Variational Memory Addressing in Generative Models

Authored by:

Andriy Mnih
Daniel Zoran
Danilo Jimenez Rezende
J?rg Bornschein

Abstract

Aiming to augment generative models with external memory, we interpret the output of a memory module with stochastic addressing as a conditional mixture distribution, where a read operation corresponds to sampling a discrete memory address and retrieving the corresponding content from memory. This perspective allows us to apply variational inference to memory addressing, which enables effective training of the memory module by using the target information to guide memory lookups. Stochastic addressing is particularly well-suited for generative models as it naturally encourages multimodality which is a prominent aspect of most high-dimensional datasets. Treating the chosen address as a latent variable also allows us to quantify the amount of information gained with a memory lookup and measure the contribution of the memory module to the generative process. To illustrate the advantages of this approach we incorporate it into a variational autoencoder and apply the resulting model to the task of generative few-shot learning. The intuition behind this architecture is that the memory module can pick a relevant template from memory and the continuous part of the model can concentrate on modeling remaining variations. We demonstrate empirically that our model is able to identify and access the relevant memory contents even with hundreds of unseen Omniglot characters in memory.

1 Paper Body

Recent years have seen rapid developments in generative modelling. Much of the progress was driven by the use of powerful neural networks to parameterize conditional distributions composed to define the generative process (e.g., VAEs [1, 2], GANs [3]). In the Variational Autoencoder (VAE) framework for example, we typically define a generative model p(z), p? (x—z) and an approximate inference model q (z—x). All conditional distributions are parameterized

by multilayered perceptrons (MLPs) which, in the simplest case, output the mean and the diagonal variance of a Normal distribution given the conditioning variables. We then optimize a variational lower bound to learn the generative model for x. Considering recent progress, we now have the theory and the tools to train powerful, potentially non-factorial parametric conditional distributions p(x—y) that generalize well with respect to x (normalizing flows [4], inverse autoregressive flows [5], etc.). Another line of work which has been gaining popularity recently is memory augmented neural networks [6, 7, 8]. In this family of models the network is augmented with a memory buffer which allows read and write operations and is persistent in time. Such models usually handle input and output to the memory buffer using differentiable ?soft? write/read operations to allow back-propagating gradients during training. Here we propose a memory-augmented generative model that uses a discrete latent variable a acting as an address into the memory buffer M. This stochastic perspective allows us to introduce a variational approximation over the addressing variable which takes advantage of target information 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

z
m
p(z)
p(m) p(a)
m
x
z
p(x—z, m(z))
x
a
m
z
p(x—m, z)
x
p(x—ma , z)

Figure 1: Left: Sketch of typical SOTA generative latent variable model with memory. Red edges indicate approximate inference distributions q(?—?). The KL(q——p) cost to identify a specific memory entry might be substantial, even though the cost of accessing a memory entry should be in the order of log —M—. Middle & Right: We combine a top-level categorical distribution p(a) and a conditional variational autoencoder with a Gaussian p(z—m). when retrieving contents from memory during training. We compute the sampling distribution over the addresses based on a learned similarity measure between the memory contents at each address and the target. The memory contents ma at the selected address a serve as a context for a continuous latent variable z, which together with ma is used to generate the target observation. We therefore interpret memory as a non-parametric conditional mixture distribution. It is non-parametric in the sense that we can change the content and the size of the memory from one evaluation of the model to another without having to relearn

the model parameters. And since the retrieved content ma is dependent on the stochastic variable a, which is part of the generative model, we can directly use it downstream to generate the observation x. These two properties set our model apart from other work on VAEs with mixture priors [9, 10] aimed at unconditional density modelling. Another distinguishing feature of our approach is that we perform sampling-based variational inference on the mixing variable instead of integrating it out as is done in prior work, which is essential for scaling to a large number of memory addresses. Most existing memory-augmented generative models use soft attention with the weights dependent on the continuous latent variable to access the memory. This does not provide clean separation between inferring the address to access in memory and the latent factors of variation that account for the variability of the observation relative to the memory contents (see Figure 1). Or, alternatively, when the attention weights depend deterministically on the encoder, the retrieved memory content can not be directly used in the decoder. Our contributions in this paper are threefold: a) We interpret memory-read operations as conditional mixture distribution and use amortized variational inference for training; b) demonstrate that we can combine discrete memory addressing variables with continuous latent variables to build powerful models for generative few-shot learning that scale gracefully with the number of items in memory; and c) demonstrate that the KL divergence over the discrete variable a serves as a useful measure to monitor memory usage during inference and training.

