Mitigating Overfitting in Variational Auto-Encoders with Weighted Self-Augmentation

Jiachun Jin

ShanghaiTech University

Abstract

Variational Auto-Encoders (VAEs) are prone to overfitting, particularly evident on simple datasets with limited training data. In this paper, we introduce Weighted Self-Augmentation (WSA) as a solution aimed at augmenting VAE training with self-generated samples. To address concerns about out-of-distribution samples potentially altering the distribution modeled by the VAE and adversely impacting training, we propose an additional likelihood-free importance weighting mechanism. This approach utilizes a discriminator to distinguish between true and generated data, effectively reducing biases introduced by generated samples. Different from traditional data augmentation methods, our approach operates independent of any prior dataset knowledge and seamlessly complements existing augmentation techniques. We performed experiments on several datasets and show that our methods can significantly mitigate VAE's model overfitting and directly lead to improved performance when it is applied to lossless compression with bits back coding. The code is available at this link.

1 Introduction

The Variational Auto-Encoders (VAEs) [Kingma and Welling, 2013, Rezende et al., 2014] are a popular class of likelihood-based generative models with widespread applications in image generation [Razavi et al., 2019], text generation [Bowman et al., 2015], music generation [Roberts et al., 2018] and molecules generation [Liu et al., 2018]. VAEs offer multiple perspectives for understanding them, and apart from performing generative tasks, they can also be applied in various tasks such as representation learning [Tschannen et al., 2018], semi-supervised learning [Kingma et al., 2014], anomaly detection [Zimmerer et al., 2018], and lossless compression [Townsend et al., 2019]. Despite training a VAE is an unsupervised learning problem, overfitting will still happen when the distribution being modeled is simple and the training data is not abundant, this hurts the ability of a VAE to generalize effectively to unseen data on non-generative tasks.

The study of mitigating overfitting in deep neural networks (DNNs) has been a long-standing endeavor, and regularization-based techniques such as dropout [Srivastava et al., 2014], batch normalization [Ioffe and Szegedy, 2015] and layer normalization [Ba et al., 2016] are wide adopted, especially in classification problems [Krizhevsky et al., 2012]. Different from regularization-based techniques, data augmentation approaches overfitting from the root of the problem. The overall principle is to artificially infate the training dataset size with some prior knowledge about the input invariances of the data [Shorten and Khoshgoftaar, 2019]. For example, practitioners always apply transformations like random rotations, flips and Gaussian noise injection to the original training data to generate more data from the existing data. The rationale is we actually know these kind of operations can increase the abundance of the training data but the (conditional) distribution that the DNN learns is not changed. Instead of using a set of predefined transformations, data augmentation can also be carried out by generative models. Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] was used for data augmentation in meta-learning [Antoniou et al., 2017] and multi-class classification problems [Grover et al., 2019]. However, applying data augmentation to generative models is much trickier [Jun et al., 2020], because preserving the integrity of the training distribution demands more effort compared to just maintaining the conditional distribution of the labels.

In this work, we propose to mitigate model overfitting in VAEs to achieve high test likelihood at the end of the training. Although a high test likelihood does not direcly indicate good performance on sample generation or representation learning, it directly indicates high compression rate when the VAE is used to conduct lossless compression [Yang et al., 2023] with bits back coding [Hinton and Van Camp, 1993], where the test Evidence Lower Bound (ELBO) equals to the expected code length [Kingma et al., 2019, Zhang et al., 2022]. Our motivation comes from this kind of multifacetedness of the VAE [Yu, 2020], in scenarios that the final goal is not to generate realistic samples, we can still leverage the fact

that VAE is a generative model and we can exploit the value of the self-generated data to mitigate model overfitting. However, during training, there is always a gap between the model distribution $p_\theta(\boldsymbol{x})$ and the true data distribution $p_{\text{data}}(\boldsymbol{x})$. And even if the VAE can model $p_{\text{data}}(\boldsymbol{x})$ well in the later stage of training, it can still produce distorted sample because of the mismatching between the prior and the aggregated posterior distribution [Makhzani et al., 2015, Zhao et al., 2017, Hao and Shafto, 2023]. So naively taking the generated samples as a training sample will introduce bias into the model and hurt the training. To this end, we propose to importance weight each generated sample before conducting data augmentation with it. The optimal weight of a generated sample $\tilde{\boldsymbol{x}}$ should be the density ratio between $p_{\text{data}}(\tilde{\boldsymbol{x}})$ and $p_\theta(\tilde{\boldsymbol{x}})$. This enables the model to adaptively decide to what extend the training should utilize the generated samples. Because of the intractability of both of them, it is approximated by a binary probabilistic classifier between the true data and the generated data, as the discriminator in Generative Adversarial Networks (GANs) [Goodfellow et al., 2014]. We name our proposed approach as Weighted Self-Augmentation (WSA), and the outline of our approach is visualized in figure 1.

For empirial studies, we demonstrate the ability of WSA to mitigate model overfitting on three datasets to which VAEs can easily overfit. We also show this leads to improved compression rate when the VAE is applied to lossless compression with bits back coding. We further demonstrate that for more complex datasets, WSA can also be safely employed due to its inherent safeguard mechanism, which can protect the model from the detrimental effects of low-quality samples. When some prior knowledge about the data is available, WSA can be applied together with other data augmentation techniques to further improve the generalization performance.

