LEARNING PRIORS FOR ADVERSARIAL AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most deep latent factors choose simple priors for simplicity, tractability or not knowing what prior to use. Recent studies show that the choice of the prior may have a profound effect on the expressiveness of the model, especially when the generation network has limited capacity. In this paper, we propose to learn a proper prior from data for AAE. We introduce the notion of code generators to transform manually selected simple priors into one that can better fit the data distribution. Experimental results show that the proposed model can generate better image quality and learn better disentangled representations than AAE in both supervised and unsupervised settings. Lastly, we present its ability to do cross-domain translation in a text-to-image synthesis task.

1 INTRODUCTION

Deep latent factor models, such as variational autoencoders (VAE) and adversarial autoencoders (AAE), are becoming increasingly popular in various tasks, such as image generation (Larsen et al., 2015), unsupervised clustering (Dilokthanakul et al. (2016), Makhzani et al. (2015)), and cross-domain translation (Wu et al., 2016). These models involve a prior distribution over latent variables and a generation distribution describing the stochastic mapping from latent variables to visible variables. Traditionally, simple priors, such as Gaussian (Kingma & Welling, 2013) or Gaussian mixture distributions (Dilokthanakul et al., 2016), are chosen for tractability, simplicity, or not knowing what prior to use, with the hopes thwilbe transformed somewhere in the deep generation network into a form suitable for characterizing the distribution of visible data. While this might hold true when the generation network has enough capacity, some recent works (Hoffman & Johnson, 2016), (Goyal et al., 2017) suggest that the choice of the prior may have a profound impact on the expressiveness of the model and that poorly selected priors may lead to undesired regularization effects.

Motivated by this observation, we propose in this paper the notion of code generators for learning a prior from data for AAE. The objective is to learn a code generator network to transform a simple prior into one that, together with the generation network, can better fit the data distribution. To this end, we introduce another adversarial loss in AAE for training the code generator. Extensive experiments on MNIST and CIFAR-10 datasets confirm its effectiveness of generating better quality images and learning better disentangled representations in both supervised and unsupervised settings. Lastly, we present its application to text-to-image synthesis.

The remainder of this paper is organized as follows: Section 2 reviews the background and related work. Section 3 presents the implementation details and training process of the proposed code generator. Section 4 compares its performance with AAE in image generation and disentanglement tasks. Lastly, we conclude this paper with remarks on future work.

2 BACKGROUND AND RELATED WORK

A latent factor model is a probabilistic model for describing the relationship between a set of latent and visible variables. The model is usually specified by a prior distribution $p(z)$ over the latent variables z and a conditional distribution $p(x|z;\theta)$ of the visible variables x given the latent variables z. The model parameters θ are often learned by maximizing the marginal log-likelihood of the data $\log p(x;\theta)$.

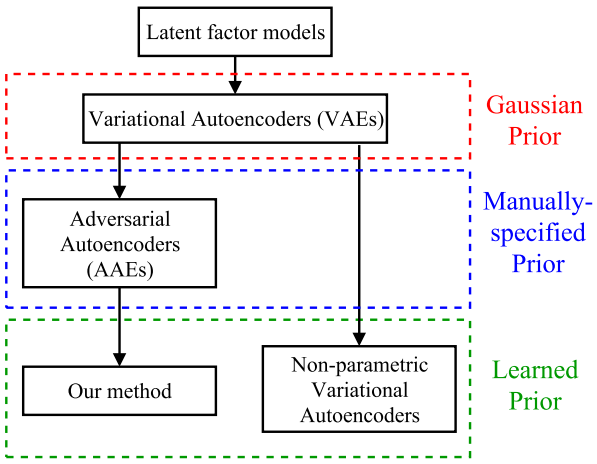


Figure 1: The relations of our work with prior arts.

Variational Autoencoders (VAE). To improve the model's expressiveness, it is common to make deep the conventional latent factor models by introducing a neural network to $p(x|z;\theta)$. One celebrated example is VAE (Kingma & Welling, 2013), which assumes the following prior $p(z)$ and $p(x|z;\theta)$:

$$\begin{aligned} p(z) &\sim \mathcal{N}(z;0,I) \\ p(x|z;\theta) &\sim \mathcal{N}(x;o(z;\theta),\sigma^2 I) \end{aligned} \tag{1}$$

where the mean $o(z;\theta)$ is modeled by the output of a neural network with parameters θ. In this case, the marginal $p(x;\theta)$ becomes intractable; the model is thus trained by maximizing the log evidence lower-bound (ELBO):

