# Distilling Intractable Generative Models

**George Papamakarios**
School of Informatics
University of Edinburgh
g.papamakarios@ed.ac.uk

**Iain Murray**
School of Informatics
University of Edinburgh
i.murray@ed.ac.uk

## Abstract

A generative model's partition function is typically expressed as an intractable multi-dimensional integral, whose approximation presents a challenge to numerical and Monte Carlo integration. In this work, we propose a new estimation method for intractable partition functions, based on distilling an intractable generative model into a tractable approximation thereof, and using the latter for proposing Monte Carlo samples. We empirically demonstrate that our method produces state-of-the-art estimates, even in combination with simple Monte Carlo methods.

## 1 Introduction

A generative model is a probability distribution $p(\mathbf{x}) = \bar{p}(\mathbf{x})/Z$ over a multi-dimensional random variable $\mathbf{x}$, where $\bar{p}(\mathbf{x})$ is a non-negative potential and $Z = \int \bar{p}(\mathbf{x})\,\mathrm{d}\mathbf{x}$ is the partition function. Apart from ensuring that $p(\mathbf{x})$ is properly normalized, $Z$ is a quantity of interest in its own right; in Bayesian inference [1], it is the marginal likelihood of the data, and as such it can be used for assessing model fit and comparing models; in undirected graphical models [2], useful properties of the model can be obtained from the derivatives of $\log Z$ w.r.t. the model parameters—for models in the exponential family, these derivatives correspond to the expected sufficient statistics under the model.

However, $Z$ is given by a multi-dimensional integral, which in all but the most trivial cases is intractable to evaluate, even when the potential $\bar{p}(\mathbf{x})$ is tractable. Numerical integration can only be used to approximate $Z$ when $\mathbf{x}$ is low-dimensional, as it scales badly with dimensionality. Monte Carlo methods [3, 4, 5] typically avoid the curse of dimensionality, but are often inaccurate or slow. Variational inference [6] is a way to strictly lower-bound $Z$, but such lower bounds are often loose.

In this work, we propose a new method for estimating $Z$, based on model distillation. The term "distillation" was pioneered by Hinton et al. [7] in the context of discriminative models, based on earlier work by Bucilă et al. [8] on model compression. We propose a novel distillation framework for generative models, whereby an intractable generative model is distilled into a tractable one, by training the latter to mimic the former as closely as possible. We then use the distilled tractable model in combination with simple Monte Carlo, to yield estimates that are as robust as those of state-of-the-art Monte Carlo methods. We showcase our framework by distilling an intractable Restricted Boltzmann Machine into a tractable Neural Autoregressive Distribution Estimator, and then using the distilled NADE to robustly estimate the RBM's intractable partition function.

This paper forms part of our more general work on "distilling model knowledge", which can be found in [9]. Chapter 4 of [9] contains further details, results and discussion on the work presented here, and discusses the links between model distillation and alternatives such as variational inference.

## 2 The distillation framework

Assume we can construct a tractable generative model $q_{\boldsymbol{\theta}}(\mathbf{x})$, parameterized by $\boldsymbol{\theta}$, which is flexible enough to represent complex distributions. Our goal is to distil $p(\mathbf{x}) = \bar{p}(\mathbf{x})/Z$ into $q_{\boldsymbol{\theta}}(\mathbf{x})$; that is, to

train $q_{\boldsymbol{\theta}}(\mathbf{x})$ to mimic $p(\mathbf{x})$ as closely as possible. We assume that the potential $\bar{p}(\mathbf{x})$ is tractable, but the partition function $Z$ and hence $p(\mathbf{x})$ are not. Our approach is based on minimizing an appropriate loss function $E(\boldsymbol{\theta})$ that measures the discrepancy between $p(\mathbf{x})$ and $q_{\boldsymbol{\theta}}(\mathbf{x})$ w.r.t. $\boldsymbol{\theta}$. The challenge is to specify a loss function that only involves tractable quantities but also captures as much information about $p(\mathbf{x})$ as possible, and find an efficient way to minimize it. In the following, we describe two loss functions that achieve this goal, together with a stochastic method of minimizing them.

