SIP (2020), vol. 9, e4, page 1 of 11 © The Authors, 2020.

doi:10.1017/ATSIP.2019.25

ORIGINAL PAPER

Learning priors for adversarial autoencoders

HUI-PO WANG, WEN-HSIAO PENG AND WEI-JAN KO

Most deep latent factor models 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 its generative network has limited capacity. In this paper, we propose to learn a proper prior from data for adversarial autoencoders (AAEs). We introduce the notion of code generators to transform manually selected simple priors into ones that can better characterize the data distribution. Experimental results show that the proposed model can generate better image quality and learn better disentangled representations than AAEs in both supervised and unsupervised settings. Lastly, we present its ability to do cross-domain translation in a text-to-image synthesis task.

Keywords: Deep learning, Adversarial autoencoders, Learned priors, Latent factor models

Received 6 May 2019; Revised 19 November 2019

I. INTRODUCTION

Deep latent factor models, such as variational autoencoders (VAEs) and adversarial autoencoders (AAEs), are becoming increasingly popular in various tasks, such as image generation [1], unsupervised clustering [2,3], cross-domain translation [4], domain generalization [5], and unsupervised representation learning [6]. These models involve specifying a prior distribution over latent variables and defining a deep generative network (i.e. the decoder) that maps latent variables to data space in stochastic or deterministic fashion. Training such deep models usually requires learning a recognition network (i.e. the encoder) regularized by the prior.

Traditionally, a simple prior, such as the standard normal distribution [7], is used for tractability, simplicity, or not knowing what prior to use. It is hoped that this simple prior will be transformed somewhere in the deep generative network into a form suitable for characterizing the data distribution. While this might hold true when the generative network has enough capacity, applying the standard normal prior often results in over-regularized models with only few active latent dimensions [8].

Some recent works [9–11] suggest that the choice of the prior may have a profound impact on the expressiveness of the model. As an example, in learning the VAE with a simple encoder and decoder, Hoffman and Johnson [9] conjecture that multimodal priors can achieve a higher variational lower bound on the data log-likelihood than is possible with the standard normal prior. Tomczak and Welling [11] confirm the truth of this conjecture by showing that their multimodal prior, a mixture of the variational posteriors, consistently outperforms simple priors on a number of datasets in terms of maximizing the data log-likelihood. Taking one step further, Goyal *et al.* [10] learn a tree-structured non-parametric Bayesian prior for capturing the hierarchy of semantics presented in the data. All these priors are learned under the VAE framework following the principle of maximum likelihood.

Along a similar line of thinking, 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 generative network, can better characterize the data distribution. To this end, we generalize the framework of AAE in several significant ways:

- We replace the simple prior with a learned prior by training the code generator to output latent variables that will minimize an adversarial loss in data space.
- We employ a learned similarity metric [1] in place of the default squared error in data space for training the autoencoder.
- We maximize the mutual information between part of the code generator input and the decoder output for supervised and unsupervised training using a variational technique introduced in InfoGAN [12].

Extensive experiments confirm its effectiveness of generating better quality images and learning better disentangled representations than AAE in both supervised and unsupervised settings, particularly on complicated datasets. In addition, to the best of our knowledge, this is one of the first few works that attempt to introduce a learned prior for AAE.

National Chiao Tung University, 1001 Ta-Hsueh Rd., Hsinchu 30010, Taiwan

Corresponding authors:
Wen-Hsiao Peng.
E-mail: wpeng@cs.nctu.edu.tw

The remainder of this paper is organized as follows: Section 2 reviews the background and related works. Section 3 presents the implementation details and the training procedure of the proposed code generator. Section 4 presents extensive experiments to show the superiority of our models over prior works. Section 5 showcases an application of our model to text-to-image synthesis. Lastly, we conclude this paper with remarks on future work.

II. 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 $\log p(x;\theta)$.

Variational Autoencoders (VAEs). To improve the model's expressiveness, it is common to make deep the conventional latent factor model by introducing a neural network to $p(x \mid z;\theta)$. One celebrated example is VAE [7], which assumes the following prior $p(z)$ and $p(x \mid z;\theta)$:

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

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):

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

where $q(z \mid x;\phi)$ is the variational density, implemented by another neural network with parameter ϕ, to approximate the posterior $p(z \mid x;\theta)$. When regarding $q(z \mid x;\phi)$ as an (stochastic) encoder and $p(x \mid z;\theta)$ 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 (AAEs). Motivated by the observation that VAE is largely limited by the Gaussian prior assumption, i.e. $p(z) \sim \mathcal{N}(z; \mathrm{o}, I)$, Makhzani *et al.* [3] 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 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 posterior distribution[1] the same as the prior $p(z)$.

Non-parametric Variational Autoencoders (Non-parametric VAEs). 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.* [10] make an attempt to learn a non-parametric prior based on the nested Chinese restaurant process for VAEs. Learning is achieved by fitting it to the aggregated posterior distribution, which amounts to maximization of ELBO. The result induces a hierarchical structure of semantic concepts in latent space.

Variational Mixture of Posteriors (VampPrior). The VampPrior [11] is a new type of prior for the VAE. It consists of a mixture of the variational posteriors conditioned on a set of learned pseudo-inputs $\{x_k\}$. In symbol, this prior is given by

$$p(z) = \frac{1}{K} \sum_{k=1}^{K} q(z \mid x_k;\phi) \tag{3}$$

Its multimodal nature and coupling with the posterior achieve superiority over many other simple priors in terms of training complexity and expressiveness.

Inspired by these learned priors [10,11] for VAE, we propose in this paper the notion of code generators to learn a proper prior from data for AAE. The relations of our work with these prior arts are illustrated in Fig. 1.

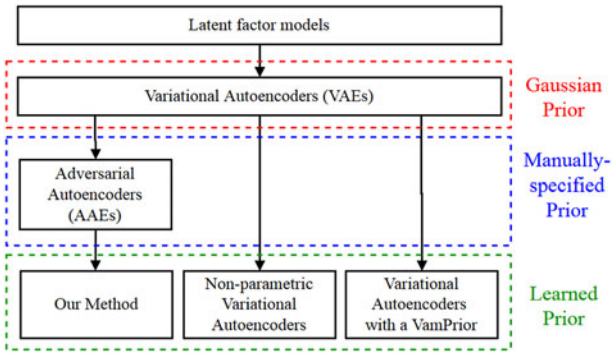

Fig. 1. The relations of our work with prior arts.

