW=10 [29]	-81.74
VAE+VGP, L=2 [31]	-81.32
DVAE [27]	-80.04
CaGeM-0, L=2, IW=1, K=20	-81.92
CaGeM-0, L=2, IW=1, K=5	-81.86
CaGeM-0, L=2, IW=1, K=10	-81.60
CaGeM-20, L=2, IW=1	-81.47
CaGeM-50, L=2, IW=1	-80.49
CaGeM-100, L=2, IW=1	-79.38

Table 1: Test log-likelihood on permutation invariant MNIST for L number of stochastic layers, IW importance weighted samples and K number clusters. Directly comparable models are IWAE and LVAE.

Table 1 shows the generative log-likelihood performances of different variants of CaGeM on the MNIST data set. We can see that the more labelled samples we use, the better the generative performance will be. Even though the results are not directly comparable, since CaGeM exploits a small fraction supervised information, we find that using only 100 labelled samples (10 samples per class), CaGeM-100 model achieves state of the art log-likelihood performance on permutation invariant MNIST with a simple 2-layered model. We also trained a ADGM-100 from [20][2] in order to make a fair comparison on generative log-likelihood in a semi-supervised setting and reached a performance of -86.06 nats. This indicates that models that are highly optimized for improving semi-supervised classification accuracy may be a suboptimal choice for generative modeling. The fully unsupervised CaGeM-0 results show that by defining clusters in the higher latent stochastic units, we achieve better performances than the closely related IWAE [6] and LVAE [29] models. It is also interesting to see from Table 1 that CaGeM-0 performs well even when the number of clusters are different from the number of classes in the labelled data set. CaGeM could further benefit from the usage of non-permutation invariant architectures suited for image data, such as the autoregressive decoders used by IAF VAE [16] and VLAE [7].

	$\leq \log p(x)$
VAE, L=2, IW=50 [6]	-106.30
IWAE, L=2, IW=50 [6]	-103.38
LVAE, L=5, FT, IW=10 [29]	-102.11
RBM [5]	-100.46
DBN [5]	-100.45
DVAE [27]	-97.43
CaGeM-500, L=2, IW=1	-100.86

Table 2: Generative test log-likelihood for permutation invariant OMNIGLOT. Directly comparable models are VAE, IWAE and LVAE.

The OMNIGLOT dataset consists of 50 different alphabets of handwritten characters, where each character is sparsely represented. In this task we use the alphabets as the cluster information, so that the z_2 representation should divide correspondingly. From Table 2 we see an improvement over other comparable VAE architectures (VAE, IWAE and LVAE), however, the performance is far from the once reported from the auto-regressive models [16, 7]. This indicates that the alphabet information is not as strong as for a dataset like MNIST.

Additional results visualizing the latent space learned by CaGeM and samples from the model for both MNIST and OMNIGLOT are available in Appendix D. Here, we also show that regardless of the fact that CaGeM was designed to optimize the semi-supervised generation task, the model preforms comparably to GANs [28] also for semi-supervised classification.

6 Conclusion

In this work we have shown how to perform semi-supervised generation with CaGeM. We showed that CaGeM improves the generative log-likelihood performance over similar deep generative approaches by creating clusters for the data in its higher latent representations using unlabelled information. CaGeM also provides a natural way to refine the clusters using additional labelled information to further improve its modelling power.

[2]We used the code from the first author publicly available in the repository named auxiliary-deep-generative-models on Github.

Acknowledgements

This research was supported by the Danish Innovation Foundation and NVIDIA Corporation with the donation of TITAN X GPUs. Marco Fraccaro is supported by Microsoft Research through its PhD Scholarship Programme.

References

[1] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. In *Deep Learning and Unsupervised Feature Learning, workshop at Neural Information Processing Systems*, 2012.

[2] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the International Conference on Machine Learning*, 2002.

[3] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 2013.

[4] S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[5] Y. Burda, R. Grosse, and R. Salakhutdinov. Accurate and conservative estimates of mrf log-likelihood using reverse annealing. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2015.

[6] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance Weighted Autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[7] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational Lossy Autoencoder. In *International Conference on Learning Representations*, 2017.

[8] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, A. van den Oord, and E. B. and. Lasagne: First release., Aug. 2015.

[9] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*. 2016.

[10] 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*. 2014.

[11] I. Gulrajani, K. Kumar, F. Ahmed, A. Ali Taiga, F. Visin, D. Vazquez, and A. Courville. PixelVAE: A latent variable model for natural images. *arXiv e-prints*, 1611.05013, Nov. 2016.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning*, 2015.