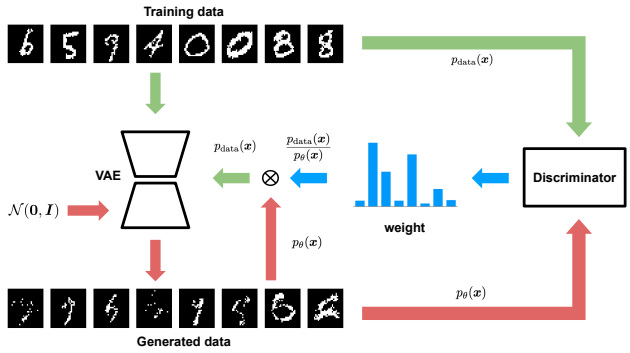


Figure 1: Outline of training a VAE with Weighted Self-Augmentation.

2 Methods

2.1 Overfitting in VAE

VAEs are a class of latent variable models with structures such as $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$, where $\boldsymbol{x}$ denotes an observation and $\boldsymbol{z}$ is the associated latent variable with prior $p(\boldsymbol{z})$. And $p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = \text{EF}(\boldsymbol{x} \mid f_\theta(\boldsymbol{z}))$, sometimes called the decoder, is a conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{z}$, and is always designed to be an exponential family distribution, its natural parameter $f_\theta(\boldsymbol{z})$ is a nonlinear function of $\boldsymbol{z}$, usually parameterized by a neural network with parameter θ. The evalution of the log likelihood $\log p_\theta(\boldsymbol{x}) = \log \int p_\theta(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})\mathrm{d}\boldsymbol{z}$ involves solving an intractable integration over $\boldsymbol{z}$, thus VAEs cannot be trained with standard maximum likelihood estimation (MLE). The training of VAEs avoids this intractability by instead maximizing a tractable lower bound of the log likelihood, which is commonly called as the Evidence Lower Bound (ELBO)

$$\log p_\theta(\boldsymbol{x}) \geq \log p_\theta(\boldsymbol{x}) - \mathbb{D}_{\text{KL}}\left[q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p_\theta(\boldsymbol{z} \mid \boldsymbol{x})\right] = \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right]}_{\text{ELBO}(\boldsymbol{x},\theta,\phi)}, \tag{1}$$

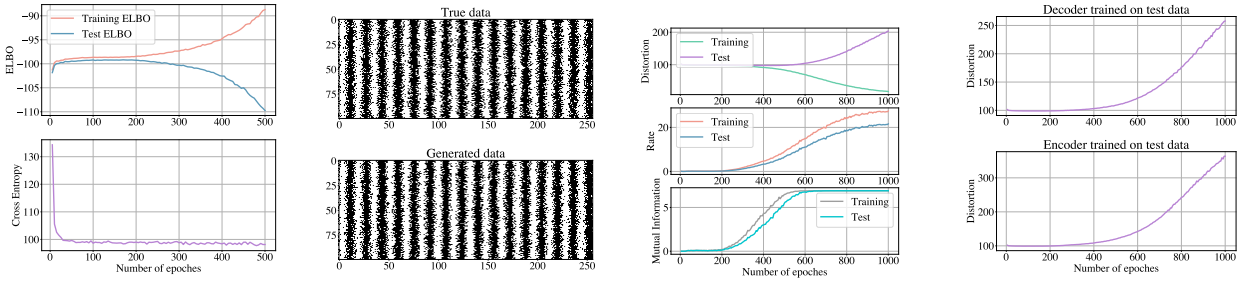
where $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$, sometimes called the encoder, is an approximate posterior distribution over the latent variables, parameterized by another neural network with parameter ϕ. The training of VAE aims to simulatanously optimize θ and ϕ over $p_{\text{data}}(\boldsymbol{x})$ with the reparameterization trick [Kingma and Welling, 2013]

$$\max_{\theta,\phi} \quad \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\text{ELBO}(\boldsymbol{x}, \theta, \phi)\right]. \tag{2}$$

In reality, we have no access to $p_{\text{data}}(\boldsymbol{x})$, but only a finite number of samples from it, collected in the training dataset $\mathcal{D}_{\text{train}} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$. Thus the training objective is approximated by the following Monte Carlo approximation of equation 2

$$\frac{1}{N}\sum_{n=1}^{N} \text{ELBO}(\boldsymbol{x}_n, \theta, \phi), \quad \boldsymbol{x}_n \sim \mathcal{D}_{\text{train}}, \tag{3}$$

and when the number of samples in the training dataset N is small, overfitting can happen that the training ELBO keep increasing while the test ELBO reversely decreases.



(a) (b) (c) (d)

Figure 2: (a) Top: Training ELBO and test ELBO on the toy dataset. Bottom: Estimated cross entropy $\mathbb{E}_{p_\theta(\boldsymbol{x})}[-\log p_{\text{data}}(\boldsymbol{x})]$. (b) Top: Samples from the $p_{\text{data}}(\boldsymbol{x})$. Bottom: Generated samples from $p_\theta(\boldsymbol{x})$. (c) Top: Change of Distortion. Middle: Change of Rate. Bottom: Change of mutual information between X and Z. (d) Distortion on test data. Top: VAE with decoder trained on test data, encoder trained on training data. Bottom: VAE with encoder trained on test data, decoder trained on training data.