2

Model and Training

We will now describe the proposed model along with the variational inference procedure we use to train it. The generative model has the form Z X p(x—M) = p(a—M) p(z—ma) p(x—z, ma) dz (1) a

z

where x is the observation we wish to model, a is the addressing categorical latent variable, z the continuous latent vector, M the memory buffer and ma the memory contents at the ath address.

The generative process proceeds by first sampling an address a from the categorical distribution p(a—M), retrieving the contents ma from the memory buffer M, and then sampling the observation x from a conditional variational auto-encoder with ma as the context conditioned on (Figure 1, B). 2

The intuition here is that if the memory buffer contains a set of templates, a trained model of this type should be able to produce observations by distorting a template retrieved from a randomly sampled memory location using the conditional variational autoencoder to account for the remaining variability. We can write the variational lower bound for the model in (1): log p(x—M)

E

a,z?q(?—M,x)

[log p(x, z, a—M)

log q(a, z—M, x)]

where q(a, z—M, x) = q(a—M, x)q(z—ma , x).

(2) (3)

In the rest of the paper, we omit the dependence on M for brevity. We will now describe the components of the model and the variational posterior (3) in detail. The first component of the model is the memory buffer M. We here do not implement an explicit write operation but consider two possible sources for the memory content: Learned memory: In generative experiments aimed at better understanding the model?s behaviour we treat M as model parameters. That is we initialize M randomly and update its values using the gradient of the objective. Few-shot learning: In the generative few-shot learning experiments, before processing each minibatch, we sample —M— entries from the training data and store them in their raw (pixel) form in M. We ensure that the training minibatch {x1 , ..., x—B— } contains disjoint samples from the same character classes, so that the model can use M to find suitable templates for each target x. The second component is the addressing variable a 2 {1, ..., —M—} which selects a memory entry ma from the memory buffer M. The varitional posterior distribution q(a—x) is parameterized as a softmax over a similarity measure between x and each of the memory entries ma : q

q (a—x) / exp Sq (ma , x),

(4)

where S (x, y) is a learned similarity function described in more detail below. Given a sample a from the posterior q (a—x), retreiving ma from M is a purely deterministic operation. Sampling from q(a—x) is easy as it amounts to computing its value for each slot in memory and sampling from the resulting categorical distribution. Given a, we can compute the probability of drawing that address under the prior p(a). We here use a learned prior p(a) that shares some parameters with q(a—x). Similarity functions: To obtain an efficient implementation for mini-batch training we use the same memory content M for the all training examples in a mini-batch and choose a specific form for the similarity function. We parameterize Sq (m, x) with two MLPs: h that embeds the memory content into the matching space and hq that does the same to the query x. The similarity is then computed as the inner product of the embeddings, normalized by the norm of the memory content embedding: hea , eq i ——ea ——2 where ea = h (ma) , eq = hq (x). Sq (ma , x) =

(5) (6)

This form allows us to compute the similarities between the embeddings of a mini-batch of —B— observations and —M— memory entries at the computational cost of O(—M——B——e—), where —e— is the dimensionality of the embedding. We also experimented with several alternative similarity functions q such as the plain inner product (hea , eq i) and the cosine similarity (hea , e i/——ea —— ? ——eq ——) and found that they did not outperform the above similarity function. For the unconditioneal prior p(a), we learn a query point ep 2 R—e— to use in similarity function (5) in place of eq . We share h between p(a) and q(a—x). Using a trainable p(a) allows the model to learn that some memory entries are more useful for generating new targets than others. Control experiments showed that there is only a very small degradation in performance when we assume a flat prior p(a) = 1/—M—. 2.1