**KL divergence.** A natural measure of discrepancy between distributions is the KL divergence from $p(\mathbf{x})$ to $q_{\boldsymbol{\theta}}(\mathbf{x})$, which is defined as

$$E_{\mathrm{KL}}(\boldsymbol{\theta}) = D_{\mathrm{KL}}(p(\mathbf{x}) \,\|\, q_{\boldsymbol{\theta}}(\mathbf{x})) = \langle \log p(\mathbf{x}) \rangle_{p(\mathbf{x})} - \langle \log q_{\boldsymbol{\theta}}(\mathbf{x}) \rangle_{p(\mathbf{x})} . \tag{1}$$

It is well known that the KL divergence is non-negative, and is equal to zero if and only if $p(\mathbf{x}) = q_{\boldsymbol{\theta}}(\mathbf{x})$ [1, section 2.6]. Thus, if $q_{\boldsymbol{\theta}}(\mathbf{x})$ is capable of representing the distribution defined by $p(\mathbf{x})$, globally minimizing $E_{\mathrm{KL}}(\boldsymbol{\theta})$ will result in $q_{\boldsymbol{\theta}}(\mathbf{x})$ exactly matching $p(\mathbf{x})$.

It is easy to see that minimizing $E_{\mathrm{KL}}(\boldsymbol{\theta})$ is equivalent to maximizing $\langle \log q_{\boldsymbol{\theta}}(\mathbf{x}) \rangle_{p(\mathbf{x})}$. This can be viewed as fitting $q_{\boldsymbol{\theta}}(\mathbf{x})$ using maximum likelihood to an infinite amount of data sampled from $p(\mathbf{x})$. This loss function is also used within expectation propagation updates [10]; variational inference [6] and its stochastic versions [11, 12] use the "reverse" KL divergence $D_{\mathrm{KL}}(q_{\boldsymbol{\theta}}(\mathbf{x}) \,\|\, p(\mathbf{x}))$ instead.

**Square error.** Another way of measuring discrepancy is to directly measure how much the values of $q_{\boldsymbol{\theta}}(\mathbf{x})$ differ from the values of $p(\mathbf{x})$. A natural choice is the average square error of the logs

$$E^0_{\mathrm{SE}}(\boldsymbol{\theta}) = \left\langle \frac{1}{2} \left\| \log q_{\boldsymbol{\theta}}(\mathbf{x}) - \log p(\mathbf{x}) \right\|^2 \right\rangle_{p(\mathbf{x})} . \tag{2}$$

Obviously, directly working with the above is not possible, since we have assumed that directly evaluating $\log p(\mathbf{x})$ is intractable. However, it is possible to circumvent this problem by using the following tractable loss function instead

$$E_{\mathrm{SE}}(\boldsymbol{\theta}) = \left\langle \frac{1}{2} \left\| \log q_{\boldsymbol{\theta}}(\mathbf{x}) - \log \bar{p}(\mathbf{x}) + c \right\|^2 \right\rangle_{p(\mathbf{x})} , \tag{3}$$

where $c$ is an appropriately chosen constant. Proposition 1 (proof cam be found in [9]) establishes the range of $c$ values for which minimizing $E_{\mathrm{SE}}(\boldsymbol{\theta})$ correctly trains $q_{\boldsymbol{\theta}}(\mathbf{x})$ to match $p(\mathbf{x})$.

**Proposition 1.** *Assuming $q_{\boldsymbol{\theta}}(\mathbf{x})$ is capable of representing the distribution defined by $p(\mathbf{x})$ and $c \leq \log Z$, then $E_{\mathrm{SE}}(\boldsymbol{\theta})$ is globally minimized if and only if $q_{\boldsymbol{\theta}}(\mathbf{x}) = p(\mathbf{x})$.*

In order to set $c$, we do not necessarily need to know $\log Z$, but we only need to lower bound it. If $\mathbf{x}$ is discrete, this becomes trivial since $\log \bar{p}(\mathbf{x}) \leq \log Z$ for any $\mathbf{x}$. Thus, setting $c$ less or equal to $\max_{\mathbf{x}} \log \bar{p}(\mathbf{x})$ is guaranteed to work. More generally, lower bounding $\log Z$ can be done using the variational free energy $\langle \log \bar{p}(\mathbf{x}) \rangle_{r(\mathbf{x})} - \langle \log r(\mathbf{x}) \rangle_{r(\mathbf{x})} \leq \log Z$ [6]. This inequality holds for any distribution $r(\mathbf{x})$, so $r(\mathbf{x})$ can be freely chosen to be convenient. Finally, it is easy to show that in the limit of $c \to -\infty$, minimizing $E_{\mathrm{SE}}(\boldsymbol{\theta})$ becomes equivalent to minimizing $E_{\mathrm{KL}}(\boldsymbol{\theta})$.