To study this in detail, we created a toy dataset. The true data distribution $p_{\text{data}}(\boldsymbol{x})$ is set to be a 256-dimensional Bernoulli distribution, the parameter of the ith dimension is $0.5\sin(i\pi/8) + 0.5$, samples are shown in figure 2(b). We sample 1000 samples from $p_{\text{data}}(\boldsymbol{x})$ as the training set and another 1000 samples as the test set. We trained a VAE for 500 epoches with batch size set to 200 and evaluated the test ELBO every 5 epoches. The result is shown in figure 2(a), dramatic overfitting on the ELBO can be observed. We also estimate the cross entropy between the model distribution and the data distribution $\mathbb{E}_{p_\theta(\boldsymbol{x})}[-\log p_{\text{data}}(\boldsymbol{x})]$ with the generated samples. We can find out in figure 2(b) that the quality of the generated samples does not degrade with the test ELBO, and the samples generated by the VAE trained for 500 epoches shown in figure 2(b) further validated this[1].

We further train the VAE to 1000 epoches and decompose the ELBO into Rate and Distortion as in [Alemi et al., 2018], where

$$\begin{aligned} \text{ELBO}(\boldsymbol{x}, \theta, \phi) &= -\big(\text{Distortion}(\boldsymbol{x}, \theta, \phi) + \text{Rate}(\boldsymbol{x}, \phi)\big), \\ \text{Distortion}(\boldsymbol{x}, \theta, \phi) &= -\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})\right], \\ \text{Rate}(\boldsymbol{x}, \phi) &= \mathbb{D}_{\text{KL}}\left[q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z})\right]. \end{aligned} \tag{4}$$

Distortion measures the ability of the VAE to reconstruct data, and Rate measures the complexity of the latent distribution. From figure 2(c), we can see that only the Distortion term behaves inconsistently during training and test, and the mutual information $I(X; Z)$ between the observation and the latent variable does not drop with Distortion. This indicates that the degradation at Distortion does not result from the decoder forgetting the latent variable as the case in posterior collapse [Lucas et al., 2019], but from the poor generalization ability of the VAE. We then train another VAE on the test data with parameter $\hat{\theta}$ and $\hat{\phi}$, and substitude either the encoder with $q_{\hat{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})$ or the decoder with $p_{\hat{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$ in the original VAE, the evaluated Distortion on test data shown in figure 2(d) indicates that overfitting does not occur in only one of them, but both of them.

We learn from the above toy experiment that both the encoder and decoder of the VAE are vulnerable to model overfitting. In terms of sample generation, an overfitted VAE can still produce reasonable samples, although it may just producing the memorized training samples. In terms of representation learning, an overfitted VAE has performance drop on its ability to reconstruct the input, but this does not lead to dropped mutual information between X and Z, which is different from posterior collapse.

[1] Note this does not indicate our model distribution is approaching the true data distribution since we do not know how the entropy of $p_\theta(\boldsymbol{x})$ changes.

The most direct problem brought by an overfitted VAE is when we want to use it to conduct lossless compression with bits back coding [Townsend et al., 2019]. Because in this case, the compression rate is directly proportional to the negative ELBO on the test data, and mitigating VAE's overfitting is essential in this scenario.

2.2 Self-Augmented Training for VAE

Mitigating model overfitting with some data augmentation is a common practice. Effective data augmentation is always conducted by some pre-defined transformations $T(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$, which adds diversity to the training data while keeping the distribution being modeled unchanged. In this way, we can learn our VAE with

$$\max_{\theta,\phi} \quad \sum_{n=1}^{N} \Big(\mathrm{ELBO}(\boldsymbol{x}_n, \theta, \phi) + \mathbb{E}_{T(\tilde{\boldsymbol{x}}|\boldsymbol{x}_n)} \left[\mathrm{ELBO}(\tilde{\boldsymbol{x}}, \theta, \phi) \right] \Big). \tag{5}$$

However, designing a useful transformation T relies some prior knowledge about the dataset. For example, when we want to do data augmentation on the MNIST dataset, we can apply some degree of rotation to the original data, but we will not apply any horizontal flip to the data, since we know beforehand that the former transformation does not change the data distribution but the latter does. The different effect of these two kinds of data augmentation is shown in figure 3. For other datasets, e.g., our toy dataset, this kind of knowledge is not always available, and the situation will become trickier when dealing with non-image data.

Other than getting augmented data $\tilde{\boldsymbol{x}}$ from hard-coded transformation from the original training data, we can get unlimited amount of data by sampling from the VAE. Sampling from a VAE is quite efficient compared with sampling from other generative models such as energy-based models [Song and Kingma, 2021] or score-based models [Song et al., 2020]. So a direct idea is to train our VAE with both real data and the self-generated data

$$\max_{\theta,\phi} \quad \frac{1}{N} \sum_{n=1}^{N} \mathrm{ELBO}(\boldsymbol{x}_n, \theta, \phi) + \mathbb{E}_{p_\theta(\tilde{\boldsymbol{x}})} \left[\mathrm{ELBO}(\tilde{\boldsymbol{x}}, \theta, \phi) \right]. \tag{6}$$

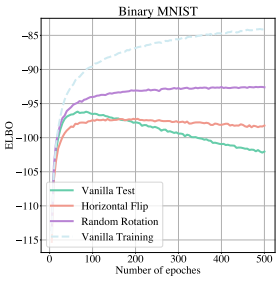


Figure 3: Different behaviours of the test ELBO on binary MNIST because of different choices of hard-coded data augmentation.