III. METHOD

In this paper, we propose to learn the prior from data instead of specifying it arbitrarily. Based on the framework 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.

A) Learning the prior

Because the 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 (see Fig. 2(b)), would lead to a prior 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 may make the likelihood

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

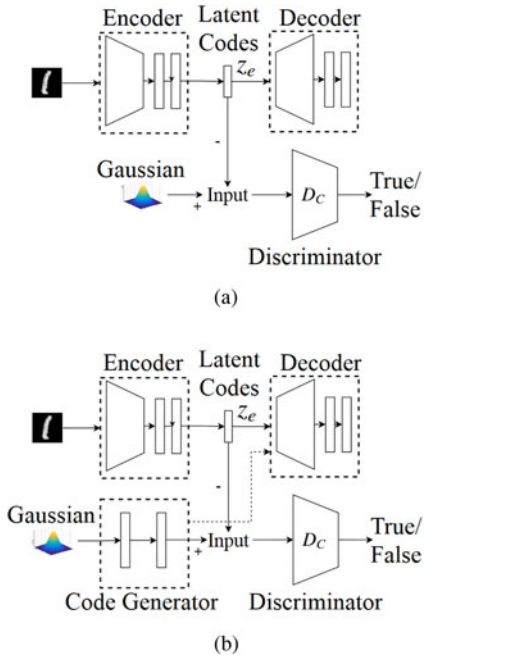

Fig. 2. The architecture of AAE without (a) and with (b) the code generator.

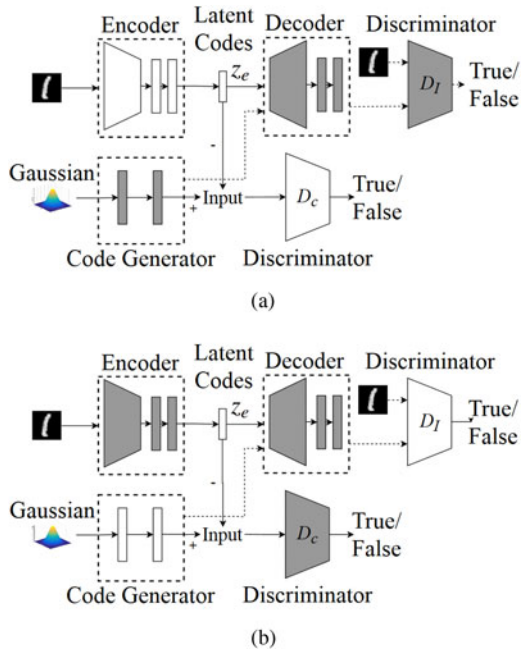

Fig. 3. Alternation of training phases: (a) the prior improvement phase and (b) the AAE phase. The shaded building blocks indicate the blocks to be updated.

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 distribution that minimizes the adversarial loss at the decoder output. Consider the example in Fig. 3(a). The decoder should generate images with a distribution that in principle matches the empirical distribution of real images in the training data, when driven by the output of the code generator. In symbols, this is to minimize

$$\mathcal{L}^I_{GAN} = \log(D_I(x)) + \log(1 - D_I(dec(z_c))), \qquad (4)$$

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 the training of the code generator and the decoder/encoder until convergence. In one phase, termed the *prior improvement phase* (see Fig. 3(a)), we update the code generator with the loss function in Eq. (4), by fixing the encoder[2]. In the other phase, termed the *AAE phase* (see Fig. 3(b)), we fix the code generator and update the autoencoder following the training procedure of AAE. Specifically, the encoder output has to be regularized by minimizing the following adversarial loss:

$$\mathcal{L}^C_{GAN} = \log(D_C(z_c)) + \log(1 - D_C(enc(x))), \qquad (5)$$

[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.

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.

Because the decoder will be updated in both phases, the convergence of the decoder relies on consistent training objectives of the two training phases. It is however noticed that the widely used pixel-wise squared error criterion in the AAE phase tends to produce blurry decoded images. This obviously conflicts with the adversarial objective in the prior improvement phase, which requires the decoder to produce sharp images. Inspired by the notion of learning similarity metrics [1] and perceptual loss [13], we change the criterion of minimizing squared error in pixel domain to be in feature domain. Specifically, in the AAE phase, we require that a reconstructed image $dec(enc(x))$ should minimize squared error in feature domain with respect to its original input x. This loss is referred to as *perceptual loss* and is defined by

$$\mathcal{L}_{rec} = \|\mathcal{F}(dec(enc(x))) - \mathcal{F}(x)\|^2, \qquad (6)$$

where $\mathcal{F}(\cdot)$ denotes the feature representation (usually the output of the last convolutional layer in the image discriminator D_I) of an image. With this, the decoder would be driven consistently in both phases toward producing decoded images that resemble closely real images.

B) Learning conditional priors

1) Supervised setting

The architecture in Fig. 3 can be extended to learn conditional priors supervisedly. Such priors find applications in conditional data generation, e.g. conditional image generation in which the decoder generates images according to

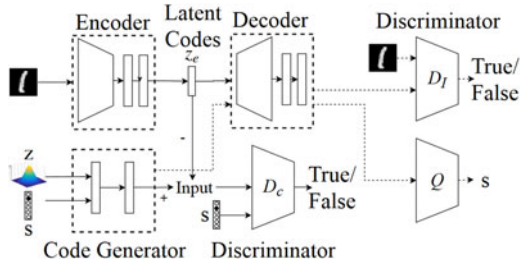

Fig. 4. Supervised learning architecture with the code generator.