**Stochastic gradient training.** Both loss functions discussed above exhibit a common pattern; they contain a tractable quantity within an intractable expectation over $p(\mathbf{x})$. That is, they are both of the form $E(\boldsymbol{\theta}) = \langle E(\mathbf{x}, \boldsymbol{\theta}) \rangle_{p(\mathbf{x})}$ for an appropriate instantiation of $E(\mathbf{x}, \boldsymbol{\theta})$. We will now describe a stochastic gradient learning procedure for optimizing loss functions of the above form. The gradient of $E(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$, albeit intractable, can be stochastically approximated as follows

$$\mathbf{g}(\boldsymbol{\theta}) = \left\langle \frac{\partial}{\partial \boldsymbol{\theta}} E(\mathbf{x}, \boldsymbol{\theta}) \right\rangle_{p(\mathbf{x})} \approx \frac{1}{S} \sum_s \frac{\partial}{\partial \boldsymbol{\theta}} E(\mathbf{x}_s, \boldsymbol{\theta}), \tag{4}$$

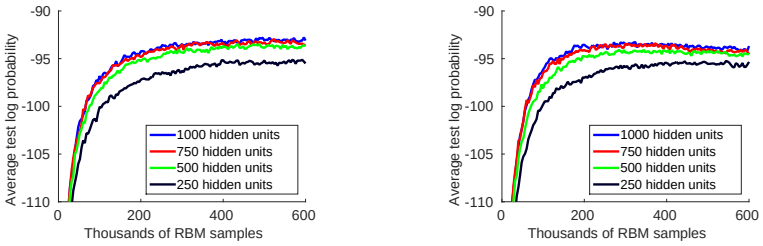
where $\{\mathbf{x}_s\}$ are samples generated from $p(\mathbf{x})$. Typically, we can generate $\{\mathbf{x}_s\}$ using Markov Chain Monte Carlo. It is easy to see that, in expectation, the stochastic gradient is equal to the original gradient. Using the stochastic gradient, we can minimize $E(\boldsymbol{\theta})$ by iterating the following three steps.

(i) Generate a minibatch $\{\mathbf{x}_s\}$ of size $S$ from $p(\mathbf{x})$ using MCMC.

(ii) Calculate the stochastic gradient $\hat{\mathbf{g}}(\boldsymbol{\theta}) = \frac{1}{S} \sum_s \frac{\partial}{\partial \boldsymbol{\theta}} E(\mathbf{x}_s, \boldsymbol{\theta})$.

(iii) Make an update on $\boldsymbol{\theta}$ using $\hat{\mathbf{g}}(\boldsymbol{\theta})$.

It is known that stochastic optimization algorithms of the above type—under certain regularity conditions—converge almost surely to a stationary point of the objective function [13, 14].




(a) Trained with KL divergence

(b) Trained with square error

Figure 1: Training progress for each NADE, as measured by average log probability of the first 500 images of the MNIST test set. The training progress was measured every 200 iterations.

## 3 Distilling an RBM into NADE

The Restricted Boltzmann Machine [15] is a bipartite undirected graphical model, consisting of a layer of binary visible variables and a layer of binary hidden variables. Calculating the partition function of the RBM is intractable, since it involves a summation over an exponential number of hidden variable states. Exact sampling from the RBM is also hard, even though approximate samples can be easily drawn using block Gibbs sampling. In our experiments, we used an RBM with 500 hidden variables that was trained on a binarized version of the MNIST dataset of handwritten digits [16]. This RBM was provided by Salakhutdinov and Murray [3], who referred to it as CD25(500).

The Neural Autoregressive Distribution Estimator [17] is a generative model for binary data that is flexible enough to model complex distributions. NADE is an autoregressive neural network with a single hidden layer. It is fully tractable, in that its likelihood and its derivatives can be easily calculated using forward and backward propagation, and exact samples from it can be easily generated with ancestral sampling. In our experiments, we used NADEs with 250, 500, 750 and 1000 hidden units.

Using our distillation framework, we distilled the RBM into each of the four NADEs. During stochastic gradient training, we used minibatches of 20 samples from the RBM, generated by block Gibbs sampling with parallel chains. In particular, we maintained 2000 parallel Markov chains and, in each iteration, we simulated each chain once and selected 20 of them (in sequence) to form the minibatch. This way, all samples within a minibatch are independent and each chain is thinned $2000/20 = 100$ times before contributing a sample. Empirically, we found that having independent samples within the minibatch and thinning across minibatches improved learning. The learning rate was adapted using ADADELTA [18], which uses a different learning rate for each parameter and is fairly easy to tune. Stochastic gradient training was run for 30,000 iterations in total.