However, even though as a generative model, the VAE is trying to model the data distribution by minimizing $\mathbb{D}_{\mathrm{KL}} \left[p_{\mathrm{data}}(\boldsymbol{x}) \parallel p_\theta(\boldsymbol{x}) \right]$, there is always an unnegligable gap between our model distribution and the true data generating distribution during training. Even if we apply this kind of augmentation after a long training process, when we think our model can approximate the data distribution well, there is still issues like the prior hole problem [Rezende and Viola, 2018], that we can often sample un-realistic out-of-distribution samples because of the mismatch between our prior and the aggregated posterior. To this sense, we propose to reduce the bias introduced by the generated samples by importance weighting each sample $\tilde{\boldsymbol{x}}$, by the density ratio $w(\tilde{\boldsymbol{x}}) = p_{\mathrm{data}}(\tilde{\boldsymbol{x}}) / p_\theta(\tilde{\boldsymbol{x}})$. So what we actually do is performing importance sampling from the true data distribution with the model distribution as a proposal distribution. In this sense, our objective becomes to

$$\begin{aligned}
&\max_{\theta,\phi} \quad \frac{1}{N} \sum_{n=1}^{N} \mathrm{ELBO}(\boldsymbol{x}_n, \theta, \phi) + \mathbb{E}_{p_\theta(\tilde{\boldsymbol{x}})} \left[w(\tilde{\boldsymbol{x}}) \, \mathrm{ELBO}(\tilde{\boldsymbol{x}}, \theta, \phi) \right] \\
&= \quad \frac{1}{N} \sum_{n=1}^{N} \mathrm{ELBO}(\boldsymbol{x}_n, \theta, \phi) + \mathbb{E}_{p_{\mathrm{data}}(\tilde{\boldsymbol{x}})} \left[\mathrm{ELBO}(\tilde{\boldsymbol{x}}, \theta, \phi) \right] \\
&\approx \quad \frac{1}{N} \sum_{n=1}^{N} \mathrm{ELBO}(\boldsymbol{x}_n, \theta, \phi) + \frac{1}{N_a} \sum_{n=1}^{N_a} w(\tilde{\boldsymbol{x}}_n) \, \mathrm{ELBO}(\tilde{\boldsymbol{x}}_n, \theta, \phi), \quad \boldsymbol{x}_n \sim \mathcal{D}_{\mathrm{train}}, \quad \tilde{\boldsymbol{x}}_n \sim p_\theta(\tilde{\boldsymbol{x}}),
\end{aligned} \tag{7}$$

where N_a denotes the number of samples we sample from our model. In this way, we can enable our VAE to "see" more training data and correct the bias introduced by sampling from the wrong distribution.

2.3 Weighted Self-Augmentation for VAE

However, since we do not have the density function for the true data distribution and our VAE's model distribution, we cannot compute the density ratio explicitly. Instead, we can build the density ratio estimator via a probabilistic binary

classifier $D_\eta : \mathcal{X} \to [0, 1]$ with parameter η , which predicts the probability that $\boldsymbol{x}$ comes from the data distribution rather than the model distribution [Bickel et al., 2007, Huszár, 2017]. By assigning a pseudo label $y = 1$ to samples from $p_{\text{data}}(\boldsymbol{x})$ and $y = 0$ to samples from $p_\theta(\boldsymbol{x})$, we can define a mixture distribution

$$p_{\text{data},\theta}(\boldsymbol{x}) = p(y = 1)p_{\text{data}}(\boldsymbol{x}) + p(y = 0)p_\theta(\boldsymbol{x}), \tag{8}$$

the Bayesian optimal classifier which outputs the posterior of $y = 1$ is given by

$$D^*(\boldsymbol{x}) = p_{\text{data},\theta}(y = 1 \mid \boldsymbol{x}) = \frac{p_{\text{data},\theta}(\boldsymbol{x} \mid y = 1)p(y = 1)}{p_\theta(\boldsymbol{x})p(y = 0) + p_{\text{data}}(\boldsymbol{x})p(y = 1)} = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + \frac{p(y=0)}{p(y=1)}p_\theta(\boldsymbol{x})}, \tag{9}$$

$p(y = 0)/p(y = 1)$ is the prior ratio between the two classes of samples, when the amounts of samples from $p_{\text{data}}(\boldsymbol{x})$ and $p_\theta(\boldsymbol{x})$ are equal, this term is 1. And in this case, we have

$$D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_\theta(\boldsymbol{x})} = \sigma\left(\log \frac{p_{\text{data}}(\boldsymbol{x})}{p_\theta(\boldsymbol{x})}\right), \tag{10}$$

here $\sigma(\cdot)$ is the *logistic sigmoid* function. With this in mind, we can learn the density ratio estimator indirectly by training our classifier D_η with

$$\min_\eta \quad -\frac{1}{2}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\log D_\eta(\boldsymbol{x})\right] - \frac{1}{2}\mathbb{E}_{p(\boldsymbol{z})}\left[\mathbb{E}_{p_\theta(\tilde{\boldsymbol{x}}|\boldsymbol{z})} \log\left(1 - D_\eta(\tilde{\boldsymbol{x}})\right)\right], \tag{11}$$

and the density ratio estimator can be built by

$$w_\eta(\tilde{\boldsymbol{x}}) = \exp\left(\sigma^{-1}(D_\eta(\tilde{\boldsymbol{x}}))\right) = \frac{D_\eta(\tilde{\boldsymbol{x}})}{1 - D_\eta(\tilde{\boldsymbol{x}})}. \tag{12}$$

