IMPORTANCE WEIGHTED AUTOENCODERS

Yuri Burda, Roger Grosse & Ruslan Salakhutdinov

Presented by Neha Das as a part of the Advanced Practical Course: Deep Learning in Real World

BASIS OF IWAE: VAE

The variational autoencoder is a generative modelling technique that can be used to model or approximate posterior distributions given some observations. This model ultimately aids us in generating data similar to the observed data.

Mathematical Foundation: In a VAE, we try to maximise the log likelihood of the generated samples and simultaneously model the actual posterior p(z|x) by q by minimizing the KL divergence between q(z) and the posterior p(z|x). However, in most cases the actual posterior is not available to us. To overcome this, we maximize the variational lowerbound or LELBO (circled), which achieves the same objective as minimizing the KL divergence.

$$\ln p(\mathbf{x} \mid \theta) = \underbrace{\int q(\mathbf{z}) \ln \frac{p(\mathbf{x}, \mathbf{z} \mid \theta)}{q(\mathbf{z})} \, \mathrm{d}\mathbf{z}}_{\mathcal{L}_{\mathrm{ELBO}}(q, \theta)} + \underbrace{\int q(\mathbf{z}) \ln \frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x}, \theta)} \, \mathrm{d}\mathbf{z}}_{\mathrm{KL}(q(\mathbf{z}) \,||\, p(\mathbf{z} \mid \mathbf{x}, \theta))}$$

Limitation: VAE makes very strong assumptions about the posterior p(z|x) it needs to model. These include:

- The posterior should be approximately factorial over the latent variables (represented by h).

  $p(z) = p(z_1,z_2,\ldots,z_n) = p(z_1).p(z_2)\ldots p(z_n)$

- It's parameters can be predicted using the observables via non-linear regression.

These assumptions limit the expressive power of the model.

IWAE OVER VAE

IWAEs share the VAE architecture but are trained with a log-likelihood lower bound $L_k(x)$ that is tighter than LELBO which basically corresponds to $L_1(x)$:

$$\mathcal{L}_k(\mathbf{x}) = \mathbb{E}_{\mathbf{z}_1,\ldots,\mathbf{z}_k \sim q(\mathbf{z}|\mathbf{x})} \left[\log \frac{1}{k} \sum_{i=1}^{k} \frac{p(\mathbf{x}, \mathbf{z}_i)}{q(\mathbf{z}_i|\mathbf{x})} \right], \quad w_i = \frac{p(\mathbf{x}, \mathbf{z}_i)}{q(\mathbf{z}_i|\mathbf{x})}$$

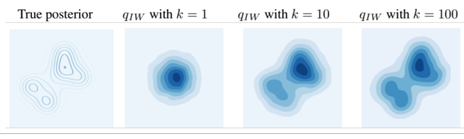
Note that this is different from an LELBO calculated as an average of k samples, which would still be smaller than $L_k(x)$:

$$\underbrace{\mathbb{E}\left[\frac{1}{k} \log \sum_{i=1}^{k} w_i\right]}_{\mathcal{L}_{\mathrm{ELBO}}(q,\theta)} \leq \underbrace{\mathbb{E}\left[\log \frac{1}{k} \sum_{i=1}^{k} w_i\right]}_{\mathcal{L}_k(\mathbf{x})} \leq \underbrace{\log \mathbb{E}\left[\frac{1}{k} \sum_{i=1}^{k} w_i\right]}_{\log p(\mathbf{x})}$$

Relation to the number of samples: As the number of samples or k increases, the above bound becomes tighter on the log likelihood of the observed data. Note that when k=1, $L_k(x)$ corresponds to VAE and when k → infinity, $L_k(x) = p(x)$