Figure 1 shows the progress of each NADE during training, as measured by the average log probability each NADE assigns to the first 500 MNIST test images. We can see that more hidden units lead to a higher log probability, with a significant difference between the NADEs with 250 and 500 hidden units. This suggests that at least 500 hidden units are needed for NADE to have enough flexibility to accurately mimic the RBM. Further results and relevant discussion are provided in [9].

## 4 Estimating the partition function

Salakhutdinov and Murray [3] provided state-of-the-art estimates for the above RBM's partition function, using annealed importance sampling [4] with 10,000 intermediate distributions. Here, we show that, having access to a distilled NADE, we can achieve estimates that are as good as theirs with simpler Monte Carlo methods, such as importance sampling and bridge sampling.

**Importance sampling.** Given a tractable proposal distribution $q_{\boldsymbol{\theta}}(\mathbf{x})$ that is non-zero wherever $p(\mathbf{x})$ is non-zero, importance sampling [1, section 29.2] rewrites the partition function as an expectation over $q_{\boldsymbol{\theta}}(\mathbf{x})$ and then stochastically approximates it as follows

$$Z = \sum_{\mathbf{x}} \bar{p}(\mathbf{x}) = \sum_{\mathbf{x}} \frac{\bar{p}(\mathbf{x})}{q_{\boldsymbol{\theta}}(\mathbf{x})} q_{\boldsymbol{\theta}}(\mathbf{x}) = \left\langle \frac{\bar{p}(\mathbf{x})}{q_{\boldsymbol{\theta}}(\mathbf{x})} \right\rangle_{q_{\boldsymbol{\theta}}(\mathbf{x})} \approx \frac{1}{S} \sum_{s} \frac{\bar{p}(\mathbf{x}_s)}{q_{\boldsymbol{\theta}}(\mathbf{x}_s)} \quad \text{with} \quad \mathbf{x}_s \sim q_{\boldsymbol{\theta}}(\mathbf{x}_s). \quad (5)$$

| Sampling method | Hidden units | Loss function | |
|---|---|---|---|
| | | KL divergence | Square error |
| Importance sampling | 250 | $450.86\ (450.06, 451.30)$ | $450.81\ (449.77, 451.31)$ |
| | 500 | $450.96\ (449.94, 451.46)$ | $450.96\ (450.33, 451.34)$ |
| | 750 | $450.79\ (450.47, 451.04)$ | $450.95\ (450.40, 451.30)$ |
| | 1000 | $450.79\ (450.40, 451.07)$ | $451.51\ (450.09, 452.08)$ |
| Bridge sampling | 250 | $451.27\ (451.14, 451.39)$ | $451.24\ (451.10, 451.38)$ |
| | 500 | $451.28\ (451.16, 451.39)$ | $451.30\ (451.17, 451.43)$ |
| | 750 | $451.22\ (451.11, 451.33)$ | $451.21\ (451.08, 451.33)$ |
| | 1000 | $451.24\ (451.13, 451.35)$ | $451.36\ (451.23, 451.49)$ |

Table 1: Estimates of RBM's log partition function, using each trained NADE as proposal distribution. Brackets show confidence intervals of 3 standard deviations.

For the above approximation to work well, $q_{\boldsymbol{\theta}}(\mathbf{x})$ must be as similar to $p(\mathbf{x})$ as possible. In our framework, NADEs that were trained by minimizing $D_{\mathrm{KL}}(p(\mathbf{x}) \,\|\, q_{\boldsymbol{\theta}}(\mathbf{x}))$ will tend to mimic closely the RBM and at the same time avoid at all costs not putting mass where the RBM has mass. Hence, such NADEs align well with the requirements importance sampling has on its proposal distributions.

**Bridge sampling.** Importance sampling tends to fail when the proposal distribution has little mass in some regions where the target distribution has significant mass. Bridge sampling [5, section 1] is an improvement over importance sampling that tries to alleviate this problem. Bridge sampling uses a "bridge distribution" $p_b(\mathbf{x}) = \bar{p}_b(\mathbf{x})/Z_b$, which is constructed to be the overlap between $p(\mathbf{x})$ and $q_{\boldsymbol{\theta}}(\mathbf{x})$. Then, using standard importance sampling, the following two quantities are estimated

$$Z_b = \left\langle \frac{\bar{p}_b(\mathbf{x})}{q_{\boldsymbol{\theta}}(\mathbf{x})} \right\rangle_{q_{\boldsymbol{\theta}}(\mathbf{x})} \quad \text{and} \quad \frac{Z_b}{Z} = \left\langle \frac{\bar{p}_b(\mathbf{x})}{\bar{p}(\mathbf{x})} \right\rangle_{p(\mathbf{x})}, \quad \text{with} \quad \bar{p}_b(\mathbf{x}) = \frac{q_{\boldsymbol{\theta}}(\mathbf{x})\bar{p}(\mathbf{x})}{C q_{\boldsymbol{\theta}}(\mathbf{x}) + \bar{p}(\mathbf{x})}, \tag{6}$$

where $C$ is a user-specified constant. By construction, regions where $p_b(\mathbf{x})$ has significant mass are also significant under $p(\mathbf{x})$ and $q_{\boldsymbol{\theta}}(\mathbf{x})$, hence importance sampling works well in this case. Having estimated $Z_b$ and $Z_b/Z$, an estimate for $Z$ can be trivially given by their ratio.