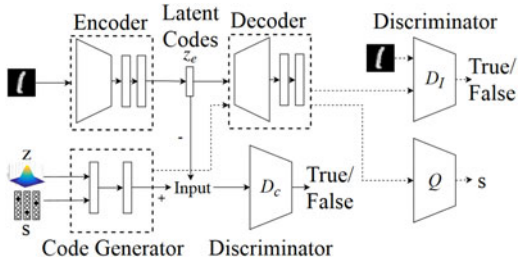

Fig. 5. Unsupervised learning architecture with the code generator.

their class labels s. To this end, we make three major changes to the initial architecture:

- Firstly, the code generator now takes as inputs a data label s and a noise variable z accounting for the intra-class variety, and produces a prior distribution conditional on the label s (see Fig. 4).
- Secondly, the end-to-end mutual information $I(s; dec(z_c))$ between the label s and the decoded image $dec(z_c)$ is maximized as part of our training objective to have both the code generator and the decoder pick up the information carried by the label variable s when generating the latent code z_c and subsequently the decoded image $dec(z_c)$. This is achieved by maximizing its variational lower bound $\mathcal{L}_I(s; dec(z_c))$ of $I(s; dec(z_c))$ [12] as given by

$$\mathcal{L}_I(s; dec(z_c)) = -E_{z,s\sim p(z,s),z_c\sim CG(z,s)}[\log Q(s \mid dec(z_c))], \tag{7}$$

  where $p(z, s) = p(z)p(s)$ is the joint distribution of the label s and the noise z, $CG(\cdot)$ is the code generator, and $Q(s \mid dec(z_c))$ is a classifier used to recover the label s of the decoded image $dec(z_c)$.
- Lastly, the discriminator D_c in latent code space is additionally provided with the label s as input, to implement class-dependent regularization at the encoder output during the AAE learning phase. That is,

$$\mathcal{L}^C_{GAN} = \log(D_C(z_c, s)) + \log(1 - D_C(enc(x), s)), \tag{8}$$

  where s is the label associated with the input image x.

The fact that the label s of an input image x needs to be properly fed to different parts of the network during training indicates the supervised learning nature of the aforementioned procedure.

2) Unsupervised setting

Taking one step further, we present in Fig. 5 a re-purposed architecture to learn conditional priors under an unsupervised setting. Unlike the supervised setting where the correspondence between the label s and the image x is explicit during training, the unsupervised setting is to learn the correspondence in an implicit manner. Two slight changes are thus made to the architecture in Fig. 4: (1) the label s at the input of the code generator is replaced with a label drawn randomly from a categorical distribution; and (2) the discriminator D_c in the latent code space is made class agnostic by removing the label input. The former is meant to produce a multimodal distribution in the latent space while the latter is to align such a distribution with that at the encoder output. Remarkably, which mode (or class) of distribution an image x would be assigned to in latent code space is learned implicitly. In a sense, we hope the code generator can learn to discover the intriguing latent code structure inherent at the encoder output. It is worth pointing out that in the absence of any regularization or guidance, there is no guarantee that this learned assignment would be in line with the semantics attached artificially to each data sample.

Algorithm 1 details the training procedure.

IV. Experiments

We first show the superiority of our learned priors over manually-specified priors, followed by an ablation study of individual components. In the end, we compare the performance of our model with AAE in image generation tasks. Unless stated otherwise, all the models adopt the same autoencoder for a fair comparison.

A) Comparison with prior works

Latent factor models with their priors learned from data rather than specified manually should better characterize the data distribution. To validate this, we compare the performance of our model with several prior arts, including AAE [3], VAE [7], and Vamprior [11], in terms of Inception Score (IS). Of these works, AAE chooses a Gaussian prior and regularizes the latent code distribution with an adversarial loss [14]. VAE [7] likewise adopts a Gaussian prior yet uses the KL-divergence for regularization. Vamprior [11] learns for VAE a Gaussian mixture prior. For the results of Vamprior [11], we run their released software [11] but replace their autoencoder with ours for a fair comparison.

Table 1 compares their Inception Score for image generation on CIFAR-10 with a latent code size of 64. As expected, both AAE [3] and VAE [7], which adopt manually-specified priors, have a lower IS of 2.15 and 3.00, respectively. Somewhat surprisingly, Vamprior [11], although using a learned prior, does not have an advantage over VAE [7] with a simple Gaussian prior in the present case. This may be attributed to the fact that the prior is limited to be a Gaussian mixture distribution. Relaxing this constraint by modeling the prior

Algorithm 1 Training procedure.

Initialize $\theta_{enc}, \theta_{dec}, \theta_{CG}, \theta_{D_I}, \theta_{D_C}, \theta_Q$
Repeat (for each epoch E_i)
  Repeat (for each mini-batch x_j)
    // AAE phase
    If label s exists then
      $z, s \sim p(z, s) = p(z)p(s)$
      $z_c \leftarrow CG(z, s)$
    Else
      $z \sim p(z)$
      $z_c \leftarrow CG(z)$
    End If

    Compute $\mathcal{L}^C_{GAN}$, $\mathcal{L}_{rec}$

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

    // Prior improvement phase
    If label s exists then
      $z, s \sim p(z, s) = p(z)p(s)$
      $z_c \leftarrow CG(z, s)$
      Compute $\mathcal{L}^I_{GAN}$ and $\mathcal{L}_I(s; dec(z_c))$
    Else
      $z \sim p(z)$
      $z_c \leftarrow CG(z)$
      Compute $\mathcal{L}^I_{GAN}$
    End If

    // Update network parameters
    $\theta_{D_I} \leftarrow \theta_{D_I} - \nabla_{\theta_{D_I}}(-\mathcal{L}^I_{GAN})$
    If label s exists then
      $\theta_{dec} \leftarrow \theta_{dec} - \nabla_{\theta_{dec}}(\mathcal{L}^I_{GAN} + \mathcal{L}_I(s; dec(z_c))$
      $\theta_{CG} \leftarrow \theta_{CG} - \nabla_{\theta_{CG}}(\mathcal{L}^I_{GAN} + \mathcal{L}_I(s; dec(z_c))$
      $\theta_Q \leftarrow \theta_Q - \nabla_{\theta_Q}(\mathcal{L}_I(s; dec(z_c)))$
    Else
      $\theta_{dec} \leftarrow \theta_{dec} - \nabla_{\theta_{dec}}(\mathcal{L}^I_{GAN})$
      $\theta_{CG} \leftarrow \theta_{CG} - \nabla_{\theta_{CG}}(\mathcal{L}^I_{GAN})$
    End If
  Until all mini-batches are processed
Until termination

Table 1. Comparison with AAE, VAE, and Vamprior on CIFAR-10.

