EXPERIMENTS WITH AIR: A GENERATIVE MODEL FOR SCENES

FINAL REPORT

Arun Drelich

May 15, 2019

1 Introduction

Generative models are an important collection of techniques for learning unsupervised representations of data. Amortised variational inference algorithms, in particular Variational Autoencoders (VAEs), are a family of generative models that allow sampling latent random variables from some prior distribution in order to generate observations from a dataset. VAEs have found a wide range of applications, due to their ability to generate complex data from a low-dimensional latent space. However, these learned latent spaces are rarely interpretable or structured. Learning good structured representations is often a difficult task that depends on building effective inductive biases into a model.

In Attend, Infer, Repeat [1], the authors demonstrate an approach that combines structured probabilistic models with deep learning in order to reconstruct a scene. A scene is assumed to consist of some number of objects, each of which can be described by an affine pose as well as a latent code describing its appearance. The model is at its core structured like a variational auto-encoder, and its parameters are learned by maximising the evidence lower bound (ELBO) on the likelihood. AIR iteratively attends to scene objects, inferring a reconstruction for each object, taking a variable number of steps for each input canvas. Crucially, this process is unsupervised.

The goals of this project were to obtain a comprehensive understanding of the various components of the AIR model, as well as the complexities that arise during implementation. A brief background on the fundamentals behind VAEs serves to introduce notation and lay a foundation for the intuition behind AIR. Following an overview of the primary components of the model, we turn to its design and implementation in PyTorch, including several non-trivial details surrounding the use of a discrete latent variable for counting objects.

2 Background: Variational Inference

The variational auto-encoder was introduced by Kingma and Welling [3] as a solution to the task of approximating a posterior probability $p(z|x)$, assuming that computing the true posterior is intractable. This task appears often: consider, for example, a dataset of images $\{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$. We assume there exists a latent variable z that describes any image x, and wish to generate an image given some z (sampling from $p(z|x)$). In addition, we may want to perform marginal inference – given some x, how might we encode it to obtain a more concise representation?

Since the true posterior is assumed to be intractable, we instead approximate it with a variational posterior $q_\phi(z|x)$, parameterised by ϕ. Similarly, we parameterise the likelihood over θ as $p_\theta(x|z)$. To learn these parameters, we maximise $p_\theta(x)$ – by maximising its lower bound $\mathcal{L}(\theta, \phi)$:

$$\begin{aligned}
\log p_\theta(x) &\geq \mathcal{L}(\theta, \phi) \\
&= \mathbb{E}_{q_\phi}\Big[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\Big] \\
&= \mathbb{E}_{q_\phi}\Big[\log p_\theta(x|z)\Big] - D(q_\phi(z|x)||p(z))
\end{aligned}$$

In practice, the negative ELBO is used, in order to learn θ and ϕ with gradient descent.

3 Overview of AIR

AIR is structured similarly to a VAE, which learns to reconstruct a canvas $\boldsymbol{x}$ by structuring it into a latent representation $\boldsymbol{z} = (\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^n)$ consisting of: a C-dimensional continuous distributed representation $\boldsymbol{z}^i_{\text{what}}$, 3D affine pose parameter $\boldsymbol{z}^i_{\text{where}}$ encoding position and scale, as well as discrete indicator z^i_{pres} for presence. There is a latent descriptor $\boldsymbol{z}^i$ for each object $i \in [0, N]$ in the scene, for a fixed maximum number of steps N.

A recursive neural network is responsible for taking timesteps, in which it iteratively attends to individual objects, outputting parameters over $q_\phi(\boldsymbol{z}^i_{\text{where}}|\boldsymbol{x})$ and $q_\phi(z^i_{\text{pres}}|\boldsymbol{x})$. Given a sampled pose $\boldsymbol{z}^i_{\text{where}}$, an object $\boldsymbol{x}^i_{\text{att}}$ is cropped from the scene and encoded to a latent $\boldsymbol{z}^i_{\text{what}}$. Presence of reconstructed $\boldsymbol{y}^i_{\text{att}}$ is controlled by the indicator z^i_{pres}. Upon sampling $z^i_{\text{pres}} = 0$, no more objects are added to the scene – z_{pres} can be thought of as a unary code.