[13] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[14] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 12 2014.

[15] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-Supervised Learning with Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, 2014.

[16] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*. 2016.

[17] M. Kingma, Diederik P; Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 12 2013.

[18] R. G. Krishnan, U. Shalit, and D. Sontag. Deep Kalman filters. *arXiv:1511.05121*, 2015.

[19] B. M. Lake, R. R. Salakhutdinov, and J. Tenenbaum. One-shot learning by inverting a compositional causal process. In *Advances in Neural Information Processing Systems*. 2013.

[20] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, 2016.

[21] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, abs/1611.00712, 2016.

[22] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional Smoothing with Virtual Adversarial Training. *arXiv preprint arXiv:1507.00677*, 7 2015.

[23] R. Ranganath, D. Tran, and D. M. Blei. Hierarchical variational models. *arXiv preprint arXiv:1511.02386*, 11 2015.

[24] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, 2015.

[25] D. J. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the International Conference on Machine Learning*, 2015.

[26] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv preprint arXiv:1401.4082*, 04 2014.

[27] J. T. Rolfe. Discrete variational autoencoders. In *Proceedings of the International Conference on Learning Representations*, 2017.

[28] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[29] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems 29*. 2016.

[30] J. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[31] D. Tran, R. Ranganath, and D. M. Blei. Variational Gaussian process. In *Proceedings of the International Conference on Learning Representations*, 2016.

A The Problem of Inactive Units

A common problem encountered when training VAEs with bottom-up inference networks is given by the so called *inactive units* in the higher layers of stochastic variables resulting in poor generative performance [6, 29]. In a 2-layer model for example, VAEs often learn $q_\phi(z_2|z_1) = p(z_2) = \mathcal{N}(z_2; 0, I)$, i.e. the variational approximation of z_2 uses no information coming from the data point x through z_1. If we rewrite the ELBO in (1) as

$$\mathcal{F}(\theta,\phi) = \mathbb{E}_{q_\phi(z_1,z_2|x)}\left[\log\frac{p_\theta(x,z_1|z_2)}{q_\phi(z_1|x)}\right] - \mathbb{E}_{q_\phi(z_1|x)}\left[KL\left[q_\phi(z_2|z_1)||p(z_2)\right]\right]$$

we can see that $q_\phi(z_2|z_1) = p(z_2)$ represents a local maxima of our optimization problem where the KL-divergence term is set to zero and the information flows by first sampling in $\widetilde{z_1} \sim q_\phi(z_1|x)$ and then computing $p_\theta(x|\widetilde{z_1})$ (and is therefore independent from z_2). Several techniques have been developed in the literature to mitigate the problem of inactive units, among which we find annealing of the KL term [4, 29] or the use of free bits [16].

Using ideas from [7], we notice that the inactive units in a VAE with 2 layers of stochastic units can be justified not only as a poor local maxima, but also from the modelling point of view. [7] give a *bits-back coding* interpretation of Variational Inference for a generative model of the form $p(x,z) = p(x|z)p(z)$, with data x and stochastic units z. The paper shows that if the decoder $p(x|z)$ is powerful enough to explain most of the structure in the data (e.g. an autoregressive decoder), then it will be convenient for the model to set $q(z|x) = p(z)$ not to incur in an extra optimization cost of $KL[q(z|x)||p(z|x)]$. As we show in Section A.1, the inactive z_2 units in a 2-layer VAE can therefore be seen as caused by the flexible distribution $p_\theta(x, z_1|z_2)$ that is able to explain most of the structure in the data without using information from z_2. By making $q_\phi(z_2|z_1) = p(z_2)$, the model can avoid the extra cost of $KL\left[q_\phi(z_2|x)||p_\theta(z_2|x)\right]$.

It is now clear that if we want a VAE to exploit the power of additional stochastic layers we need to define it so that the benefits of encoding meaningful information in z_2 is greater than the cost $KL\left[q_\phi(z_2|x)||p_\theta(z_2|x)\right]$ that the model has to pay. This is achieved in the CaGeM, since the generative model is aided to do representation learning in the higher stochastic layers.