Method	Inception Score
AAE [3] w/ a Gaussian prior	2.15
VAE [7] w/ a Gaussian prior	3.00
Vamprior [11]	2.88
Our method w/ a learned prior	6.52

with a neural network, our model achieves the highest IS of 6.52.

Figure 6 further visualizes sample images generated with these models by driving the decoder with latent codes drawn from the prior or the code generator in our case. It is observed that our model produces much sharper images than the others. This confirms that a learned and flexible prior is beneficial to the characterization and generation of data.

To get a sense of how our model performs as compared to other state-of-the-art generative models, Table 2 compares their Inception Score on CIFAR-10. Caution must be exercised in interpreting these numbers as these models adopt different decoders (or generative networks). With the current implementation, our model achieves a comparable score to other generative models. Few sample images of these models are provided in Fig. 7 for subjective evaluation.

B) Ablation study

In this section, we conduct an ablation study to understand the effect of (A) the learned prior, (B) the perceptual loss, and (C) the updating of the decoder in both phases on Inception Score [19]. To this end, we train an AAE [3] model with a 64-D Gaussian prior on CIFAR-10 as the baseline. We then enable incrementally each of these design choices. For a fair comparison, all the models are equipped with an identical autoencoder architecture yet trained with their respective objectives.

From Table 3, we see that the baseline has the lowest IS and that replacing the manually-specified prior with our learned prior increases IS by about 0.9. Furthermore, minimizing the perceptual loss instead of the conventional mean squared error in training the autoencoder achieves an even higher IS of 3.69. This suggests that the perceptual loss does help make more consistent the training objectives for the decoder in the AAE and the prior improvement phases. Under such circumstances, allowing the decoder to be updated in both phases tops the IS.

C) In-depth comparison with AAE

Since our model is inspired by AAE [3], this section provides an in-depth comparison with it in terms of image generation. In this experiment, the autoencoder in our model is trained based on minimizing the perceptual loss (i.e. the mean squared error in feature domain), whereas by convention, AAE [3] is trained by minimizing the mean squared error in data domain.

Figure 8 displays side-by-side images generated from these models when trained on MNIST and CIFAR-10 datasets. They are produced by the decoder driven by samples from their respective priors. In this experiment, two observations are immediate. First, our model can generate sharper images than AAE [3] on both datasets. Second, AAE [3] experiences problems in reconstructing visually-plausible images on the more complicated CIFAR-10. These highlight the advantages of optimizing the autoencoder with the perceptual loss and learning the code generator through an adversarial loss, which in general produces subjectively sharper images.

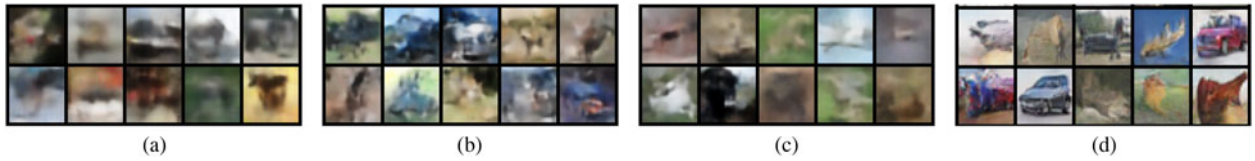

Fig. 6. Sample images produced by (a) AAE, (b) VAE, (c) Vamprior, and (d) the proposed model. (a) AAE [3], (b) VAE [7], (c) Vamprior [11], and (d) Proposed model.

Table 2. Comparison with other state-of-the-art generative models on CIFAR-10.

Method	Inception Score
BEGAN [15]	5.62
DCGAN [16]	6.16
LSGAN [17]	5.98
WGAN-GP [18]	7.86
Our method w/ a learned prior	6.52

Table 3. Inception score of generated images with the models trained on CIFAR-10: A, B, and C denote respectively the design choices of enabling the learned prior, the perceptual loss, and the updating of the decoder in both phases.

Method	A	B	C	IS
AAE [3] w/ a Gaussian prior and MSE loss				2.15
AAE w/ a learned prior and MSE loss	✓			3.04
AAE w/ a learned prior and perceptual loss	✓	✓		3.69
Ours (full)	✓	✓	✓	6.52

Another advantage of our model is its ability to have better adaptability in higher dimensional latent code space. Figure 9 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 Fig. 8, it is seen that the increase in code dimension has little impact on our model, but exerts a strong influence on AAE [3]. In the present case, AAE [3] 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 [3]. This emphasizes the importance of having a prior that can adapt automatically to a change in the dimensionality of the code space and data.

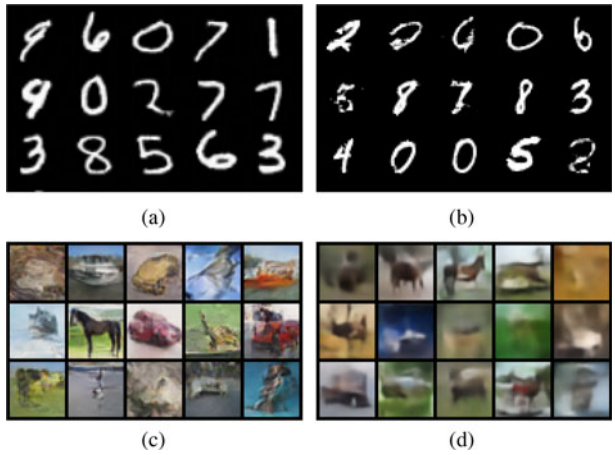

Fig. 8. Images generated by our model and AAE trained on MNIST (upper) and CIFAR-10 (lower). (a) Our model + 8-D latent code, (b) AAE [3] + 8-D latent code, (c) Our model + 64-D latent code, and (d) AAE [3] + 64-D latent code.

D) Disentangled representations

Learning disentangled representations is desirable in many applications. It refers generally to learning a data 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 data-driven priors, we repeat the disentanglement tasks in [3], and compare its performance with AAE [3].