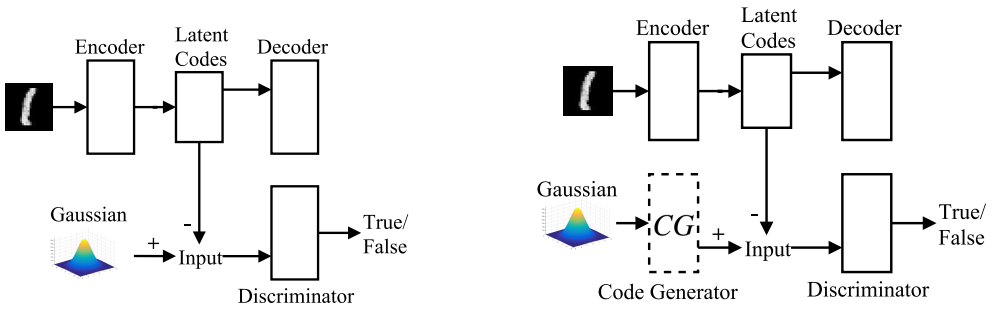
$$\mathcal{L}(\phi,\theta) = E_{q(z|x;\phi)} \log p(x|z;\theta) - KL(q(z|x;\phi) \parallel p(z)) \tag{2}$$

where $q(z|x;\phi)$ is the variational density, implemented by another neural network with parameter ϕ, to approximate the posterior $p(z|x;\theta)$. When regarding $q(z|x;\phi)$ as an (stochastic) encoder and $p(z|x;\theta)$ as a (stochastic) decoder, Equation (2) bears an interpretation of training an autoencoder with the latent code z regularized by the prior $p(z)$ through the KL-divergence.

Adversarial Autoencoders (AAE). Motivated by the observation that VAE is largely limited by the Gaussian prior assumption, i.e., $p(z) \sim \mathcal{N}(z;0,I)$, Makhzani et al. (Makhzani et al., 2015) relax this constraint by allowing $p(z)$ to be any distribution. Apparently, the KL-divergence becomes intractable when $p(z)$ is arbitrary. They thus replace the the KL-divergence with an adversarial loss imposed on the encoder output, requiring that the latent code z produced by the encoder should have an aggregated distribution[1] the same as the prior $p(z)$.

Non-parametric Variational Autoencoders (Non-parametric VAE). While AAE allows the prior to be arbitrary, how to select a prior that can best characterize the data distribution remains an open issue. Goyal et al. (Goyal et al., 2017) make an attempt to learn a non-parametric prior based on the nested Chinese restaurant process for VAE. Their model induces a hierarchical structure of semantic concepts in the latent code space.

Along much the same line of thinking as (Goyal et al., 2017), we propose in this paper the notion of code generator to learn a proper prior from data for AAE. The relations of our work with these prior arts are better illustrated in Figure 1.



(a) AAE

(b) AAE + code generator

Figure 2: The architecture of AAE without (left) and with (right) the code generator.

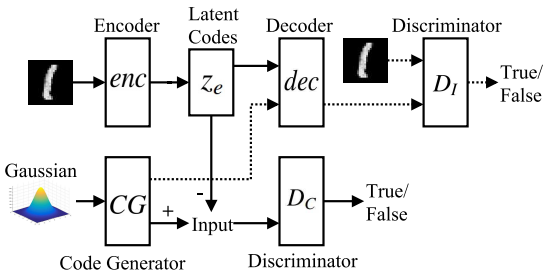


Figure 3: The overall training architecture.

3 Learning The Prior

In this paper, we propose to learn the prior from data instead of specifying it arbitrarily. Built on the foundation of AAE, we introduce a neural network (which we call the code generator) to transform the manually-specified prior into a better form. Figure 2 presents its role in the overall architecture, and contrasts the architectural difference relative to AAE.

Because this code generator itself has to be learned, we need an objective function to shape the distribution at its output. Normally, we wish to find a prior that, together with the decoder in Figure 3, would lead to a distribution that maximizes the data likelihood. We are however faced with two challenges. First, the output of the code generator could be any distribution, which makes the likelihood function and its variational lower bound intractable. Second, the decoder has to be learned simultaneously, which creates a moving target for the code generator. To address the first challenge, we propose to impose an adversarial loss on the output of the decoder when training the code generator. That is, we want the code generator to produce a prior that minimizes the adversarial loss at the decoder output. Using the example in Figure 4, the decoder should generate images with a distribution that should in principle match the distribution of real images in the training data, when driven by the output samples from the code generator. In symbols, this is to minimize

$$\mathcal{L}^{I}_{GAN} = \log(D_I(x)) + \log(1 - D_I(dec(z_c))), \tag{3}$$

where $z_c = CG(z)$ is the output of the code generator CG driven by a noise sample $z \sim p(z)$, D_I is the discriminator in image space, and $dec(z_c)$ is the output of the decoder driven by z_c. To address the second challenge, we propose to alternate training of the code generator and the decoder/encoder until convergence. In one phase, termed the *prior improvement phase*. we update the code generator with the loss function in Equation (3), by fixing the encoder[2]. In the other phase, termed the *AAE phase*, we fix the code generator and update the autoencoder following the training procedure of

[1]The aggregated distribution is defined as $q(z) = \int q(z|x;\phi)p_d(x)dx$, where $p_d(x)$ denotes the empirical distribution of the training data

[2]Supposedly, the decoder needs to be fixed in this phase. It is however found beneficial in terms of convergence to update also the decoder.