A.1 Inactive units in a 2-layer VAE

First consider a model $p(x)$ without latent units. We consider the asymptotic average properties, so we take the expectation of the log-likelihood over the (unknown) data distribution $p_{\text{data}}(x)$:

$$\begin{aligned}\mathbb{E}_{p_{\text{data}}(x)}\left[\log p(x)\right] &= \mathbb{E}_{p_{\text{data}}(x)}\left[\log\left(p_{\text{data}}(x)\frac{p(x)}{p_{\text{data}}(x)}\right)\right] \\ &= -\mathcal{H}(p_{\text{data}}) - KL(p_{\text{data}}(x)||p(x)) \ ,\end{aligned}$$

where $\mathcal{H}(p) = -\mathbb{E}_{p(x)}\left[\log p(x)\right]$ is the entropy of the distribution and $KL(\cdot||\cdot)$ is the KL-divergence. The expected log-likelihood is then simply the baseline entropy of the data generating distribution minus the deviation between the data generating distribution and our model for the distribution.

For the latent variable model $p_{\text{lat}}(x) = \int p(x|z)p(z)dz$ the log-likelihood bound is:

$$\log p_{\text{lat}}(x) \geq \mathbb{E}_{q(z|x)}\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right] \ .$$

We take the expectation over the data generating distribution and apply the same steps as above

$$\begin{aligned}\mathbb{E}_{p_{\text{data}}(x)}\left[\log p_{\text{lat}}(x)\right] &\geq \mathbb{E}_{p_{\text{data}}(x)}\mathbb{E}_{q(z|x)}\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right] \\ &= -\mathcal{H}(p_{\text{data}}) - KL(p_{\text{data}}(x)||p_{\text{lat}}(x)) - \mathbb{E}_{p_{\text{data}}(x)}\left[KL(q(z|x)||p(z|x))\right] \ ,\end{aligned}$$

where $p(z|x) = p(x|z)p(z)/p_{\text{lat}}(x)$ is the (intractable) posterior of the latent variable model. This results shows that we pay an additional price (the last term) for using an approximation to the posterior.

The latent variable model can choose to ignore the latent variables, $p(x|z) = \hat{p}(x)$. When this happens the expression falls back to the log-likelihood without latent variables. We can therefore get an (intractable) condition for when it is advantageous for the model to use the latent variables:

$$\mathbb{E}_{p_{\text{data}}(x)}\left[\log p_{\text{lat}}(x))\right] > \mathbb{E}_{p_{\text{data}}(x)}\left[\log \hat{p}(x))\right] + \mathbb{E}_{p_{\text{data}}(x)}\left[KL(q(z|x)||p(z|x))\right] \; .$$

The model will use latent variables when the log-likelihood gain is so high that it can compensate for the loss $KL(q(z|x)||p_{\text{lat}}(z|x))$ we pay by using an approximate posterior distribution.

The above argument can also be used to understand why it is harder to get additional layers of latent variables to become active. For a two-layer latent variable model $p(x, z_1, z_2) = p(x|z_1)p(z_1|z_2)p(z_2)$ we use a variational distribution $q(z_1, z_2|x) = q(z_2|z_1, x)q(z_1|x)$ and decompose the log likelihood bound using $p(x, z_1, z_2) = p(z_2|z_1, x)p(z_1|x)p_{\text{lat},2}(x)$:

$$\begin{aligned}\mathbb{E}_{p_{\text{data}}(x)}\left[\log p_{\text{lat},2}(x)\right] \geq \mathbb{E}_{p_{\text{data}}(x)}\mathbb{E}_{q(z_1,z_2|x)}&\left[\log \frac{p(x|z_1)p(z_1|z_2)p(z_2)}{q(z_1, z_2|x)}\right] \\ &= -H(p_{\text{data}}) - KL(p_{\text{data}}(x)||p_{\text{lat},2}(x)) \\ &\quad - \mathbb{E}_{p_{\text{data}}(x)}\mathbb{E}_{q(z_1|x)}\left[KL(q(z_2|z_1, x)||p(z_2|z_1, x))\right] \\ &\quad - \mathbb{E}_{p_{\text{data}}(x)}KL(q(z_1|x)||p(z_1|x)) \; .\end{aligned}$$

Again this expression falls back to the one-layer model when $p(z_1|z_2) = \hat{p}(z_1)$. So whether to use the second layer of stochastic units will depend upon the potential diminishing return in terms of log likelihood relative to the extra KL-cost from the approximate posterior.

B Objective function for semi-supervised generation

As our main goal is to learn representations that will lead to good generative performance, we interpret the classification of the additional labelled data as a secondary task that aids in learning a z_2 feature space that can be easily separated into clusters. We can then see this as a form of *semi-supervised clustering* [2], where we know that some data points belong to the same cluster and we are free to learn a data manifold that makes this possible.