1) Supervised learning

This section presents experimental results of using the network architecture in Fig. 4 to learn supervisedly a code generator CG that outputs a conditional prior given the image label s for characterizing the image distribution. In particular, the remaining uncertainty about the image's appearance given its label is modeled by transforming a Gaussian noise z through the code generator CG. By having the noise z be independent of the label s, we arrive at a disentangled representation of images. At test time, the generation of an image x for a particular class is achieved by inputting the class label s and a Gaussian noise z to the code generator $CG(\cdot)$ and then passing the resulting code z_c through the decoder $x = dec(z_c)$.

To see the sole contribution from the learned prior, the training of the AAE baseline [3] also adopts the perceptual loss and the mutual information maximization; that is, the

Fig. 7. Subjective quality evaluation of generated images produced by state-of-the-art generative models. (a) BEGAN [15], (b) DCGAN [16], (c) LSGAN [17], (d) WGAN-GP [18], and (e) Proposed model.

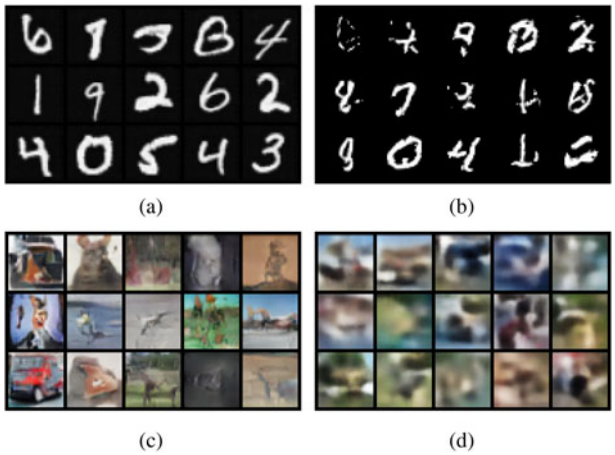

Fig. 9. Images generated by our model and AAE trained on MNIST (upper) and CIFAR-10 (lower). In this experiment, the latent code dimension is increased significantly to 64-D and 2000-D for MNIST and CIFAR-10, respectively. For AAE, the re-parameterization trick is applied to the output of the encoder as suggested in [3]. (a) Our model + 100-D latent code, (b) AAE [3] + 100-D latent code, (c) Our model + 2000-D latent code, and (d) AAE [3] + 2000-D latent code.

only difference to our model is the direct use of the label s and the Gaussian noise z as the conditional prior.

Figure 10 displays images generated by our model and AAE [3]. Both models adopt a 10-D one-hot vector to specify the label s and a 54-D Gaussian to generate the noise z. To be fair, the output of our code generator has an identical dimension (i.e. 64) to the latent prior of AAE [3]. Each row of Fig. 10 corresponds to images generated by varying the label s while fixing the noise z. Likewise, each column shows images that share the same label s yet with varied noise z.

On MNIST and SVHN, 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 main digit changes with the label s regardless of the noise z, and that along each column, the style varies without changing the main digit. On CIFAR-10, the two models behave differently. While both produce visually plausible images, ours generate more semantically discernible images that match their labels.

Figure 11 visualizes the output of the code generator with the t-distributed stochastic neighbor embedding (t-SNE). It is seen that the code generator learns a distinct conditional distribution for each class of images. It is believed that the more apparent inter-class distinction reflects the more difficult it is for the decoder to generate images of different classes. Moreover, the elliptic shape of the intra-class distributions in CIFAR-10 may be ascribed to the higher intra-class variability.

2) Unsupervised learning

This section presents experimental results of using the network architecture in Fig. 5 to learn unsupervisedly a code generator *CG* that outputs a prior for best characterizing the image distribution. In the present case, the label s is drawn randomly from a categorical distribution and independently from the Gaussian input z, as shown in Fig. 5. The categorical distribution encodes our prior belief about data clusters, with the number of distinct values over which it is defined specifying the presumed number of clusters