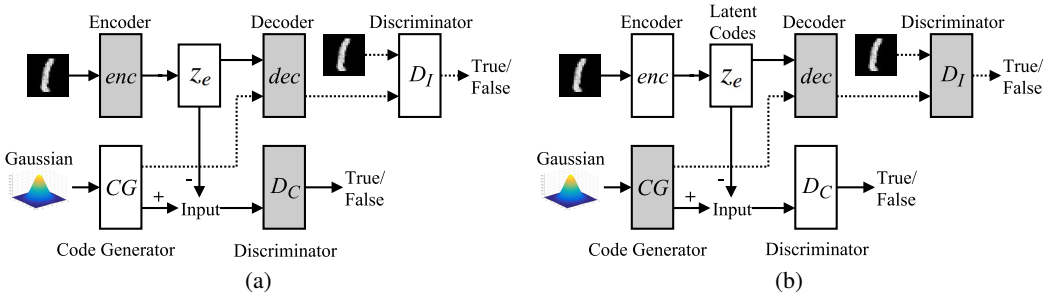


Figure 4: Alternation of training phases: (a) the AAE phase and (b) the prior improvement phase.

Algorithm 1 Training algorithm for AAE with the code generator.

$\theta_{enc}, \theta_{dec}, \theta_{CG}, \theta_{D_I}, \theta_{D_C} \longleftarrow$ Initialize network parameters
Repeat (for each epochs E_i)
  Repeat (for each mini-batch x_j)
    // AAE phase
    $z \sim p(z)$
    $z_c \leftarrow CG(z)$
    $z_e \leftarrow enc(x_i)$
    $\mathcal{L}^C_{GAN} \leftarrow \log(D_C(z_c)) + \log(1 - D_C(z_e))$
    $x_{rec} \leftarrow dec(z_e)$
    $\mathcal{L}_{rec} \leftarrow \|x_j - x_{rec}\|$

    // Update network parameters for AAE phase
    $\theta_{D_C} \leftarrow \theta_{D_C} - \nabla_{\theta_{D_C}}(\mathcal{L}^C_{GAN})$
    $\theta_{enc} \leftarrow \theta_{enc} - \nabla_{\theta_{enc}}(-\mathcal{L}^C_{GAN} + \mathcal{L}_{rec})$
    $\theta_{dec} \leftarrow \theta_{dec} - \nabla_{\theta_{dec}}(\mathcal{L}_{rec})$

    // Prior improvement phase
    $z \sim p(z)$
    $z_c \leftarrow CG(z)$
    $x_{noise} \leftarrow dec(z_c)$
    $\mathcal{L}^I_{GAN} \leftarrow \log(D_I(x_{noise})) + \log(1 - D_I(x_i))$

    // Update network parameters for prior improvement phase
    $\theta_{D_I} \leftarrow \theta_{D_I} - \nabla_{\theta_{D_I}}(\mathcal{L}^I_{GAN})$
    $\theta_{dec} \leftarrow \theta_{dec} - \nabla_{\theta_{dec}}(-\mathcal{L}^I_{GAN})$
    If epoch index i mod λ equals 0 then
      $\theta_{CG} \leftarrow \theta_{CG} - \nabla_{\theta_{CG}}(-\mathcal{L}^I_{GAN})$
    End If
  Until all mini-batches are seen
Until terminate

AAE. Specifically, the encoder output has to be regularized by the following adversarial loss:

$$\mathcal{L}^C_{GAN} = \log(D_C(z_c)) + \log(1 - D_C(enc(x))), \tag{4}$$

where $z_c = CG(z)$ is the output of the code generator, $enc(x)$ is the encoder output given the input x, and D_C is the discriminator in latent code space. In addition, the L_1 norm of the reconstruction error of x needs to be minimized. More details of the training process can be found in Algorithm 1. One point worth mentioning is that a hyper parameter λ is introduced in the *prior improvement phase* to have the code generator updated less frequently to improve stability. It is found that the encoder output cannot be regularized properly if the code generator changes frequently.

4 EXPERIMENTS

We compare the performance of our model with AAE, which adopts manually-specified priors, in image generation and disentanglement tasks. In Section 4.1, we show that using the same encoder and decoder architecture, our model can generate higher quality images. In Section 4.2, we demonstrate that our model can better learn disentangled representations in both supervised and unsupervised settings. Lastly, we present an application of our model to text-to-image synthesis in Section 4.3.