Gradients and Training

For the continuous variable z we use the methods developed in the context of variational autoencoders [1]. We use a conditional Gaussian prior p(z—ma) and an approximate conditional posterior q(z—x, ma). However, since we have a discrete latent variable a in the model we can not simply backpropagate gradients through it. Here we show how to use VIMCO [11] to estimate the gradients for 3

this model. With VIMCO, we essentially optimize the multi-sample variational bound [12, 13, 11]: " # K 1 X p(x, ma , z) log p(x) E log =L (7) K q(a, z—x) a(k) ?q(a—x) k=1

z(k) ?q(z—ma ,x)

Multiple samples from the posterior enable VIMCO to estimate low-variance gradients for those parameters of the model which influence the non-differentiable discrete variable a. The corresponding gradient estimates are: ? ? X r? L ' ! (k) r? log p? (x, a(k) , z(k)) r? log q? (z—a, x) (8) a(k) ,z (k) ? q(?—x)

r L'

X

a(k) ,z (k)

! ? with ! (k) = P

(k)

(k)

? q(?—x)

(k)

r log q (a(k) —x)

p(x, a(k) , z(k)) q(a(k) , z(k) —x) X 0 1 log ! ? (k) K 1 0

, ! ? (k) =

! ? (k) 1 X (k0) = log ! ? K 0 k

and !

!

k

! (k)

k 6=k

For z-related gradients this is equivalent to IWAE [13]. Alternative gradient estimators for discrete latent variable models (e.g. NVIL [14], RWS [12] or Gumbel-max relaxation-based approaches [15, 16]) might work here too, but we have not investigated their effectiveness. Notice how the gradients r log p(x—z, a) provide updates for the memory contents ma (if necessary), while the gradients r log p(a) and r log q(a—x) provide updates for the embedding MLPs. The former update the mixture components while the latter update their relative weights. The log-likelihood bound (2) suggests that we can decompose the overall loss into three terms: the expected reconstruction error Ea,z?q [log p(x—a, z)] and the two KL terms which measure the information flow from the approximate posterior to the generative model for our latent variables: KL(q(a—x)——p(a)), and Ea?q [KL(q(z—a, x)——p(z—a))].

3

Related work

Attention and external memory are two closely related techniques that have recently become important building blocks for neural models. Attention has been widely used for supervised learning tasks such as translation, image classification and image captioning. External memory can be seen as an input or an internal state and attention mechanisms can either be used for selective reading or incremental updating. While most work involving memory and attention has been done in the context supervised learning, here we are interested in using them effectively in the generative setting. In [17] the authors use soft-attention with learned memory contents to augment models to have more parameters in the generative model. External memory as a way of implementing one-shot generalization was introduced in [18]. This was achieved by treating the exemplars conditioned on as memory entries accessed through a soft attention mechanism at each step of the incremental generative process similar to the one in DRAW [19]. Generative Matching Networks [20] are a similar architecture which uses a single-step VAE generative process instead of an iterative DRAW-like one. In both cases, soft attention is used to access the exemplar memory, with the address weights computed based on a learned similarity function between an observation at the address and a function of the latent state of the generative model. In contrast to this kind of deterministic soft addressing, we use hard attention, which stochastically picks a single memory entry and thus might be more appropriate in the few-shot setting. As the memory location is stochastic in our model, we perform variational inference over it, which has not been done for memory addressing in a generative model before. A similar approach has however been used for training stochastic attention for image captioning [21]. In the context of memory, hard attention has been used in RLNTM ? a version of the Neural Turing Machine modified to use stochastic hard addressing [22]. However, RLNTM has been trained using REINFORCE rather than variational inference. A number of architectures for VAEs augmented with mixture priors have 4