The optimal features for the classification task could be very different from the representations learned for the generative task. This is why it is important not to update the parameters of the distributions over z_1, z_2 and x, in both generative model and inference model, using labelled data information. If this is not done carefully, the model could be prone to overfitting towards the labelled data. We define as θ_y the subset of θ containing the parameters in $p_\theta(y|z_2)$, and as ϕ_y the subset of ϕ containing the parameters in $q_\phi(y|z_1, x)$. θ_y and ϕ_y then represent the incoming arrows to y in Figure 1. We update the parameters θ and ϕ jointly by maximizing the new objective

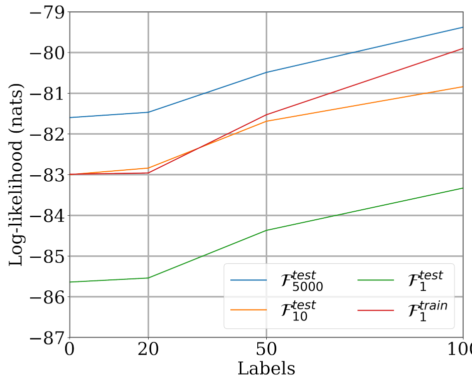
$$\mathcal{I} = \sum_{\{x_u\}} \mathcal{F}(\theta, \phi) - \alpha \left(\sum_{\{x_l, y_l\}} \left(\mathcal{H}_p(\theta_y, \phi_y) + \mathcal{H}_q(\phi_y)\right)\right)$$

where $\{x_u\}$ is the set of unlabelled training points, $\{x_l, y_l\}$ is the set of labelled ones, and $\mathcal{H}_p$ and $\mathcal{H}_q$ are the standard categorical cross-entropies for the $p_\theta(y|x)$ and $q_\phi(y|x)$ classifiers respectively. Notice that we consider the cross-entropies only a function of θ_y and ϕ_y, meaning that the gradients of the cross-entropies with respect to the parameters of the distributions over z_1, z_2 and x will be 0, and will not depend on the labelled data (as needed when learning meaningful representations of the data to be used for the generative task). To match the relative magnitudes between the ELBO $\mathcal{F}(\theta, \phi)$ and the two cross-entropies we set $\alpha = \beta \frac{N_u + N_l}{N_l}$ as done in [15, 20], where N_u and N_l are the numbers of unlabelled and labelled data points, and β is a scaling constant.

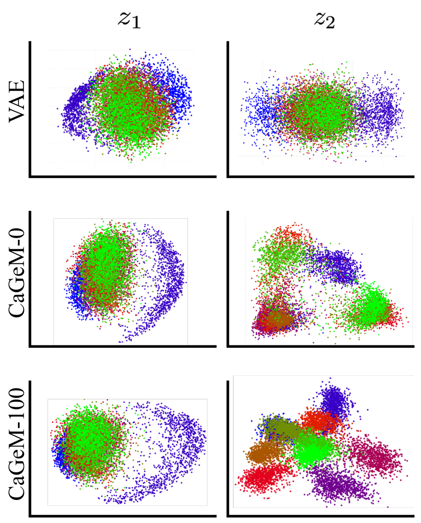
C Experimental details

The model is parameterized by feed-forward neural networks (`NN`) and linear layers (`Linear`), so that, for Gaussian outputs, each collection of incoming edges to a node in Figure 1 is defined as:

$$d = \texttt{NN}(x) \qquad \mu = \texttt{Linear}(d) \qquad \log \sigma^2 = \texttt{Linear}(d) \; .$$



(a) Log-likelihood scores for CaGeM on MNIST with 0, 20, 50 and 100 labels with 1, 10 and 5000 IW samples.



(b) PCA plots of the stochastic units z_1 and z_2 in a 2-layered model trained on MNIST. The colors corresponds to the true labels.

Figure 2

For Bernoulli distributed outputs we simply define a feed-forward neural network with a sigmoid activation function for the output. Between dense layers we use the rectified linear unit as non-linearity and batch-normalization [12]. We only collect statistics for the batch-normalization during unlabelled inference. For the log-likelihood experiments we apply temperature on the KL-terms during the first 100 epochs of training [4, 29]. The stochastic layers are defined with $\texttt{dim}(z_1) = 64$, $\texttt{dim}(z_2) = 32$ and 2-layered neural feed-forward networks with respectively 1024 and 512 units in each layer. Training is performed using the Adam optimizer [14] with an initial learning rate of 0.001 and annealing it by $.75$ every 50 epochs. β is set to 0.1 as in [20]. The input data is dynamically binarized. The experiments are implemented with Theano [1], Lasagne [8] and Parmesan[3].