4.1 IMAGE GENERATION

Latent factor models with the priors learned from data rather than specified arbitrarily should ideally better characterize the data distribution. To verify this, we compare the performance of our model with AAE (Makhzani et al., 2015), in terms of image generation. In this experiment, our model learns a GAN-driven prior from data, whereas by convention, AAE assumes a Gaussian prior. For a fair comparison, we require that both models have access to the same encoder and decoder networks, with the network parameters trained to optimize their respective priors.

Figure 5 displays side-by-side images generated from these models when trained on MNIST and CIFAR-10 datasets. They are produced by drawing samples from the priors and passing them through their respective decoders. In this experiment, two observations are immediate. First, our model can generate sharper images than AAE on both datasets. Second, AAE experiences problems in reconstructing visually-plausible images on the more complicated CIFAR-10. These highlight the necessity of learning the prior from data, especially when the capacity of the autoencoder is limited.

Another advantage of our model is its ability to better regularize the distribution of image representations in high-dimensional latent code space. Figure 6 presents images generated by the two models when the dimension of the latent code is increased significantly from 8 to 100 on MNIST, and from 64 to 2000 on CIFAR-10. As compared to Figure 5, it is seen that the increase of code dimension has little impact on our model, but exerts a strong influence on AAE. In the present case, AAE can hardly produce recognizable images, particularly on CIFAR-10, even after the re-parameterization trick has been applied to the output of the encoder as suggested in (Makhzani et al., 2015). This emphasizes the importance of having a prior that can adapt automatically to the code space and the data.

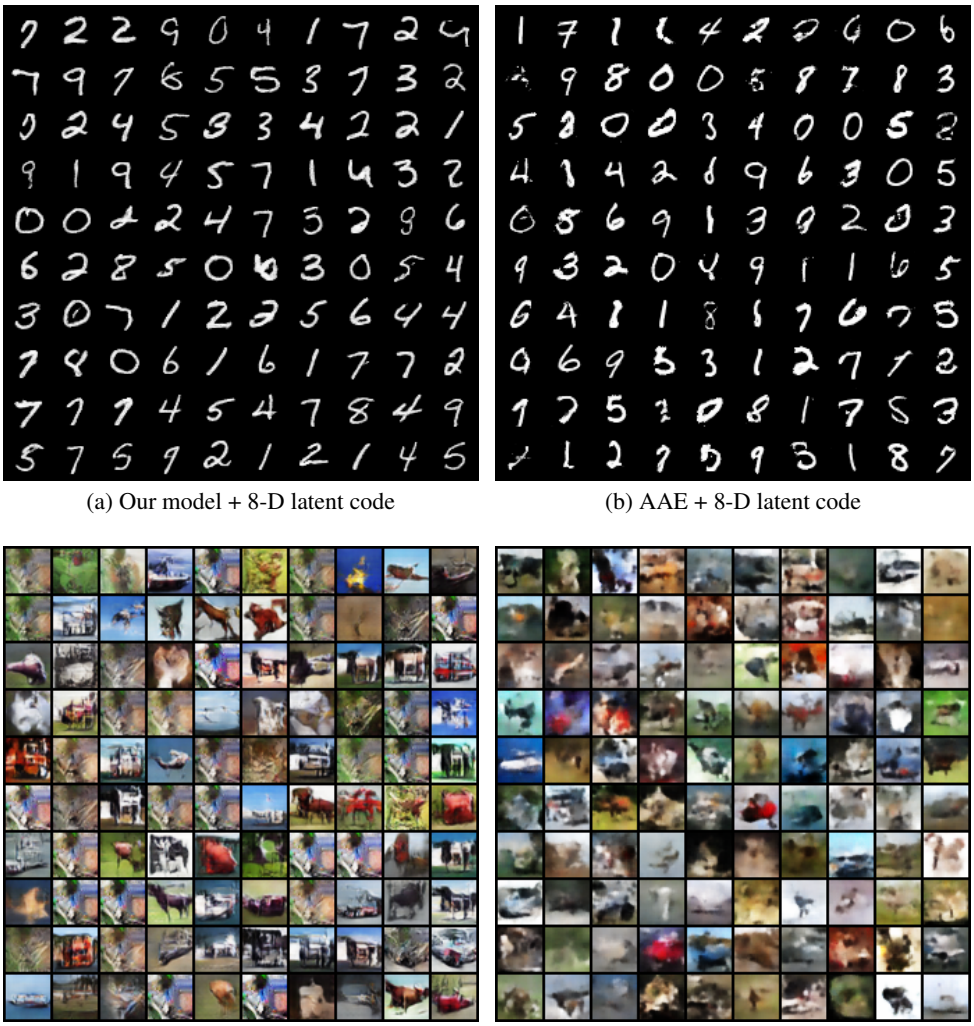
4.2 DISENTANGLED REPRESENTATION

Learning disentangled representation is desirable in many applications. It refers generally to learning a representation whose individual dimensions can capture independent factors of variation in the data. To demonstrate the ability of our model to learn disentangled representations and the merits of GAN-driven priors, we repeat the disentanglement tasks in (Makhzani et al., 2015), and compare its performance with AAE.