With the same number of samples from $p(\mathbf{x})$ and $q_{\boldsymbol{\theta}}(\mathbf{x})$, $p_b(\mathbf{x})$ is the asymptotically optimal bridge distribution when $C = Z$ [19]. However, $Z$ is what the method tries to estimate. Thus, the method can be set up as a fixed point system; initialize $C$ (e.g. to 1), calculate $\bar{p}_b(\mathbf{x})$, set $C$ to the resulting estimate of $Z$, and repeat. In practice we found that a few iterations (e.g. 10) suffice for convergence.

We used bridge sampling with the distilled NADEs as proposal distributions. In order to obtain good quality samples from the RBM, we took advantage of the fact that NADE samples are similar to RBM samples, and, after sampling from NADE, we initialized parallel RBM Gibbs chains with them.

**Results.** Table 1 shows the estimates of $\log Z$ obtained by using each trained NADE as proposal distribution with both Monte Carlo methods. We used $10{,}000$ exact samples from NADE for each estimate. For comparison, Salakhutdinov and Murray [3] report an estimate of $451.28$, with a $3$ standard deviation confidence interval of $(450.97, 451.52)$. Compared to this, importance sampling slightly underestimates $\log Z$. On the other hand, bridge sampling with NADE provides accurate and reliable estimates of $\log Z$, and performs comparably to the more sophisticated method of [3].

## 5 Conclusions

In this work we have demonstrated that proposal distributions constructed by model distillation can significantly improve the performance of simple Monte Carlo methods in estimating intractable partition functions. Our experiments focused on partition functions that are given by an intractable summation over an exponential number of states, rather than by a continuous integral. Nevertheless, the framework is equally applicable to the continuous case as well, where the real-valued version of NADE (i.e. RNADE [20]) can serve as the distilled tractable model.

**Acknowledgments**

George Papamakarios was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh, and by Microsoft Research through its PhD Scholarship Programme. The authors would like to thank the anonymous reviewers for their constructive comments.

## References

[1] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002.

[2] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[3] Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of Deep Belief Networks. In *Proceedings of the 25th International Conference on Machine Learning*, pages 872–879, 2008. Code and data available from `http://www.utstat.toronto.edu/~rsalakhu/rbm_ais.html`.

[4] Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

[5] Radford M. Neal. Estimating ratios of normalizing constants using linked importance sampling. Technical Report No. 0511, Department of Statistics, University of Toronto, 2005. arXiv:math/0511216v1.

[6] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

[7] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531v1*, 2015.

[8] Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–541, 2006.

[9] George Papamakarios. *Distilling Model Knowledge*. MScR thesis, Centre for Doctoral Training in Data Science, University of Edinburgh, 2015. Available from `http://arxiv.org/abs/1510.02437`.

[10] Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 362–369, 2001.

[11] Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286, 2014.

[12] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, 2013.

[13] Léon Bottou. On-line learning and stochastic approximations. In *On-line Learning in Neural Networks*, pages 9–42. Cambridge University Press, 1998.

[14] Laurent Younes. On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. In *Stochastics and Stochastics Models*, pages 177–228, 1999.

[15] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 194–281. MIT Press, 1986.

[16] Yann Le Cun, Corinna Cortes, and Christopher J. C. Burges. MNIST handwritten digit database. `http://yann.lecun.com/exdb/mnist/`. Accessed on 20 July 2015.

[17] Hugo Larochelle and Iain Murray. The Neural Autoregressive Distribution Estimator. *JMLR: W&CP*, 15: 29–37, 2011.

[18] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv:1212.5701v1*, 2012.

[19] Xiao-Li Meng and Wing Hung Wong. Simulating ratios of normalizing constants via a simple identity: A theoretical exploration. *Statistica Sinica*, 6:831–860, 1996.

[20] Benigno Uria, Iain Murray, and Hugo Larochelle. RNADE: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems 26*, pages 2175–2183. 2013.