D Additional results

In Figure 2a we show in detail how the performance of CaGeM increases as we add more labelled data points. We can also see that the ELBO $\mathcal{F}_1^{test}$ tightens when adding more labelled information, as compared to $\mathcal{F}_1^{\text{LVAE}} = -85.23$ and $\mathcal{F}_1^{\text{VAE}} = -87.49$ [29].

The PCA plots of the z_2 variable of a VAE, CaGeM-0 and CaGeM-100 are shown in Figure 2b. We see how CaGeMs encode clustered information into the higher stochastic layer. Since CaGeM-0 is unsupervised, it forms less class-dependent clusters compared to the semi-supervised CaGeM-100, that fits its z_2 latent space into 10 nicely separated clusters. Regardless of the labelled information added during inference, CaGeM manages to activate a high amount of units, as for CaGeM we obtain $KL[q_\phi(z_2|x,y)||p(z_2)] \approx 17$ nats, while a LVAE with 2 stochastic layers obtains ≈ 9 nats.

The generative model in CaGeM enables both random samples, by sampling the class variable $y \sim p_\theta(y|z_2)$ and feeding it to $p_\theta(x|z_1, y)$, and class conditional samples by fixing y. Figure 3 shows the generation of MNIST digits from CaGeM-100 with $\texttt{dim}(z_2) = 2$. The images are generated by applying a linearly spaced mesh grid within the latent space z_2 and performing random generations (left) and conditional generations (right). When generating samples in CaGeM, it is clear how the latent units z_1 and z_2 capture different modalities within the true data distribution, namely style and class. This also explains the lack of clustering in z_1 from Figure 2b. Reconstructions and samples of OMNIGLOT characters from the model can be found in Figure 4.

Regardless of the fact that CaGeM was designed to optimize the semi-supervised generation task, the model can also be used for classification by using the classifier $p_\theta(y|x)$. In Table 3 we show that the

[3]A variational repository named parmesan on Github.

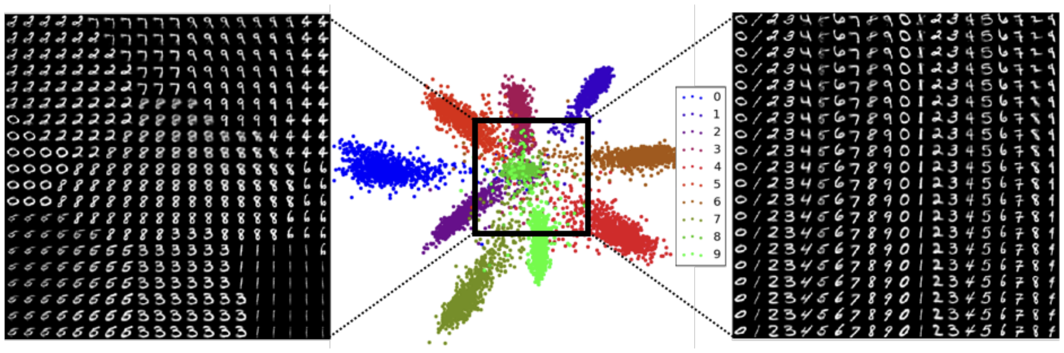


Figure 3: MNIST visualizations from CaGeM-100 with a 2-dimensional z_2 space. The middle plot shows the latent space, from which we generate random samples (left) and class conditional random samples (right) with a mesh grid (black bounding box). The relative placement of the samples in the scatter plot corresponds to a digit in the mesh grid.