4.2.1 SUPERVISED LEARNING

This session presents experimental results of a network architecture that incorporates the GAN-driven prior in learning supervisedly to disentangle the label information of images from the other information. Its block diagram is depicted in Fig. 7, where the code generator takes as input the label information of an image and an independent Gaussian noise to impose a conditional latent code distribution on the image representation. This has an interpretation of associating each class of images with a code space governed by some conditional distribution, which presumably reflects the data variability caused by information other than the label. In particular, this conditional distribution itself needs to be learned from data using the GAN-based training procedure presented in Figure 7. Note that the discriminator in image space must also be conditioned on the label information.

At test time, image generation for a particular class is achieved by inputting the class label and a Gaussian noise to the code generator and then passing the resulting code through the decoder. Fig. 8 displays images generated by our model and AAE trained on MNIST and CIFAR-10 datasets. Both models adopt a 10-D one-hot vector to specify the label and a 54-D Gaussian to generate the noise. To be fair, the output of our code generator has an identical dimension (i.e., 64) to the latent code



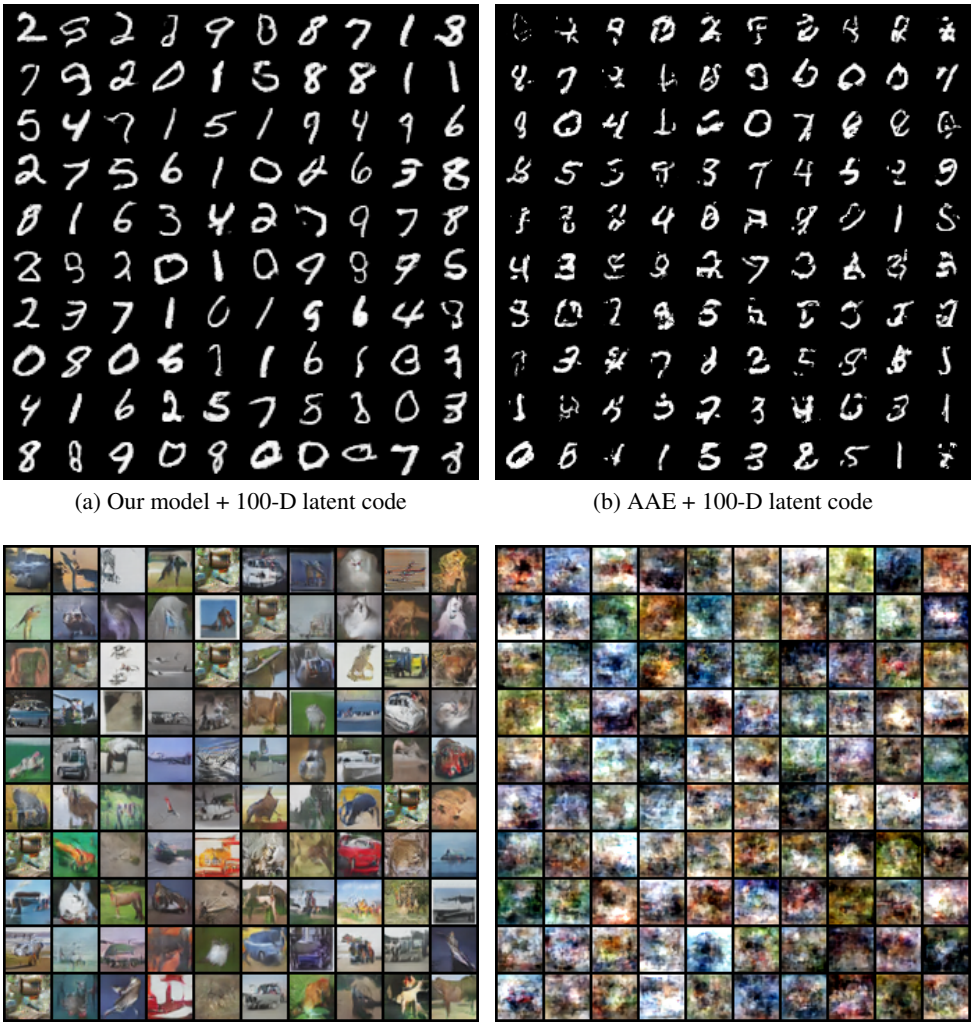
(a) Our model + 8-D latent code

(b) AAE + 8-D latent code

(c) Our model + 64-D latent code

(d) AAE + 64-D latent code

Figure 5: Images generated by our model and AAE trained on MNIST (upper) and CIFAR-10 (lower).