Figure 2: A: Typical learning curve when training a model to recall MNIST digits (M ? training data (each step); x ? M; —M— = 256): In the beginning the continuous latent variables model most of the variability of the data; after ? 100k update steps the stochastic memory component takes over and both the NLL bound and the KL(q(a—x)——p(a)) estimate approach log(256), the NLL of an optimal probabilistic lookup-table. B: Randomly selected samples from the MNIST model with learned memory: Samples within the same row use a common ma .

been proposed, but they do not use the mixture component indicator variable to index memory and integrate out the variable instead [9, 10], which prevents them from scaling to a large number of mixing components. An alternative approach to generative few-shot learning proposed in [23] uses a hierarchical VAE to model a large number of small related datasets jointly. The statistical structure common to observations in the same dataset are modelled by a continuous latent vector shared among all such observations. Unlike our model, this model is not memory-based and does not use any form of attention. Generative models with memory have also been proposed for sequence modelling in [24], using dif-

ferentiable soft addressing. Our approach to stochastic addressing is sufficiently general to be applicable in this setting as well and it would be interesting how it would perform as a plug-in replacement for soft addressing.

4

Experiments

We optimize the parameters with Adam [25] and report experiments with the best results from learning rates in {1e-4, 3e-4}. We use minibatches of size 32 and K=4 samples from the approximate posterior q(?—x) to compute the gradients, the KL estimates, and the log-likelihood bounds. We keep the architectures deliberately simple and do not use autoregressive connections or IAF [5] in our models as we are primarily interested in the quantitative and qualitative behaviour of the memory component. 4.1

MNIST with fully connected MLPs

We first perform a series of experiments on the binarized MNIST dataset [26]. We use 2 layered enand decoders with 256 and 128 hidden units with ReLU nonlinearities and a 32 dimensional Gaussian latent variable z. Train to recall: To investigate the model?s capability to use its memory to its full extent, we consider the case where it is trained to maximize the likelihood for random data points x which are present in M. During inference, an optimal model would pick the template ma that is equivalent to x with probability q(a—x)=1. The corresponding prior probability would be p(a) ? 1/—M—. Because there are no further variations that need to be modeled by z, its posterior q(z—x, m) can match the prior p(z—m), yielding a KL cost of zero. The model expected log likelihood would be -log —M—, equal to the log-likelihood of an optimal probabilistic lookup table. Figure 2A illustrates that our model converges to the optimal solution. We observed that the time to convergence depends on the size of the memory and with —M— ¿ 512 the model sometimes fails to find the optimal solution. It is noteworthy that the trained model from Figure 2A can handle much larger memory sizes at test time, e.g. achieving NLL ? log(2048) given 2048 test set images in memory. This indicates that the matching MLPs for q(a—x) are sufficiently discriminative. 5

Figure 3: Approximate inference with q(a—x): Histogram and corresponding top-5 entries ma for two randomly selected targets. M contains 10 examples from 8 unseen test-set character classes.

Figure 4: A: Generative one-shot sampling: Left most column is the testset example provided in M; remaining columns show randomly selected samples from p(x—M). The model was trained with 4 examples from 8 classes each per gradient step. B: Breakdown of the KL cost for different models trained with varying number of examples per class in memory. KL(q(a—x)——p(a)) increases from 2.0 to 4.5 nats as KL(q(z—ma , x)——p(z—ma)) decreases from 28.2 to 21.8 nats. As the number of examples per class increases, the model shifts the responsibility for modeling the data from the continuous variable z to the discrete a. The overall testset NLL for the different models improves from 75.1 to 69.1 nats.