Since we'd like to model the number of objects in a scene, it would be more appropriate to have a single latent variable $n \sim \text{Geom}(p)$ rather than several $z^i_{\text{pres}} \sim \text{Bernoulli}(p'_i)$. In [1], the distribution over unary code $\boldsymbol{z}_{\text{pres}}$ is transformed into a geometric prior over n.

Experiments were performed in [1] on a dataset of Multi-MNIST images. Between zero and three MNIST digits are placed on a 50×50 canvas uniformly at random. Our implementation loaded a Multi-MNIST dataset using the Edward library's [9] `observations` module, which randomly scales each 28×28 image by a factor of $1/s$, $s \sim \mathcal{N}(1.3, 0.1)$ and positions them uniformly on the canvas, discarding samples with overlapping digits [1]

3.1 Gradient Estimation

As in a standard variational autoencoder, the parameters of the AIR model are learned by minimising the negative ELBO $\mathcal{L}(\theta, \phi)$. Crucially, the loss term is an expectation, and it's therefore necessary to obtain a Monte Carlo estimator for $\frac{\partial}{\partial\theta}\mathcal{L}$. Given samples from the variational posterior q_ϕ, computing $\frac{\partial}{\partial\theta}\log p_\theta(\boldsymbol{x}, \boldsymbol{z})$ is trivial as p is differentiable with regard to θ. On the other hand, $\frac{\partial}{\partial\phi}\log q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is more involved. Each $\boldsymbol{z}^i$ is obtained as a result of sampling from its respective variational posterior, with the reconstructed canvas a downstream step in the computation graph. In order to compute $\frac{\partial}{\partial\phi}\mathcal{L}$, gradients must be back-propagated through stochastic computation nodes.

In [8], Schulman et. al. introduce some new formalisms for reasoning about such stochastic computation graphs. They show that, in general, it is possible to either a) transform a stochastic node into a leaf which does not depend on any learnable parameters or b) use a gradient estimator based on the log-likelihood of that sample. The first, also known as the reparameterisation "trick" [3], can be applied to continuous random variables $\boldsymbol{z}_{\text{where}}$ and $\boldsymbol{z}_{\text{what}}$ by assigning latent Gaussian priors to each descriptor. Sampling these from q_ϕ is then possible by defining $\epsilon \sim \mathcal{N}(0, 1)$ so that $z \sim \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$ is equivalent to $z = \mu_\phi(x) + \sigma_\phi(x) * \epsilon$. However, not all distributions are reparameteriseable. To obtain gradients for the discrete z^i_{pres} we use a score function estimator, called a likelihood ratio estimator in [1]. Assuming $l(z)$ is a deterministic function (representing our loss term) that depends on stochastic node $z \sim q_\phi(z)$:

$$\frac{\partial}{\partial\phi}\mathbb{E}_{z\sim q_\phi}\big[l(z)\big] = \frac{\partial}{\partial\phi}\mathbb{E}_{z\sim q_\phi}\Big[\frac{\partial \log q_\phi(z)}{\partial\phi}l(z)\Big]$$

Unfortunately, the score function estimator is known to exhibit high variance, which is the primary reason why training AIR is difficult.