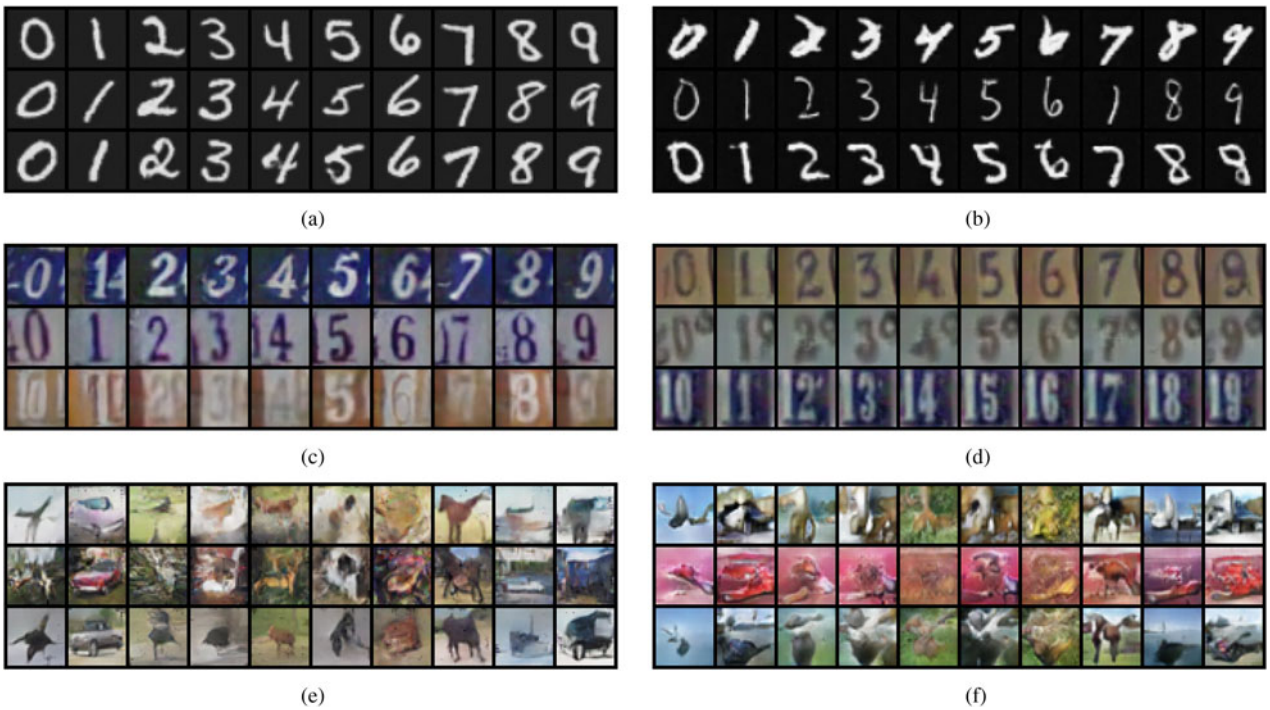

Fig. 10. Images generated by the proposed model (a)(c)(e) and AAE (b)(d)(f) trained on MNIST, SVHN, and CIFAR-10 datasets in the supervised setting. Each column of images has the same label/class information but varied Gaussian noise. On the other hand, each row of images has the same Gaussian noise but varied label/class variables. (a) Our model, (b) AAE [3], (c) Our model, (d) AAE [3], (e) Our model, and (f) AAE [3].

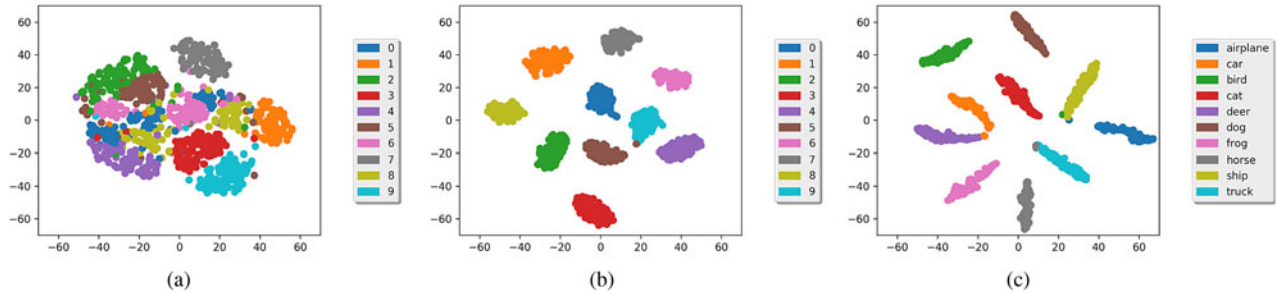

Fig. 11. Visualization of the code generator output in the supervised setting. (a) MNIST, (b) SVHN, and (c) CIFAR-10.

in the data. The Gaussian serves to explain the data variability within each cluster. In regularizing the distribution at the encoder output, we want the code generator *CG* to make sense of the two degrees of freedom for a disentangled representation of images.

At test time, image generation is done similarly to the supervised case. We start by sampling s and z, followed by feeding them into the code generator and then onwards to the decoder. In this experiment, the categorical distribution is defined over 10-D one-hot vectors and the Gaussian is 90-D. As in the supervised setting, after the model is trained, we alter the label variable s or the Gaussian noise z one at a time to verify whether the model has learned to cluster images unsupervisedly. We expect that a good model should generate images with certain common properties (e.g. similar backgrounds or digit types) when the Gaussian part z is altered while the label part s remains fixed.

The results in Fig. 12 show that on MNIST, both our model and AAE successfully learn to disentangle the digit type from the remaining information. Based on the same presentation order as in the supervised setting, we see that each column of images (which correspond to the same label s) does show images of the same digit. On the more complicated SVHN and CIFAR-10 datasets, each column mixes images from different digits/classes. It is however worth noting that both models have a tendency to cluster images with similar backgrounds according to the label variable s. Recall that without any semantic guidance, there is no guarantee that the clustering would be in line with the semantics attached artificially to each data sample.

Figure 13 further visualizes the latent code distributions at the output of the code generator and the encoder. Several observations can be made. First, the encoder is regularized well to produce an aggregated posterior distribution similar to that at the code generator output. Second, the code