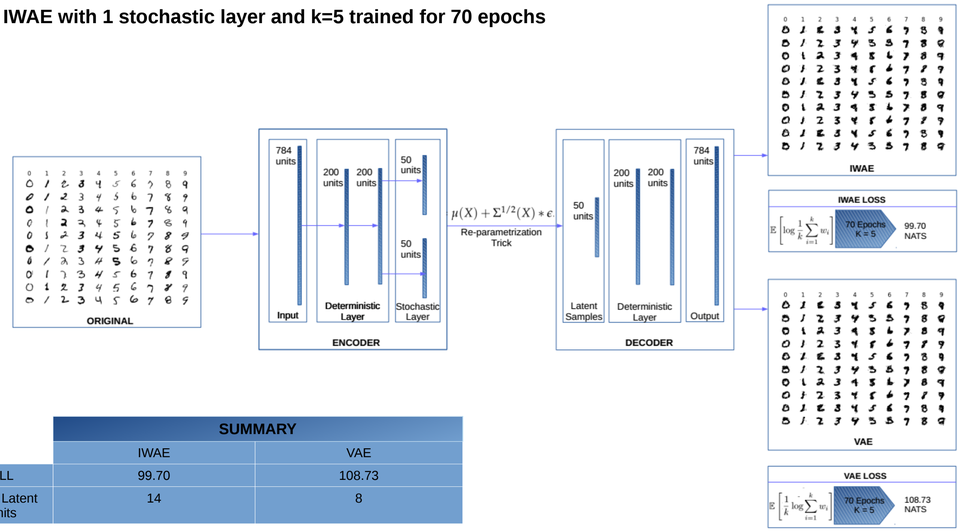
Approximate Posterior flexibility: IWAE loss ensures that the resultant implcit importance weighted posterior is flexible and can model complex distributions[2]. The following figure, taken from [2], illustrates this.

Fig 1. **IWAE can model more complex distributions than VAE**



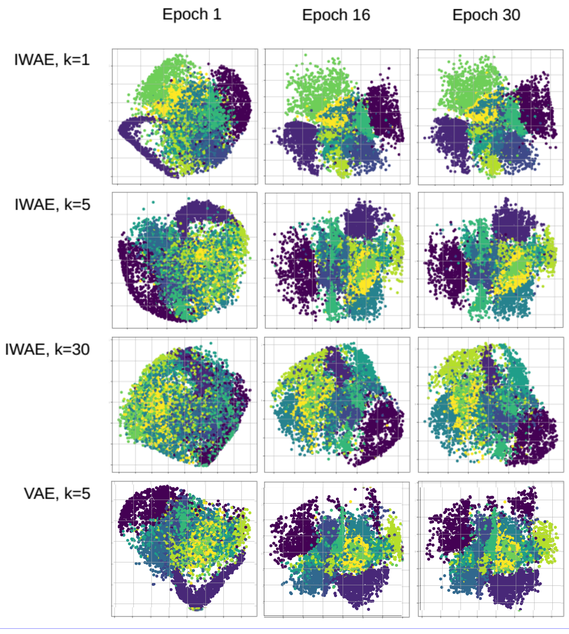
ARCHITECTURE AND EXPERIMENTAL RESULTS

Fig 2. **IWAE with 1 stochastic layer and k=5 trained for 70 epochs**



	SUMMARY	
	IWAE	VAE
NLL	99.70	108.73
Active Latent Units	14	8

Fig 3. **Scatter Plots for 2-D Latents**



RE-INTERPRETING IWAE: IMPLICIT IMPORTANCE WEIGHTED POSTERIOR [2]

Instead of following the standard interpretation of importance-weighted autoencoders is that they maximize a tighter lower bound on the marginal likelihood, we can also say: that it optimizes the standard variational lower bound, but using a more complex distribution.

In [2], this distribution was mathematically derived as:

$$q_{IW}(z_i|x, z_{\setminus i}) = k\tilde{w}_i q(z_i|x) = \left(\frac{\frac{p(x,z_i)}{q(z_i|x)}}{\frac{1}{k}\sum_{j=1}^{k} \frac{p(x,z_j)}{q(z_j|x)}} \right) q(z_i|x)$$

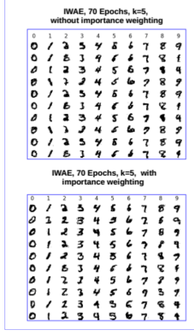
From the above statements, it follows that reweighting the latent samples with importance weights might lead to the generation of more accurate samples.