3.2 Variance Reduction with Neural Baselines

Due to the noisiness of the score function estimator, the authors in [1] used neural baselines [5] to reduce its variance. The term *baseline* is used here to refer to an appropriately-structured value c that can be subtracted

[1] A sample was found to have overlapping digits if any pixel had a value greater than 255 – which may exclude certain cases of overlapping greys.

from the expectation without affecting its value:

$$\frac{\partial}{\partial \phi}\mathbb{E}_{z\sim q_\phi}\big[l(z)\big] = \frac{\partial}{\partial \phi}\mathbb{E}_{z\sim q_\phi}\big[l(z) - c\big]$$

The value c subtracted can be the output of a data-dependent term $C_\psi(x)$ that optimally centers the learning signal. As long as $C_\psi(x)$ does not depend on hidden nodes of $l(z)$, it does not affect gradient estimates of $\nabla l(z)$. This function $C_\psi(x)$ is the neural baseline referred to by the AIR authors. We implemented neural baselines for variance reduction with a similarly-structured RNN that takes as input the original canvas and sampled latent codes at each time-step and is trained to minimise the mean squared error with the AIR model's log-likelihood. As shown in Section 6, the use of baselines significantly reduces variance of the score-function estimator.

4 Implementation

This project was implemented using PyTorch [6] as the automatic differentiation and tensor math library. Some emphasis was placed on modularity and design – each sub-component neural network of AIR extending `torch.nn.Module`.

4.1 Inference Network

The "inference network" described in section 2.1 of [1] is implemented as an LSTM cell, evaluated at each time step i. It takes as input the original canvas $\boldsymbol{x}$, and previously-sampled latent codes $\boldsymbol{z}^{i-1}$, producing an embedding $\boldsymbol{h}^i = f_1(\boldsymbol{x}, \boldsymbol{z}^{i-1}, \boldsymbol{h}^{i-1})$. This embedding is passed through a neural network f_2 to obtain parameters over the distributions of the latent codes at each timestep. The variational posterior $q_\phi(\boldsymbol{z}^i_{\text{where}}, \boldsymbol{z}^i_{\text{pres}}|\boldsymbol{x}, \boldsymbol{z}^{1:(i-1)})$ is parameterised by $f_2(f_1(\boldsymbol{x}, \boldsymbol{z}^{i-1}, \boldsymbol{h}^{i-1}))$.

4.2 Attention mechanism

Given an affine pose descriptor $\boldsymbol{z}^i_{\text{where}}$, a spatial transformer (STN) [2] crops the attention object $\boldsymbol{x}^i_{\text{att}}$ from the canvas. This transformer network allows for differentiable attention mechanisms: cropping glimpse windows from a canvas given an affine pose, and placing reconstructions back into the canvas given an inverse pose.

The authors of [2] describe a convolutional localisation network. Here, $q_\phi(\boldsymbol{z}^i_{\text{where}}|\boldsymbol{x})$ acts analogously to that localisation network, learning transformation parameters that result in optimal loss scores. As in [2], a grid generator outputs a grid of coordinates corresponding to pixels in the output crop, and a sampler uses the transformation parameters $\boldsymbol{z}^i_{\text{where}}$ to crop $\boldsymbol{x}^i_{\text{att}}$.

4.3 Attention window encoder and decoder

The variational posterior $q_\phi(\boldsymbol{z}^i_{\text{what}}|\boldsymbol{x}^i_{\text{att}})$ is implemented as a neural network that predicts parameters from which to sample $\boldsymbol{z}^i_{\text{what}}$.

Similarly, a decoder network learns to reconstruct $\boldsymbol{y}^i_{\text{att}}$, given $\boldsymbol{z}^i_{\text{what}}$.

4.4 Baseline network

Finally, we implemented a data-dependent baseline for variance reduction of the score function estimator. Input to the network is the original flattened canvas $\boldsymbol{x}$, concatenated with the latent variables $\boldsymbol{z}^{i-1}_{\text{where}}, \boldsymbol{z}^{i-1}_{\text{what}}, z^{i-1}_{\text{pres}}$ sampled in the previous attention step. The structure of the baseline network mirrors that of the inference network's RNN. Let $\boldsymbol{h}^i = R(\boldsymbol{x}, \boldsymbol{z}^{i-1}, \boldsymbol{h}^{i-1})$ represent the RNN's H-dimensional hidden state at step i, and let $f_\theta : \mathbb{R}^H \to \mathbb{R}$ be a 2-layer MLP with parameters θ. The baseline term can then be defined as:

$$C_\psi(\boldsymbol{x}) = \sum_{i=1}^{N} f_\theta(\boldsymbol{h}^i) * z^i_{\text{pres}}$$