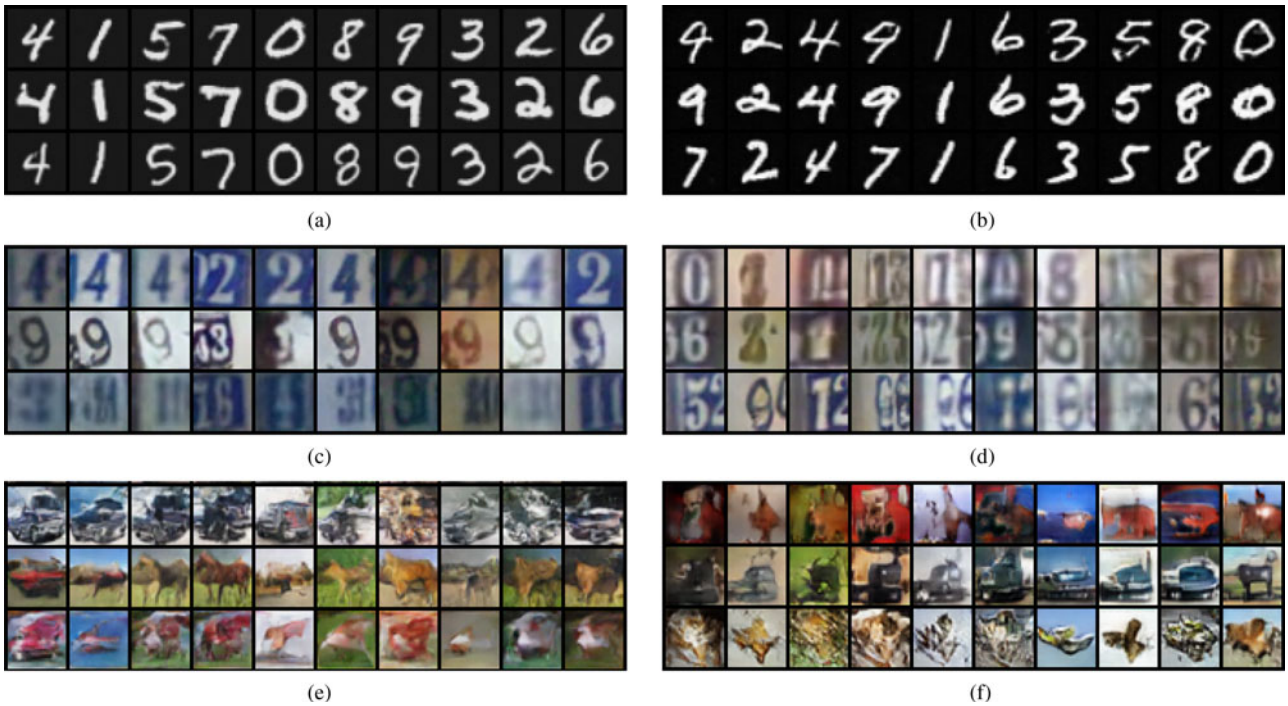

Fig. 12. Images generated by the proposed model (a)(c)(e) and AAE (b)(d)(f) trained on MNIST, SVHN, and CIFAR-10 datasets in the unsupervised setting. Each column of images has the same label/class information but varied Gaussian noise. On the other hand, each row of images has the same Gaussian noise but varied label/class variables. (a) Our model, (b) AAE, (c) Our model, (d) AAE, (e) Our model, and (f) AAE.

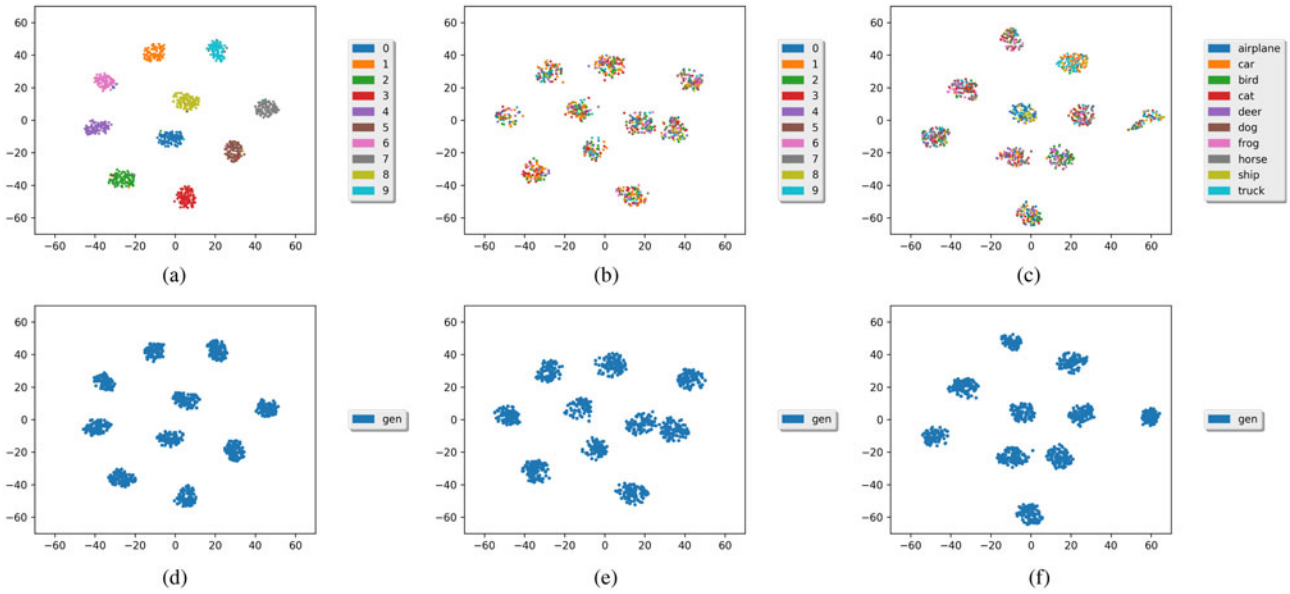

Fig. 13. Visualization of the encoder output versus the code generator output in the unsupervised setting. (a) Encoder (MNIST), (b) Encoder (SVHN), (c) Encoder (CIFAR-10), (d) Code generator (MNIST), (e) Code generator (SVHN), and (f) Code generator (CIFAR-10).

generator learns distinct conditional distributions according to the categorical label input s. Third, quite by accident, the encoder learns to cluster images of the same digit on MNIST, as has been confirmed in Fig. 12. As expected, such semantic clustering in code space is not obvious on more complicated SVHN and CIFAR-10, as is evident from the somewhat random assignment of latent codes to images of the same class or label.

V. APPLICATION: TEXT-TO-IMAGE SYNTHESIS

This section 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. 4, 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. 14 present images generated by our model when trained on 102 Category Flower dataset [20]. The generation process is much the same as that described in Section 1. It is seen that most images match reasonably

Fig. 14. Generated images from text descriptions. (a) This vibrant flower features lush red petals and a similar colored pistil and stamen and (b) This flower has white and crumpled petals with yellow stamen.

Fig. 15. 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.

the text descriptions. In Fig. 15, 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.

VI. 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

1 Larsen A.B.L.; Snderby S.K.; Larochelle H.; Winther O.: Autoencoding beyond pixels using a learned similarity metric, in *Proc. Int. Conf. Machine Learning (ICML)*, 2016, 1558–1566.

2 Dilokthanakul N.; Mediano P.A.; Garnelo M.; Lee M.C.; Salimbeni H.; Arulkumaran K.; Shanahan M.: Deep unsupervised clustering with gaussian mixture variational autoencoders, arXiv preprint arXiv:1611.02648, 2016.

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

4 Wu J.; Zhang C.; Xue T.; Freeman B.; Tenenbaum J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling, in *Advances in Neural Information Processing Systems (NIPS)*, 2016, 82–90.

5 Li H.; Jialin Pan S.; Wang S.; Kot A.C.: Domain generalization with adversarial feature learning, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, 5400–5409.

6 Zhang Y.; Fu Y.; Wang P.; Li X.; Zheng Y.: Unifying inter-region autocorrelation and intra-region structures for spatial embedding via collective adversarial learning, in *Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, 2019, 1700–1708.