Learned memory: We train models with —M— 2 {64, 128, 256, 512, 1024} randomly initialized mixture components (ma 2 R256). After training, all

models converged to an average KL(q(a—x)——p(a)) ? 2.5 ? 0.3 nats over both the training and the test set, suggesting that the model identified between e2.2 ? 9 and e2.8 ? 16 clusters in the data that are represented by a. The entropy of p(a) is significantly higher, indicating that multiple ma are used to represent the same data clusters. A manual inspection of the q(a—x) histograms confirms this interpretation. Although our model overfits slightly more to the training set, we do generally not observe a big difference between our model and the corresponding baseline VAE (a VAE with the same architecture, but without the top level mixture distribution) in terms of the final NLL. This is probably not surprising, because MNIST provides many training examples describing a relatively simple data manifold. Figure 2B shows samples from the model. 4.2

Omniglot with convolutional MLPs

To apply the model to a more challenging dataset and to use it for generative few-shot learning, we train it on various versions of the Omniglot [27] dataset. For these experiments we use convolutional en- and decoders: The approximate posterior q(z—m, x) takes the concatenation of x and m as input and predicts the mean and variance for the 64 dimensional z. It consists of 6 convolutional layers with 3 ? 3 kernels and 48 or 64 feature maps each. Every second layer uses a stride of 2 to get an overall downsampling of 8 ? 8. The convolutional pyramid is followed by a fully-connected MLP with 1 hidden layer and 2—z—output units. The architecture of p(x—m, z) uses the same downscaling pyramid to map m to a —z—-dimensional vector, which is concatenated with z and upscaled with transposed convolutions to the full image size again. We use skip connections from the downscaling layers of m to the corresponding upscaling layers to preserve a high bandwidth path from m to x. To reduce overfitting, given the relatively small size of the Omniglot dataset, we tie the parameters of the convolutional downscaling layers in q(z—m) and p(x—m, z). The embedding MLPs for p(a) and q(a—x) use the same convolutional architecture and map images x and memory content ma into 6

Figure 5: Robustness to increasing memory size at test-time: A: Varying the number of confounding memory entries: At test-time we vary the number of classes in M. For an optimal model of disjoint data from C classes we expect L = average L per class + log C (dashed lines). The model was trained with 4 examples from 8 character classes in memory per gradient step. We also show our best soft-attenttion baseline model which was trained with 16 examples from two classes each gradient step. B: Memory contains examples from all 144 test-set character classes and we vary the number of examples per class. At C=0 we show the LL of our best unconditioned baseline VAE. The models were trained with 8 character classes and {1, 4, 8} examples per class in memory. a 128-dimensional matching space for the similarity calculations. We left their parameters untied because we did not observe any improvement nor degradation of performance when tying them. With learned memory: We run experiments on the 28 ? 28 pixel sized version of Omniglot which was introduced in [13]. The dataset contains 24,345 unlabeled examples in the training, and 8,070 examples in the test set from 1623 different character classes. The goal of this experiment is to show that our architecture can learn to use the top-level memory to model

highly multi-modal input data. We run experiments with up to 2048 randomly initialized mixture components and observe that the model makes substantial use of them: The average KL(q(a—x)——p(a)) typically approaches log —M—, while KL(q(z—?)——p(z—?)) and the overall training-set NLL are significantly lower compared to the corresponding baseline VAE. However big models without regularization tend to overfit heavily (e.g. training-set NLL ¡ 80 nats; testset NLL ¿ 150 nats when using —M—=2048). By constraining the model size (—M—=256, convolutions with 32 feature maps) and adding 3e-4 L2 weight decay to all parameters with the exception of M, we obtain a model with a testset NLL of 103.6 nats (evaluated with K=5000 samples from the posterior), which is about the same as a two-layer IWAE and slightly worse than the best RBMs (103.4 and ?100 respectively, [13]). Few-shot learning: The 28 ? 28 pixel version [13] of Omniglot does not contain any alphabet or character-class labels. For few-shot learning we therefore start from the original dataset [27] and scale the 104 ? 104 pixel sized examples with 4 ? 4 max-pooling to 26 ? 26 pixels. We here use the 45/5 split introduced in [18] because we are mostly interested in the quantitative behaviour of the memory component, and not so much in finding optimal regularization hyperparameters to maximize performance on small datasets. For each gradient step, we sample 8 random character-classes from random alphabets. From each character-class we sample 4 examples and use them as targets x to form a minibatch of size 32. Depending on the experiment, we select a certain number of the remaining examples from the same character classes to populate M. We chose 8 character-classes and 4 examples per class for computational convenience (to obtain reasonable minibatch and memory sizes). In control experiments with 32 character classes per minibatch we obtain almost indistinguishable learning dynamics and results. To establish meaningful baselines, we train additional models with identical encoder and decoder architectures: 1) A simple, unconditioned VAE. 2) A memory-augmented generative model with soft-attention. Because the soft-attention weights have to depend solely on the variables in the generative model and may not take input directly from the encoder, we have to use z as the top-level latent variable: p(z), p(x—z, m(z)) and q(z—x). The overall structure of this model resembles the structure of prior work on memory-augmented generative models (see section 3 and Figure 1A), and is very similar to the one used in [20], for example. For the unconditioned baseline VAE we obtain a NLL of 90.8, while our memory augmented model reaches up to 68.8 nats. Figure 5 shows the scaling properties of our model when varying the number of conditioning examples at test-time. We observe only minimal degradation compared 7