(a) Our model + 100-D latent code

(b) AAE + 100-D latent code

(c) Our model + 2000-D latent code

(d) AAE + 2000-D latent code

Figure 6: Images generated by our model and AAE trained on MNIST (upper) and CIFAR-10 (lower). In this experiment, the latent code dimension is increased significantly to 64-D and 2000-D for MNIST and CIFAR-10, respectively. For AAE, the re-parameterization trick is applied to the output of the encoder as suggested in (Makhzani et al., 2015).

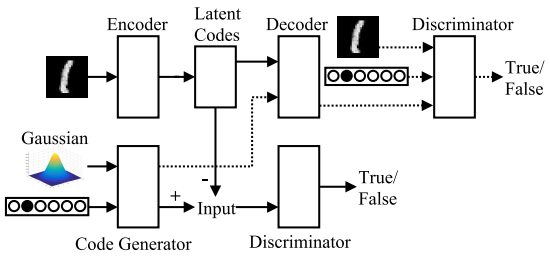


Figure 7: Supervised learning architecture with the code generator.

of AAE. Each row of Fig. 8 corresponds to images generated by varying the label while fixing the noise. Likewise, each column shows images that share the same label yet with varied noise.

On MNIST, both models work well in separating the label information from the remaining (style) information. This is evidenced from the observation that along each row, the digit changes with the label input regardless of the noise variable, and that along each column, the style varies without changing the digit.

On CIFAR-10, the two models behave differently. Ours can still disentangle the class information from the remaining information, even though some classes exhibit mode collapse, which is manifested in producing several closely similar images. In contrast with our model, AAE can not generate sharp images and fail to separate each classes.