Note that every time we update the parameter of the VAE, the model distribution $p_\theta(\boldsymbol{x})$ will change, and we should train the discriminator again, this is expensive. So we instead alternatively train our VAE and the discriminator for every batch of training data. In other words, we first fix the parameters of the VAE, and generate the same number of fake data as the batch size to ensure $p(y = 0)/p(y = 1) = 1$ in equation 9, then train η with the generated and true data with equation 11. Next, we fix η and train our VAE with

$$\max_{\theta,\phi} \quad \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\text{ELBO}(\boldsymbol{x}_n, \theta, \phi)\right] + \mathbb{E}_{p(\boldsymbol{z})}\left[\mathbb{E}_{p_\theta(\tilde{\boldsymbol{x}}|\boldsymbol{z})}\left[\frac{D_\eta(\boldsymbol{x})}{1 - D_\eta(\boldsymbol{x})}\,\text{ELBO}(\tilde{\boldsymbol{x}}, \theta, \phi)\right]\right]. \tag{13}$$

In our implementation, we do not carry out WSA from the start since the generated samples are poor during the early stage and the discriminator also needs some epoches of training to be reliable. So at the beginning of training, we train the VAE and discriminator but we do not feed the weighted samples back to the VAE. The pseudocode is given in algorithm 1.

Algorithm 1: Training VAE with WSA

Input : a batch of training data $\mathcal{B}$, # of epoches to warm up T_{warmup}.

if current epoch $< T_{\text{warmup}}$ **then**

1. Train VAE on $\mathcal{B}$ with equation 3.
2. Sample a batch of fake data from the VAE.
3. Train the discriminator with $\mathcal{B}$ and the fake data with equation 11.

end

else

1. Sample a batch of fake data from the VAE.
2. Calculate the importance weight $w(\tilde{\boldsymbol{x}})$ for each generated data with equation 12.
2. Train VAE on both $\mathcal{B}$ and the weighted fake data with equation 13.
3. Train the discriminator with $\mathcal{B}$ and the fake data with equation 11.

end

3 Experiments

3.1 Experimental Setting

We first show that training a VAE via WSA can alleviate model overfitting on statically binarized MNIST[2], statically binarized OMNIGLOT[3] and the Caltech 101 Silhouettes[4], VAEs are prone to be affected by overfitting on these datasets. The latent dimension of our VAE is set to 16, and both the encoder and the decoder are MLPs with two layers of 500 hidden units with ReLU activation. The discriminator D_η is built by a convolutional neural network with 3 layers. We start to apply WSA after 50 epoches of warmup. We compare our proposed WSA with the following methods:

1. Vanilla VAE. The VAE is trained by simply maximizing the ELBO as in equation 3.
2. Dropout regularization [Srivastava et al., 2014]. Dropout layers with probability p after the ReLU activation are added in both the VAE's encoder and the decoder. We report the best result with p chosen from the set $\{0.05, 0.1, 0.2, 0.3\}$.
3. Amortized inference regularization by denoising (AIR) [Shu et al., 2018]. The VAE's decoder is trained as the vanilla VAE with equation 3, and for a training point $\boldsymbol{x}$, the encoder is trained by maximizing

$$\alpha \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x}+\epsilon)}\left[\log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{z}+\epsilon)}\right] + (1-\alpha)\,\mathrm{ELBO}(\boldsymbol{x}, \theta, \phi),$$

   where α is a mixing coefficient and we fix it to 0.5, and $\epsilon \sim \mathcal{N}(0, \rho^2 \boldsymbol{I})$ is some zero-mean Gaussian noise with standard deviation ρ. We report the best result with ρ chosen from the set $\{0.1, 0.2, 0.4, 0.8\}$.
4. Reversed half-asleep (RHA) [Zhang et al., 2022]. Like AIR, VAE's decoder is trained with equation 3, and train the encoder with a mixture distribution of the true and the generated distribution, without a weighting term for the generated data, as in equation 6. To prevent the training process from being disrupted by distorted samples, the authors suggested applying RHA in a post-hoc fashion, that a vanilla VAE is firstly trained, and the self-generated samples are only used to further train the saved models. However, we find that on the datasets mentioned above, applying RHA during training is still feasible, since they are comparatively easily for the VAE to learn and generate high quality samples.