Model Generative Matching Nets Generative Matching Nets Generative Matching Nets Variational Memory Addressing Variational Memory Addressing Variational Memory Addressing Variational Memory Addressing

Ctest 1 2 4 1 2 4 16

1 83.3 86.4 88.3 86.5 87.2 87.5 89.6

2 78.9 84.9 87.3 83.0 83.3 83.3 85.1

3 75.7 82.4 86.7 79.6 80.9 81.2 81.5

4 72.9 81.0 85.4 79.0 79.3 80.7 81.9

5 70.1 78.8 84.0 76.5 79.1 79.5 81.3
10 59.9 71.4 80.2 76.2 77.0 78.6 79.8
19 45.8 61.2 73.7 73.9 75.0 76.7 77.0

Table 1: Our model compared to Generative Matching Networks [20]: GMNs have an extra stage that computes joint statistics over the memory context which gives the model a clear advantage when multiple conditiong examples per class are available. But with increasing number of classes C it quickly degrades. LL bounds were evaluated with K=1000 posterior samples.

to a theoretically optimal model when we increase the number of concurrent character classes in memory up to 144, indicating that memory readout works reliably with —M— 2500 items in memory. The soft-attention baseline model reaches up to 73.4 nats when M contains 16 examples from 1 or 2 character-classes, but degrades rapidly with increasing number of confounding classes (see Figure 5A). Figure 3 shows histograms and samples from q(a—x), visually confirming that our model performs reliable approximate inference over the memory locations. We also train a model on the Omniglot dataset used in [20]. This split provides a relatively small training set. We reduce the number of feature channels and hidden layers in our MLPs and add 3e-4 L2 weight decay to all parameters to reduce overfitting. The model in [20] has a clear advantage when many examples from very few character classes are in memory because it was specifically designed to extract joint statistics from memory before applying the soft-attention readout. But like our own soft-attention baseline, it quickly degrades as the number of concurrent classes in memory is increased to 4 (table 1). Few-shot classification: Although this is not the main aim of this paper, we can use the trained model to perform discriminative few-shot classification: We can estimate p(c—x) ? P E [p(x, z, ma)/p(x)] or by using the feed forward approximation p(c—x) ? m has P a label c z?q(z—a,x) q(a—x). Without any further retraining or finetuneing we obtain classification accuracies ma has label c of 91%, 97%, 77% and 90% for 5-way 1-shot, 5-way 5-shot, 20-way 1-shot and 20-way 5-shot respectively with q(a—x).