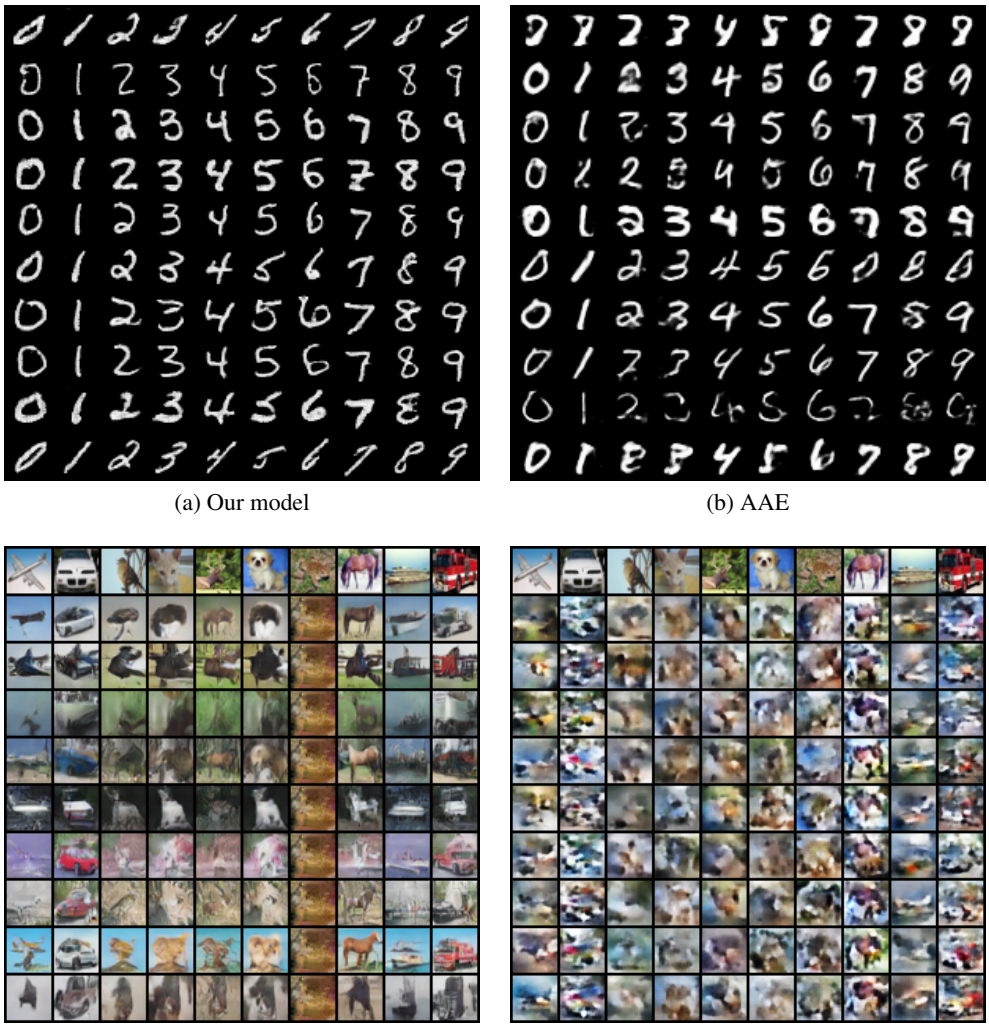
4.2.2 UNSUPERVISED LEARNING

This session presents experimental results of our model in learning unsupervisedly to disentangle the label information of images from the remaining information. As illustrated in Fig. 9, this is achieved by dividing the input to the code generator into two parts, one driven by an uniform categorial distribution and the other by a Gaussian. The categorical distribution encodes our prior belief about the data clusters. The number of distinct values over which it is defined specifies the presumed number of clusters in the data. The Gaussian serves to explain the data variability within each cluster. These two distributions are further mingled together by the fully connected layers in the code generator, to form a prior that is best suited for characterizing the data distribution. Again, the code generator (and thus the resulting prior) is learned from data using the GAN-based training procedure.

At test time, image generation is done similarly to the way for the supervised case. We start by sampling the categorical and Gaussian distributions, followed by feeding the samples into the code generator and then onwards to the decoder. In this experiment, the categorical distribution is defined over 10-D one-hot vectors, which denote the label variable, and the Gaussian is of 90-D. As in the supervised setting, after the model is trained, we alter the label variable or the Gaussian noise one at a time to verify whether the model has learned to cluster images. We expect that a good model should generate images of the same digit or of the same class when the Gaussian part is altered while fixing the label part.

The results in Fig. 10 confirm our model's disentanglement ability. Based on the same presentation order as in the supervised setting, we see that each column of images (which correspond to the same label variable) do show images of the same digit on MNIST and of similar class on CIFAR-10. The mode collapse phenomenon however becomes more apparent on CIFAR-10. By contrast, AAE does not work so well, when using the same autoencoder architecture[3] and the same prior dimension for a fair comparison. It is seen that on MNIST, some columns mix multiple digits and on CIFAR-10, AAE even experiences difficulty in reconstructing visually recognizable images.

[3]One major difference between our AAE implementation and that in Makhzani et al. (2015) is that we adopt convolutional neural networks rather than fully connected layers in the autoencoder.



(a) Our model (b) AAE

(c) Our model (d) AAE

Figure 8: Images generated by the proposed model (a)(c) and AAE (b)(d) trained on MNIST and CIFAR-10 datasets in the supervised setting. Each column of images have the same label/class information but varied Gaussian noise. On the other hand, each row of images have the same Gaussian noise but varied label/class variables. For CIFAR-10, we additionally put the ground truth in the first row for reference.

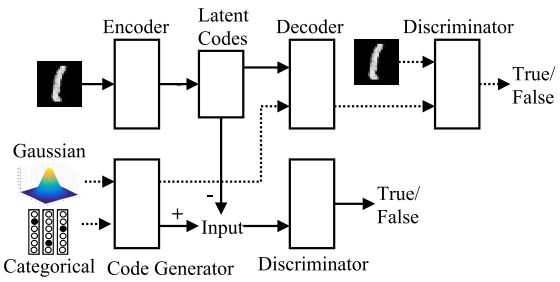
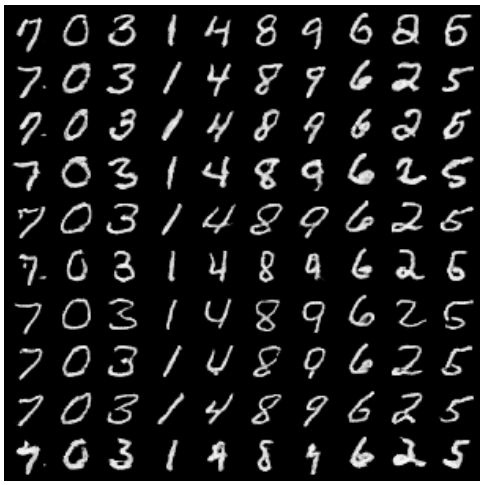
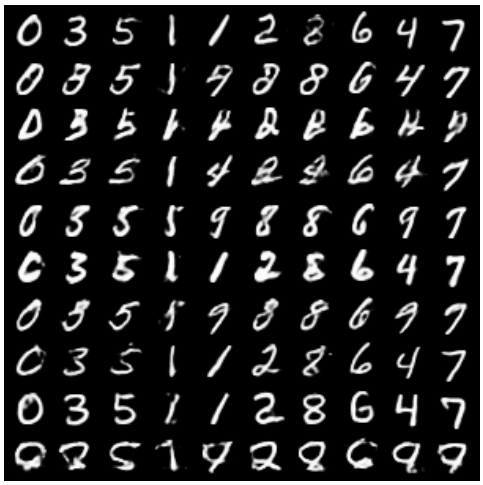