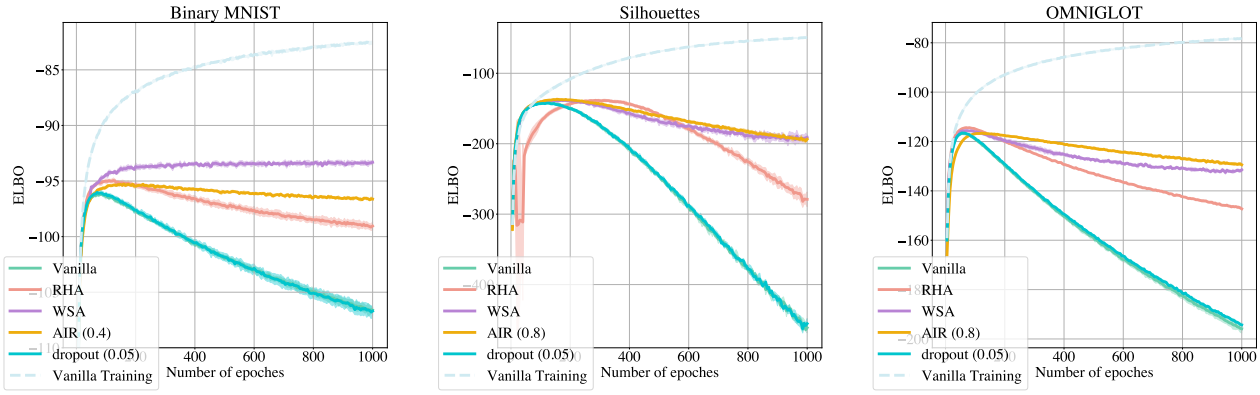
3.2 Experiments on Generalization Performance

We train the VAEs for 1000 epoches with Adam [Kingma and Ba, 2014], the learning rate is set to 5×10^4. The test ELBO is evaluated every 5 epoches, and is shown in figures 4(a), 4(b) and 4(c). We can see our proposed method can significantly mitigate overfitting compared to simply training a VAE with vanilla ELBO maximization. WSA can always outperform RHA, we think this is because RHA only use the generated samples to train the encoder, however, overfitting also happens in the decoder. We can also observe that the test ELBO of RHA on the Silhouettes data has a large variance at the beginning of the training, we think this results from some low-quality samples unstablizing the training, our proposed WSA can avoid this because of the usage of the discriminator. Simple dropout regularizations seems has not much effect on mitigating overfitting in our cases. The most effective comparing approach is AIR, AIR with $\rho = 0.4$ beats other comparing methods but is still inferior to WSA on the binary MNIST dataset. On Silhouettes, AIR with $\rho = 0.8$ is as effective as WSA and outperforms all the others, and on OMNIGLOT, AIR with $\rho = 0.8$ outperforms all the comparing approaches including WSA. However, tuning the hyperparameter ρ is cumbersome and always requires an extra validation set. In our experiments, we did not use an extra validation set, but just reported the best result of AIR with different choices ρ, this is somehow unfair to WSA. WSA has no hyperparameter to tune and in this sense is easy to deploy.

3.3 Application of Lossless Compression

Latent variable models can be used to carry out lossless compression with bits back coding [Hinton and Van Camp, 1993]. And the Bits Back with ANS (BB-ANS) framework [Townsend et al., 2019, Duda, 2013] allows us to implement

[2] link to this
[3] link to this
[4] link to this



(a) (b) (c)

Figure 4: (a) Test ELBOs on binary MNIST. (b) Test ELBOs on Silhouettes. (c) Test ELBOs on OMNIGLOT.

Dataset	Epoches	Vanilla	Dropout (best p)	AIR (best ρ)	RWS	WSA
Binary MNIST	200	0.1813 ± 0.0002	0.1813 ± 0.0002	**0.1777 ± 0.0002**	0.1771 ± 0.0003	**0.1747 ± 0.0003**
	600	0.1904 ± 0.0003	0.1902 ± 0.0006	**0.1787 ± 0.0001**	0.1808 ± 0.0005	**0.1738 ± 0.0003**
	1000	0.1965 ± 0.0003	0.1965 ± 0.0007	**0.1793 ± 0.0002**	0.1831 ± 0.0006	**0.1736 ± 0.0003**
OMNIGLOT	200	0.2380 ± 0.0012	0.2368 ± 0.0003	**0.2179 ± 0.0002**	0.2213 ± 0.0003	**0.2210 ± 0.0014**
	600	0.2949 ± 0.0017	0.2927 ± 0.0009	**0.2285 ± 0.0005**	0.2470 ± 0.0005	**0.2362 ± 0.0002**
	1000	0.3323 ± 0.0022	0.3297 ± 0.0003	**0.2364 ± 0.0008**	0.2629 ± 0.0007	**0.2409 ± 0.0011**
Silhouettes	200	0.2718 ± 0.0029	0.2722 ± 0.0031	**0.2546 ± 0.0006**	0.2656 ± 0.0032	**0.2564 ± 0.0029**
	600	0.3998 ± 0.0025	0.3986 ± 0.0003	**0.2961 ± 0.0019**	0.3053 ± 0.0042	**0.3063 ± 0.0038**
	1000	0.4817 ± 0.0022	0.4832 ± 0.0015	**0.3268 ± 0.0018**	0.3961 ± 0.0056	**0.3283 ± 0.0044**

Table 1: Compression rates measured in Bits Per Dimension, the best two results are indicated in bold.

bit back coding with a VAE which usually has continuous latent variables. We plug in the trained VAEs into BB-ANS and carry out lossless compression on the test sets. The code length of a test sample $\boldsymbol{x}$ should be approximately equal to the negative ELBO with log base 2, i.e.

$$\text{code length} = -(\log_2 p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) + \log_2 p(\boldsymbol{z}) - \log_2 q_\phi(\boldsymbol{z} \mid \boldsymbol{x})).$$

We report the compression rates measured in Bits Per Dimension (BPD)[5] in table 1. We can see that the performance of the VAEs on lossless compression aligns well with their performance on generalization, and our proposed WSA even though has no hyperparameter to tune, is still very competitive, always ranks the top two on all three datasets.