By using z^i_{pres} as a mask, the baseline network does not need to learn to predict likelihoods for non-present objects. Latent states $\boldsymbol{z}^i$ are treated as inputs – gradients are detached from AIR's computation graph to avoid affecting the expected value of the gradient estimator.

Gradient Estimator	Variance
Backprop. ($\frac{\partial \mathcal{L}}{\partial \theta}$)	5.83×10^{-2}
SF ($\frac{\partial \mathcal{L}}{\partial \phi}$)	2.03×10^{4}
SF-BASELINE ($\frac{\partial \mathcal{L}}{\partial \phi}$)	6.47×10^{2}

Table 1: Variance reduction with neural baselines

This network was trained to minimise the mean squared error of the likelihood used by the score function estimator. A learning rate of 1×10^{-3} was used in [1], a factor of 10 higher than the learning rate 1×10^{-4} used by the main AIR network. In our implementation, we found a significantly higher learning rate of 0.1 to be more effective.

5 Experiments

We experimented with several different hyperparameters and learning strategies. A fixed geometric prior with success probability 0.01 was compared with annealing success probabilities from close to 1 down to a very low value (e.g. 1×10^{-7}) over the course of training. Prior annealing was implemented with an exponential decay rate. According to the authors in [1], this method is intended to encourage the model to explore the use of objects early in training while discouraging taking unnecessary steps later, "favouring sparse reconstructions".

Gaussian priors were defined over pose and attribute latent codes: $\boldsymbol{z}^i_{\text{where}} \sim \mathcal{N}\Big((3,0,0), \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\Big)$ and $\boldsymbol{z}^i_{\text{what}} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. Following the experiments in [1], we used a Gaussian likelihood $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, with fixed standard deviation 0.3 and $\boldsymbol{\mu}$ a summation of the attention window decoder's translated and scaled outputs over timesteps. Images were normalised to $[0, 1]$.

In Table 1, we show the effect of incorporating a data-dependent baseline term on the variance of the score function estimator. The AIR model and baseline network were trained for 160k steps, and the learned model parameters fixed. A batch was selected at random to be used for evaluation. Gradient estimates were obtained by running a forward pass followed by a backwards pass on the loss tensor, and taking the vector norm of gradients for relevant parameters. One thousand samples were computed for each estimator and the variance computed across those samples. The gradient of the loss with respect to a fully-deterministic subgraph (i.e. $\mathbb{E}\big[\frac{\partial \mathcal{L}}{\partial \theta}\big]$) exhibited significantly lower variance than the score function (SF) estimates of gradients back-propagated through stochastic nodes $\big(\mathbb{E}\big[\frac{\partial \mathcal{L}}{\partial \phi}\big]\big)$. The inclusion of a neural baseline $C_\psi(\boldsymbol{x})$ saw an order-of-magnitude decrease in variance, and proved necessary in order for training to reliably converge.

Count accuracies were measured using true counts given by the dataset. A desireable quality of AIR is its ability to learn to count objects in a scene without supervision, due to it taking a variable number of steps in order to describe a scene. The authors in [1] report accuracies above 95% after around 20k training steps. Learning to predict z^i_{pres} is crucial to the model's performance. Unfortunately, in our implementation we were only able to achieve accuracies closer to 45% after $\sim$ 200k steps. These results were somewhat dependent on annealing schedules and prior parameters, and varied strongly with initialisation. Ultimately this was likely due to an implementation bug.