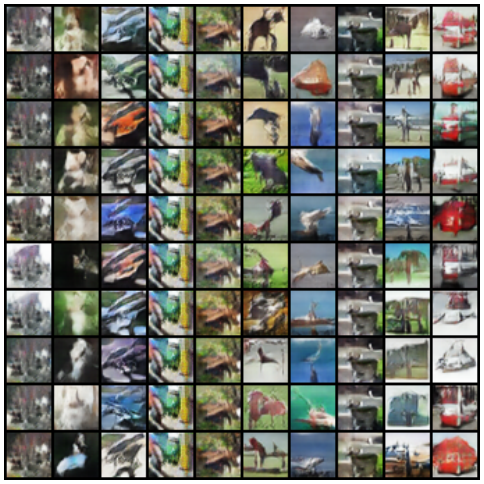
Figure 9: Unsupervised learning architecture with the code generator.



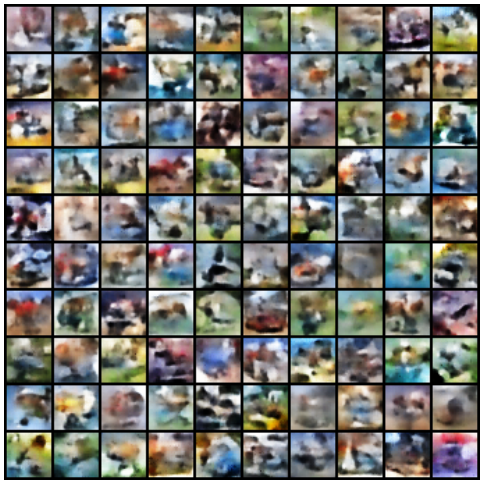
(a) Our model



(b) AAE



(c) Our model



(d) AAE

Figure 10: Images generated by the proposed model (a)(c) and AAE (b)(d) trained on MNIST and CIFAR-10 datasets in the unsupervised setting. Each column of images have the same label/class information but varied Gaussian noise. On the other hand, each row of images have the same Gaussian noise but varied label/class variables.

(a) This vibrant flower features lush red petals and a similar colored pistil and stamen

(b) This flower has white and crumpled petals with yellow stamen

Figure 11: Generated images from text descriptions.

Figure 12: Generated images in accordance with the varying color attribute in the text description "The flower is pink in color and has petals that are rounded in shape and ruffled." From left to right, the color attribute is set to pink, red, yellow, orange, purple, blue, white, green, and black, respectively. Note that there is no green or black flower in the dataset.

4.3 Text-to-Image Synthesis

This session presents an application of our model to text-to-image synthesis. We show that the code generator can transform the embedding of a sentence into a prior suitable for synthesizing images that match closely the sentence's semantics. To this end, we learn supervisedly the correspondence between images and their descriptive sentences using the architecture in Fig. 7, where given an image-sentence pair, the sentence's embedding (which is a 200-D vector) generated by a pre-trained recurrent neural network is input to the code generator and the discriminator in image space as if it were the label information, while the image representation is learned through the autoencoder and regularized by the output of the code generator. As before, a 100-D Gaussian is placed at the input of the code generator to explain the variability of images given the sentence.

The results in Fig. 11 present images generated by our model when trained on 102 Category Flower dataset (Nilsback & Zisserman, 2008). The generation process is much the same as that described in Section 4.2.1. It is seen that most images match reasonably the text descriptions. In Fig. 12, we further explore how the generated images change with the variation of the color attribute in the text description. We see that most images agree with the text descriptions to a large degree.

5 Conclusion

In this paper, we propose to learn a proper prior from data for AAE. Built on the foundation of AAE, we introduce a code generator to transform the manually selected simple prior into one that can better fit the data distribution. We develop a training process that allows to learn both the autoencoder and the code generator simultaneously. We demonstrate its superior performance over AAE in image generation and learning disentangled representations in supervised and unsupervised settings. We also show its ability to do cross-domain translation. Mode collapse and training instability are two major issues to be further investigated in future work.

References

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.

Prasoon Goyal, Zhiting Hu, Xiaodan Liang, Chenyu Wang, and Eric Xing. Nonparametric variational auto-encoders for hierarchical representation learning. *arXiv preprint arXiv:1703.07027*, 2017.

Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, 2016.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pp. 82–90, 2016.