3.4 Applied with Other Data Augmentations

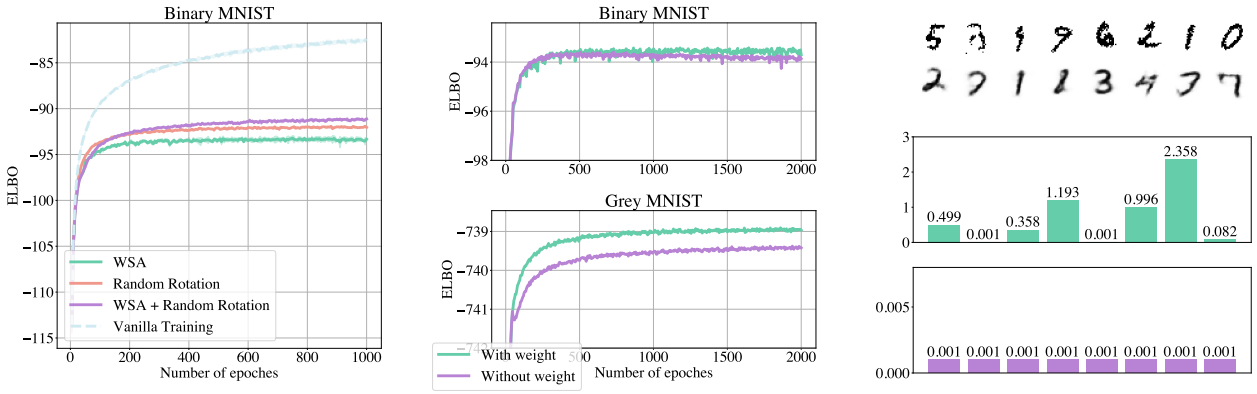
When we do have some prior knowledge about our dataset, e.g., performing some random rotation on these three datasets can add abundance of the training but actually do not change the distribution, then we can perform such kind of data augmentaion accompany with WSA. On binary MNIST, we randomly rotate each batch of data by a random degree chosen from $[-30, 30]$. We show in figure 5(a) that the training can benefit from both the hard-coded random rotation data augmentation and our proposed WSA. This suggests that WSA can be applied as a plug-in and can always bring more data abundance to improve VAE's generalization performance.

3.5 Effect of the Weighting Term

In this subsection, we conduct ablation studies on the effect of the weighting term for generated data samples $w(\tilde{\boldsymbol{x}})$ on binary MNIST and grey MNIST. We use CNN based encoder and decoder for grey MNIST because of the increased dataset complexity. Note that we can switch off the weighting terms by fixing them to 1, in this case, all the generated

[5] BPD = code length$/D$, where D is the dimensionality of the data.

samples are treated as true data samples during the training. The test ELBOs are shown in figure 5(b). We can see on the simple binary MNIST dataset, self augmentation with a weighting term performs slightly better than the one without. However, on grey MNIST, performance degrades for self-augmentation without a weighting term. Figure 5(c) shows the samples generated by the VAEs and their corresponding estimated importance weights by the discriminator. We can see on the relatively simple binary MNIST, the VAE can generate some realistic sample which are assigned by the discriminator with high weights. However, on grey MNIST, the VAE fails to generate samples realistic enough to 'fool' the discriminator, and all the generated samples are assigned near-zero weights, in this case WSA degrades to vanilla VAE training, although little benefits can be obtained from the generated samples, this avoids the VAE from learning a biased data distribution. So the weighting term in WSA can adaptively decide to what extend our training should utilize the generated samples.



(a) (b) (c)

Figure 5: (a) Application of WSA with Random Rotation Augmentation. (b) Test ELBOs for Self-Augmentation training with and without the weighting term on binary MNIST adn grey MNIST. (c) Top: Sample generated from the VAE. Bottom: Weights assigned by the discriminator for each sample.

4 Related Work

Training a generative model using both real data and generated data is known as two phases of learning. The model is trained with real data in the wake phase and with generated data in the sleep phase. The wake-sleep algorithm [Hinton et al., 1995, Bornschein and Bengio, 2014] used the samples generated by a directed latent variable to train an amortized inference network and real data to train the generative network. Training an energy-based model with MLE also requires samples from the model, to provide gradient of the intractable log partition function [Song and Kingma, 2021].

As a generative model, VAEs are less widely adopted to conduct data augmentation because of the less competitive generation performance. Generative Adversarial Networks are widely used to do data augmentation for classification [Perez and Wang, 2017, Yamaguchi et al., 2020] and meta-learning [Antoniou et al., 2017]. With the great success in generative modeling, diffusion models are recently used to conduct data augmentation as well [Shivashankar and Miller, 2023] . Reversely, conducting data augmentation for generative modeling is trickier since we have to take great care to ensure the distribution being modeled in not changed by augmentation [Jun et al., 2020, Tran et al., 2021].

Correcting Bias of a generative model with importance weighting for downstream tasks was also investigated in [Grover et al., 2019], the authors used a binary classifier to reweight samples from a separate generative model for data augmentation for classification and model-based off-policy evaluation. The major difference is in our approach, both the generative model and the target model to train is the VAE.