7 Kingma D.P.; Welling M.: Auto-encoding variational bayes, in *Proc. Int. Conf. Learning Representations (ICLR)*, 2013.

8 Burda Y.; Grosse R.; Salakhutdinov R.: Importance weighted autoencoders, arXiv preprint arXiv:1509.00519, 2015.

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

10 Goyal P.; Hu Z.; Liang X.; Wang C.; Xing E.P.: Nonparametric variational auto-encoders for hierarchical representation learning, in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2017, 5094–5102.

11 Tomczak J.; Welling M.: Vae with a vampprior, in *Proc. Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2018, 1214–1223.

12 Chen X.; Duan Y.; Houthooft R.; Schulman J.; Sutskever I.; Abbeel P.: Infogan: interpretable representation learning by information maximizing generative adversarial nets, in *Advances in Neural Information Processing Systems (NIPS)*, 2016, 2172–2180.

13 Johnson J.; Alahi A.; Fei-Fei L.: Perceptual losses for real-time style transfer and super-resolution, in *Proc. Eur. Conf. Computer Vision (ECCV)*, 2016, 694–711.

14 Goodfellow I.; Pouget-Abadie J.; Mirza M.; Xu B.; Warde-Farley D.; Ozair S.; Courville A.; Bengio Y.: Generative adversarial nets, in *Advances in Neural Information Processing Systems (NIPS)*, 2014, 2672–2680.

15 Berthelot D.; Schumm T.; Metz L.: Began: Boundary equilibrium generative adversarial networks, arXiv preprint arXiv:1703.10717, 2017.

16 Radford A.; Metz L.; Chintala S.: Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv:1511.06434, 2015.

17 Mao X.; Li Q.; Xie H.; Lau R.Y.; Wang Z.; Paul Smolley S.: Least squares generative adversarial networks, in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2017, 2794–2802.

18 Gulrajani I.; Ahmed F.; Arjovsky M.; Dumoulin V.; Courville A.C.: Improved training of wasserstein gans, in *Advances in Neural Information Processing Systems (NIPS)*, 2017, 5767–5777.

19 Salimans T.; Goodfellow I.; Zaremba W.; Cheung V.; Radford A.; Chen X.: Improved techniques for training gans, in *Advances in Neural Information Processing Systems (NIPS)*, 2016, 2234–2242.

20 Nilsback M.-E.; Zisserman A.: Automated flower classification over a large number of classes, in *Proc. Indian Conf. Computer Vision, Graphics and Image Processing*, 2008.

APPENDIX

IMPLEMENTATION DETAILS

Tables A1–A3 present the implementation details of each component in our model. Each cell in the tables presents the type of neural networks, the output size, usage of batch normalization,

Table A.1. Implementation details of the encoder and decoder networks.

Encoder	Decoder
Input 32 × 32 images	Input latent code ∈ $R^{code\ size}$
3 × 3 conv. 64 RELU stride 2 pad 1	4 × 4 upconv. 512 BN. RELU stride 1
3 × 3 residual blcok 64	4 × 4 up sampling residual block 256 stride 2
3 × 3 down sampling residual blcok 128 stride 2	4 × 4 up sampling residual block 128 stride 2
3 × 3 down sampling residual blcok 256 stride 2	4 × 4 up sampling residual block 64 stride 2
3 × 3 down sampling residual block 512 stride 2	3 × 3 conv. image channels Tanh
4 × 4 avg. pooling stride 1	
FC. 2 × code size BN. RELU	
FC. code size Linear	

Table A.2. Implementation details of the code generator networks.

Code Generator	Residual block
Input noise ∈ $R^{noise\ size}$	Input feature map
FC. 2 × noise size BN. RELU	3 × 3 conv. out_channels RELU stride 2 pad 1
FC. latent code size BN. Linear	3 × 3 conv. out_channels RELU stride 1 pad 1
	skip connection output = input + residual
	RELU

Table A.3. Implementation details of the image and code discriminator.

Image Discriminator D/Q	Code Discriminator
Input 32×32 images	Input latent code
4×4 conv. 64 LRELU stride 2 pad 1	FC 1000 LRELU
4×4 conv. 128 BN LRELU stride 2 pad 1	FC 500 LRELU
4×4 conv. 256 BN LRELU stride 2 pad 1	FC 200 LRELU
FC. 1000 LRELU	FC 1 Sigmoid
FC 1 Sigmoid for D	
FC 10 Softmax for Q	

the type of activation function, the size for strides, and the size of padding.

Hui-Po Wang received his B.S. degree in computer science from the National Tsing Hua University in 2017. He is currently pursuing his M.S. degree in the National Chiao Tung University. His research interest lies in deep generative models, unsupervised domain adaptation, and related vision applications.

Wen-Hsiao Peng received his Ph.D. degree from the National Chiao Tung University (NCTU), Taiwan in 2005. He was with the Intel Microprocessor Research Laboratory, USA from 2000 to 2001. Since 2003, he has actively participated in the ISO/IEC and ITU-T video coding standardization process. He is a Professor with the Computer Science Department, NCTU, and was a Visiting Scholar with the IBM Thomas J. Watson Research Center, USA, from 2015 to 2016. His research interests include video/image coding and multimedia analytics. Dr. Peng is an IEEE CASS VSPCTC member. He was Publication Chair for 2019 IEEE ICIP, Technical Program Co-chair for 2011 IEEE VCIP, 2017 IEEE ISPACS and 2018 APSIPA ASC, Area Chair for IEEE ICME and VCIP, Track Chair for IEEE ISCAS. He served asa Guest Editor/SEB Member for IEEE JETCAS and an Associate Editor for IEEE TCSVT. He was elected the Chair-elect of IEEE CASS VSPCTC anda Distinguished Lecturer of APSIPA.

Wei-Jan Ko received the B.S. degree in traffic science from the Central Police University and the M.S. degree in computer science from the National Chiao Tung University (NCTU), Taiwan, in 2014 and 2018, respectively. He is currently a doctoral student in computer science at NCTU. He has been a teaching assistant in the Introduction of Artificial intelligence course in NCTU, and an instructor of Artificial intelligence and Deep Learning in NCTU MOOC(Massive open online course) platform. His current research interests include Deep Generative Model and Intelligent Systems with smart city.