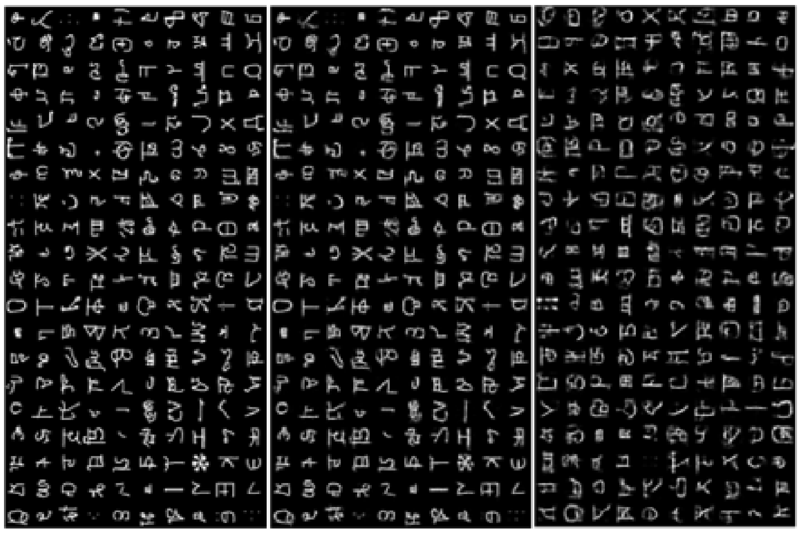


Figure 4: Generations for the OMNIGLOT data set from CaGeM-500. (left) The input images, (middle) the reconstructions, and (right) random samples from z_2.

semi-supervised classification accuracies obtained with CaGeM are comparable to the performance of GANs [28].

LABELS	20	50	100
M1+M2 [15]	-	-	3.33% (±0.14)
VAT [22]	-	-	2.12%
CATGAN [30]	-	-	1.91% (±0.1)
SDGM [20]	-	-	1.32% (±0.07)
LADDER NETWORK [24]	-	-	1.06% (±0.37)
ADGM [20]	-	-	0.96% (±0.02)
IMP. GAN [28]	16.77% (±4.52)	2.21% (±1.36)	0.93% (±0.65)
CAGEM	15.86%	2.42%	1.16%

Table 3: Semi-supervised test error % benchmarks on MNIST for 20, 50, and 100 randomly chosen and evenly distributed labelled samples. Each experiment was run 3 times with different labelled subsets and the reported accuracy is the mean value.

E Related Work

As we have seen from our experiments, CaGeM offers a way to exploit the added flexibility of a second layer of stochastic units, that stays active as the modeling performances can greatly benefit from capturing the natural clustering of the data. Other recent works have presented alternative methods to mitigate the problem of inactive units when training flexible models defined by a hierarchy of stochastic layers. [6] used importance samples to improve the tightness of the ELBO, and showed that this new training objective helped in activating the units of a 2-layer VAE. [29] trained Ladder Variational Autoencoders (LVAE) composed of up to 5 layers of stochastic units, using a top-down inference network that forces the information to flow in the higher stochastic layers. Contrarily to the bottom-up inference network of CaGeM, the top-down approach used in LVAEs does not enforce a clear separation between the role of each stochastic unit, as proven by the fact that all of them encode some class information. Longer hierarchies of stochastic units unrolled in time can be found in the sequential setting [18, 9]. In these applications the problem of inactive stochastic units appears when using powerful autoregressive decoders [9, 7], but is mitigated by the fact that new data information enters the model at each time step.

The discrete variable y of CaGeM was introduced to be able to define a better learnable representation of the data, that helps in activating the higher stochastic layer. The combination of discrete and continuous variables for deep generative models was also recently explored by several authors. [21, 13] used a continuous relaxation of the discrete variables, that makes it possible to efficiently train the model using stochastic backpropagation. The introduced Gumbel-Softmax variables allow to sacrifice log-likelihood performances to avoid the computationally expensive integration over y. [27] presents a new class of probabilistic models that combines an undirected component consisting of a bipartite Boltzmann machine with binary units and a directed component with multiple layers of continuous variables.

Traditionally, semi-supervised learning applications of deep generative models such as Variational Auto-encoders and Generative Adversarial Networks [10] have shown that, whenever only a small fraction of labelled data is available, the supervised classification task can benefit from additional unlabelled data [15, 20, 28]. [20, 15] are indeed very similar approaches to CaGeM, besides the fact that they solely treat the classifier as part of the Q model resulting in relatively poor generative performance. In this work we consider the semi-supervised problem from a different perspective, and show that the generative task of CaGeM can benefit from additional labelled data. As a by-product of our model, we also obtain competitive semi-supervised classification results, meaning that CaGeM is able to share statistical strength between the generative and classification tasks.

When modeling natural images, the performance of CaGeM could be further improved using more powerful autoregressive decoders such as the ones in [11, 7]. Also, an even more flexible variational approximation could be obtained using auxiliary variables [23, 20] or normalizing flows [25, 16].