Fig 4. **Algorithm for obtaining reweighted samples**

Algorithm 1 Sampling from q_{IW}

1: $k \leftarrow$ *number of samples*
2: $q(z|x) = f_\phi(x)$
3: **for** i in $1 \ldots k$ **do**
4: $\quad z_i \sim q(z|x)$
5: $\quad w_i = \frac{p(x,z_i)}{q(z_i|x)}$
6: Each $\tilde{w} = w_i / \sum_{i=1}^{k} w_i$
7: $j \sim Cat(\tilde{w})$
8: Return z_j

Fig 5. **With and without importance weighting**



DIFFERENCES & CHALLENGES OF REIMPLEMENTATION

The original implementation was in **theano**. The results shown here, unless otherwise indicated are from a re-implementation of the paper in **tensorflow**. Comparison of the two approaches

ASPECT	ORIGINAL	REIMPLEMENTED
Number of epochs	3280	70
Architecture	1,2 Stochastic Layer	1 Stochastic Layer
Number of hidden units	1 Stoch. Layer [50] 2 Stoch. Layer [50,100]	[50].[2]
K (Num of samples)	1,5,50	1,5,50
Min NLL achieved (k=5)	85.54	99.70
Number of Active Units in the Latent Layer (k=5)	22	14

Challenges

- **Time**: 1 epoch for k=50 took roughly 20 minutes to run on an 8GB RAM system without GPU capabilities. Running 3280 epochs would have taken up over 1000 hours. As a result, the network was trained for a mere 70 epochs over different k (number of samples) and hidden layers. A steady decrease in loss during the training reinforced confidence in the network.
- **Nan in loss and gradients**: This happened mainly because a few variables inside log (while calculating log likelihood, or evaluating the gradient) ended up us 0. Adding a small constant of the order of 1e-32 *while training* helped alleviate this. This constant was removed during the testing process. Addition
- **Testing the correctness of the Network:** Testing and implementation of the network went hand in hand. Tensorboard was used to verify the graph flow. Small handcrafted test cases were used verify module correctness
- **Debugging:** Debugging in tensorflow is rather cumbersome. We used both tensorflow.Print and tensorflow.Python.Debug for debbuging errors in the network. Although, the latter is harder to get used to, it appears to be a one stop shop for viewing all the tensorvariables during a session run

CONCLUSIONS & INSIGHTS

- **Negative Log Likelihood** (Fig 2): From the experimental results, IWAE clearly achieves a better negative log likelihood than plan VAE even when VAE utilises the trick of averaging over several samples.
- **Active Latent Units** (Fig 2)**:** Training with an IWAE objective hellps in building a richer latent space than VAE (14 versus 8).
- **Form of the posterior** (Fig 3)**:** While it is hard to draw conclusions from the scatter plot results for latent units in 2 dimensions, one may note that with increasing number of samples (k), the posterior form stretches from its formely circular shape. This becomes more apparent as the training progresses. This indicates that with IWAE, we are capable of approximating a more complex posterior than VAE.
- **Sampling from the importance weighted posterior** (Fig 4 & 5) **:** Sampling with importance weighting was noted to be marginally improved than without in the experimental results. This gap may widen as the training progresses.

REFERENCES

[1] Yuri Burda, Roger Grosse & Ruslan Salakhutdinov, Importance Weighted Autoencoders.
[2] Chris Cremer, Quaid Morris & David Duvenaud, Reinterpreting Importance Weighted Autoencoders.
[3] Carl Doersch, Tutorial on Variational Autoencoders
[4] Slides by Maximillian Soelch for Inference in Latent Variables