5

Conclusions

In our experiments we generally observe that the proposed model is very well behaved: we never used temperature annealing for the categorical softmax or other tricks to encourage the model to use memory. The interplay between p(a) and q(a—x) maintains exploration (high entropy) during the early phase of training and decreases naturally as the sampled ma become more informative. The KL divergences for the continuous and discrete latent variables show intuitively interpretable results for all our experiments: On the densely sampled MNIST dataset only a few distinctive mixture components are identified, while on the more disjoint and sparsely sampled Omniglot dataset the model chooses to use many more memory entries and uses the continuous latent variables less. By interpreting memory addressing as a stochastic operation, we gain the ability to apply a variational approximation which helps the model to perform precise memory lookups during inference and training. Compared to soft-attention approaches, we loose the ability to naively backprop through

read-operations and we have to use approximations like VIMCO. However, our experiments strongly suggest that this can be a worthwhile trade-off. Our experiments also show that the proposed variational approximation is robust to increasing memory sizes: A model trained with 32 items in memory performed nearly optimally with more than 2500 items in memory at test-time. Beginning with M 48 our hard-attention implementation becomes noticeably faster in terms of wall-clock time per parameter update than the corresponding soft-attention baseline. Even though we use K=4 posterior samples during training and the soft-attention baseline only requires a single one. 8

Acknowledgments We thank our colleagues at DeepMind and especially Oriol Vinyals and Sergey Bartunov for insightful discussions.

2 References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013. [2] Danilo Jimenez 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. [3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672?2680, 2014. [4] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. arXiv preprint arXiv:1505.05770, 2015. [5] Diederik P Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. arXiv preprint arXiv:1606.04934, 2016. [6] Sreerupa Das, C. Lee Giles, and Guo zheng Sun. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory. In In Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society, pages 791?795. Morgan Kaufmann Publishers, 1992. [7] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 2440?2448. Curran Associates, Inc., 2015. [8] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka GrabskaBarwi?nska, Sergio G?mez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. Nature, 538(7626):471?476, 2016. [9] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. arXiv preprint arXiv:1611.02648, 2016. [10] Eric Nalisnick, Lars Hertel, and Padhraic Smyth. Approximate inference for deep latent gaussian mixtures. In NIPS Workshop on Bayesian Deep Learning, 2016. [11] Andriy Mnih and Danilo J Rezende. Variational inference for monte carlo objectives. arXiv preprint arXiv:1602.06725, 2016. [12] J?rg Bornschein and Yoshua Ben-

gio. Reweighted wake-sleep. arXiv preprint arXiv:1406.2751, 2014. [13] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. arXiv preprint arXiv:1509.00519, 2015. [14] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. arXiv preprint arXiv:1402.0030, 2014. [15] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712, 2016. [16] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. stat, 1050:1, 2017. 9

[17] Chongxuan Li, Jun Zhu, and Bo Zhang. Learning to generate with memory. In International Conference on Machine Learning, pages 1177?1186, 2016. [18] Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. arXiv preprint arXiv:1603.05106, 2016. [19] DRAW: A Recurrent Neural Network For Image Generation, 2015. [20] Sergey Bartunov and Dmitry P Vetrov. Fast adaptation in generative models with generative matching networks. arXiv preprint arXiv:1612.02192, 2016. [21] Jimmy Ba, Ruslan R Salakhutdinov, Roger B Grosse, and Brendan J Frey. Learning wake-sleep recurrent attention models. In Advances in Neural Information Processing Systems, pages 2593?2601, 2015. [22] Wojciech Zaremba and Ilya Sutskever. Reinforcement learning neural turing machines. arXiv preprint arXiv:1505.00521, 362, 2015. [23] Harrison Edwards and Amos Storkey. Towards a Neural Statistician. 2 2017. [24] Mevlana Gemici, Chia-Chun Hung, Adam Santoro, Greg Wayne, Shakir Mohamed, Danilo J Rezende, David Amos, and Timothy Lillicrap. Generative temporal models with memory. arXiv preprint arXiv:1702.04649, 2017. [25] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014. [26] Hugo Larochelle. Binarized mnist dataset http://www.cs.toronto.edu/larocheh/public/ datasets/binarized_mnist/binarized_mnist_[train—val
2011. [27] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. Science, 350(6266):1332?1338, 2015.

10