The closest related work is [Zhang et al., 2022], where the authors study the generalization gap in VAEs and claim that the overfitting in VAEs is dominated by amortized inference. Their solution is to first train a vanilla VAE and then fix the VAE's decoder and further train the encoder with the generated samples. With the weighting term, our proposed WSA sidesteppings the post-hoc training, and we can always train with the self-generated data in situ. Also, we use the weighted self-generated samples to safely train both the encoder and decoder, this further mitigates model overfitting.

5 Conclusion

References

A. Alemi, B. Poole, I. Fischer, J. Dillon, R. A. Saurous, and K. Murphy. Fixing a broken elbo. In International conference on machine learning, pages 159–168. PMLR, 2018.

A. Antoniou, A. Storkey, and H. Edwards. Data augmentation generative adversarial networks. arXiv preprint arXiv:1711.04340, 2017.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In Proceedings of the 24th international conference on Machine learning, pages 81–88, 2007.

J. Bornschein and Y. Bengio. Reweighted wake-sleep. arXiv preprint arXiv:1406.2751, 2014.

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

J. Duda. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding. arXiv preprint arXiv:1311.2540, 2013.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. Advances in neural information processing systems, 27, 2014.

A. Grover, J. Song, A. Kapoor, K. Tran, A. Agarwal, E. J. Horvitz, and S. Ermon. Bias correction of learned generative models using likelihood-free importance weighting. Advances in neural information processing systems, 32, 2019.

X. Hao and P. Shafto. Coupled variational autoencoder. arXiv preprint arXiv:2306.02565, 2023.

G. E. Hinton and D. Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In Proceedings of the sixth annual conference on Computational learning theory, pages 5–13, 1993.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The" wake-sleep" algorithm for unsupervised neural networks. Science, 268(5214):1158–1161, 1995.

F. Huszár. Variational inference using implicit distributions. arXiv preprint arXiv:1702.08235, 2017.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning, pages 448–456. pmlr, 2015.

H. Jun, R. Child, M. Chen, J. Schulman, A. Ramesh, A. Radford, and I. Sutskever. Distribution augmentation for generative modeling. In International Conference on Machine Learning, pages 5006–5019. PMLR, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. International Conference on Learning Representations, 2013.

D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. Advances in neural information processing systems, 27, 2014.

F. Kingma, P. Abbeel, and J. Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In International Conference on Machine Learning, pages 3408–3417. PMLR, 2019.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 2012.

Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt. Constrained graph variational autoencoders for molecule design. Advances in neural information processing systems, 31, 2018.

J. Lucas, G. Tucker, R. B. Grosse, and M. Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. Advances in Neural Information Processing Systems, 32, 2019.

A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. arXiv preprint arXiv:1511.05644, 2015.

L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621, 2017.

A. Razavi, A. Van den Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. Advances in neural information processing systems, 32, 2019.

D. J. Rezende and F. Viola. Taming vaes. arXiv preprint arXiv:1810.00597, 2018.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In International conference on machine learning, pages 1278–1286. PMLR, 2014.

A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck. A hierarchical latent vector model for learning long-term structure in music. In International conference on machine learning, pages 4364–4373. PMLR, 2018.

C. Shivashankar and S. Miller. Semantic data augmentation with generative models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 863–873, 2023.

C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of big data, 6(1): 1–48, 2019.

R. Shu, H. H. Bui, S. Zhao, M. J. Kochenderfer, and S. Ermon. Amortized inference regularization. Advances in Neural Information Processing Systems, 31, 2018.

Y. Song and D. P. Kingma. How to train your energy-based models. arXiv preprint arXiv:2101.03288, 2021.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456, 2020.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014.

J. Townsend, T. Bird, and D. Barber. Practical lossless compression with latent variables using bits back coding. arXiv preprint arXiv:1901.04866, 2019.

N.-T. Tran, V.-H. Tran, N.-B. Nguyen, T.-K. Nguyen, and N.-M. Cheung. On data augmentation for gan training. IEEE Transactions on Image Processing, 30:1882–1897, 2021.

M. Tschannen, O. Bachem, and M. Lucic. Recent advances in autoencoder-based representation learning. arXiv preprint arXiv:1812.05069, 2018.

S. Yamaguchi, S. Kanai, and T. Eda. Effective data augmentation with multi-domain learning gans. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 6566–6574, 2020.

Y. Yang, S. Mandt, L. Theis, et al. An introduction to neural data compression. Foundations and Trends® in Computer Graphics and Vision, 15(2):113–200, 2023.

R. Yu. A tutorial on vaes: From bayes' rule to lossless compression. arXiv preprint arXiv:2006.10273, 2020.

M. Zhang, P. Hayes, and D. Barber. Generalization gap in amortized inference. Advances in Neural Information Processing Systems, 35:26777–26790, 2022.

S. Zhao, J. Song, and S. Ermon. Infovae: Information maximizing variational autoencoders. arXiv preprint arXiv:1706.02262, 2017.

D. Zimmerer, S. A. Kohl, J. Petersen, F. Isensee, and K. H. Maier-Hein. Context-encoding variational autoencoder for unsupervised anomaly detection. arXiv preprint arXiv:1812.05941, 2018.