6 Discussion

In our experiments, we were able to reproduce some of the results reported in [1] successfully. The model learned good representations for $\boldsymbol{z}_{\text{where}}$ and $\boldsymbol{z}_{\text{what}}$, attending to scene objects and reconstructing them. Unfortunately, training failed to reliably converge to a good representation for $\boldsymbol{z}_{\text{pres}}$ and the model was unable to distinguish individual objects with high accuracy. Instead, it would often use multiple attention steps to explain a single object, as can be observed in Figure 2. Often, the model would also take several steps to describe non-existent objects, likely because the reconstruction loss term dominated and was not penalised strongly enough. Occasionally, it was unable to distinguish two adjacent objects from a single large object and reconstruction suffered. More rarely, fewer objects were reconstructed than originally present. In general though, the learned model was able to avoid generating spurious objects. Between the reconstruction term

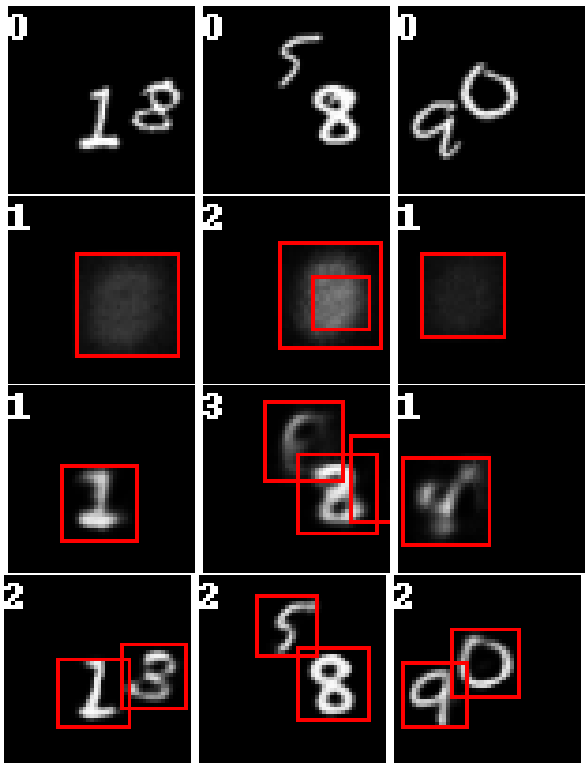

Figure 1: From the top: Original, reconstructions after 1000, 10,000 and 200,000 training steps.

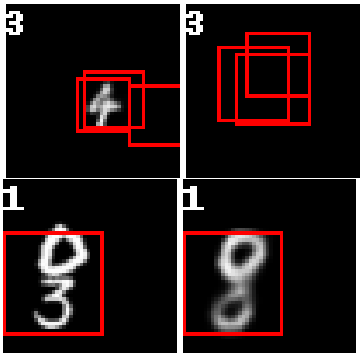

Figure 2: Examples of common failure conditions of AIR. Top left to top right: multiple attention steps for a single object, loss not penalised by taking unnecessary steps. Bottom left, bottom right: incorrect z^i_{where} and corresponding bad reconstruction.

and priors on content and pose it appears that the hallucination of more objects than originally present was sufficiently penalised during training.

There is strong potential for future work based on AIR. The authors themselves proposed possible lines of work such as: semi- or fully supervised learning, alternative likelihoods, and using state-of-the-art architectures as sub-components of the network [1]. In addition, it may be worth examining the design of latent priors more carefully. Relaxation of the discrete geometric prior to a continuous Concrete distribution [4], which has a closed-form density and can be reparameterised, may allow for significant variance reduction of gradient estimators. Introducing normalising flows [7] may allow for specifying better approximate posteriors that provide tighter evidence lower bounds.

References

[1] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3225–3233. Curran Associates, Inc., 2016.

[2] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks, 2015.

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

[4] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables, 2016.

[5] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks, 2014.

[6] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[7] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

[8] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs, 2015.

[9] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.