Learning a Generative Adversarial Network for High Resolution Artwork Synthesis

Wei Ren Tan, *Student Member, IEEE,* Chee Seng Chan, *Senior Member, IEEE,*
Hernán E. Aguirre, *Member, IEEE,* and Kiyoshi Tanaka, *Member, IEEE*

arXiv:1708.09533v1 [cs.CV] 31 Aug 2017

***Abstract*—Artwork is a mode of creative expression and this paper is particularly interested in investigating if machine can learn and synthetically create artwork that are usually non-figurative and structured abstract. To this end, we propose an extension to the Generative Adversarial Network (GAN), namely as the ArtGAN to synthetically generate high quality artwork. This is in contrast to most of the current solutions that focused on generating structural images such as birds, flowers and faces. The key innovation of our work is to allow back-propagation of the loss function w.r.t. the labels (randomly assigned to each generated images) to the generator from the categorical autoencoder-based discriminator that incorporates an autoencoder into the categorical discriminator for additional complementary information. In order to synthesize a high resolution artwork, we include a novel magnified learning strategy to improve the correlations between neighbouring pixels. Based on visual inspection and Inception scores, we demonstrate that ArtGAN is able to draw high resolution and realistic artwork, as well as generate images of much higher quality in four other datasets (i.e. CIFAR-10, STL-10, Oxford-102 and CUB-200).**

***Index Terms*—Generative Adversarial Networks, Image Synthesis, Deep Learning**

I. Introduction

Recently, Goodfellow et al. [1] introduced an interesting features learning method via adversarial training a deep generative model, called the Generative Adversarial Networks (GAN). Unlike the traditional deep discriminative models [2]–[4], the representations learnt by GAN can be visualized through the trained generator (of the GAN) in the form of synthetic images. Since then, many extensions of GAN [5]–[11] have been introduced and shown significant promise in synthetically generating structural images using the MNIST [12], CIFAR-10 [13] and ImageNet [14] datasets.

In this paper, we would like to investigate if machine can generate (more challenging) images such as fine art paintings using the GAN model. This is because artwork is a mode of creative expression, coming in different kinds of forms, including drawing, naturalistic, abstraction, etc. For instance, Color Field paintings are characterized primarily by large fields of flat, solid color spread across the canvas, making them non-figurative. While Cubism may contain object, it is broken up and reassembled in a structured abstracted form. Therefore, it is very hard to understand the context in the artwork. That is to say, we are interested to understand the capability of GAN in learning visual representations when given an unstructured data such as the artwork.

To this end, we propose a novel Generative Adversarial Network, namely as the **ArtGAN**[1]. We anticipate that a good way to look at this problem is to understand how human learns to draw. An artist teacher wrote an online article[2] and pointed out that an effective learning requires to focus on a particular type of skills at a time, e.g. practice to draw a particular object or one kind of movement. Accordingly, ArtGAN takes a randomly chosen label information and a noise vector as inputs. The chosen label is used as the true label when computing the loss function for the generated image. The idea is to allow the generator to learn better by leveraging the feedback information from the labels. Inspired by recent works [15], [16], a categorical autoencoder-based discriminator that incorporates an autoencoder into the categorical discriminator for additional complementary information is also introduced. Rather than deploying two separate computationally expensive networks (a categorical discriminator and an autoencoder), our categorical autoencoder-based discriminator partly shares the same architecture and weights. In specific, the *encoder* in the autoencoder is shared by the categorical discriminator. The overall architecture of the ArtGAN is illustrated in Figure 1.

Additionally, we proposed a novel magnified learning strategy so that ArtGAN can synthetically generate high-resolution artwork. The motivation behind this approach is to generate a set of pixels that vote for a single pixel. One may naively train an ensemble of GANs to achieve this goal. However, training multiple networks explicitly is computationally expensive and unnecessary to achieve similar performance gain [17], [18]. Hence, we propose an alternative approach by generating

W.R. Tan, A. Hernan and K. Tanaka are with Shinshu University, Nagano, Japan. e-mail: {14st203c,ahernan,ktanaka}@shinshu-u.ac.jp

C.S. Chan is with Center of Image and Signal Processing, Fauclty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia. e-mail: cschan@um.edu.my

[1]We name the network as ArtGAN since the nature of this work is to synthetically generate artwork.

[2]http://www.learning-to-see.co.uk/effective-practice

Fig. 1: Overview of the ArtGAN architecture. $\mathbf{z}$ and $\hat{c}$ are concatenated and fed to the generator to produce synthetic image $G(\mathbf{z}, \hat{c})$. Either the downsampled generated image $G(\mathbf{z}, \hat{c})$ or real data $\hat{\mathbf{x}}$ is used as the input $\mathbf{x}$ to the (categorical autoencoder-based) discriminator. The discriminator produces three outputs: the class prediction $p(\mathbf{c}|\mathbf{x})$, adversarial prediction $p(y|\mathbf{x})$, and the reconstructed image $D_{AE}(\mathbf{x})$.

images with higher resolution compared to the available image size in the dataset, e.g. 64×64 pixels instead of 32×32 pixels for CIFAR-10 samples. Then, the generated images are downsampled to the original size (i.e. 32×32 as accordance with previous example), which resembles a form of voting scheme. An advantage of this approach is that the correlations between the pixels within the same downsampling block can be learnt better.

In summary, our key contributions are 1) we propose a novel conditional GAN variant, namely as the ArtGAN to emulate the concept of effective learning to generate very challenging images (i.e. artwork). To the best of our knowledge, no existing empirical research has addressed the implementation of a generative model on a large scale paintings dataset. 2) we propose a novel magnified learning to synthesize better quality images. 3) Empirically, we show that our model is capable of generating high quality artwork that exhibit similar visual representations within *genre*, *artist*, or *style*. 4) Our model is also able to generate high resolution images on CIFAR-10 [13], STL-10 [19], Oxford-102 [20], and CUB-200 [21] that look natural and contain clear object structures in them.

A preliminary version of this work was presented earlier [22]. The present work adds to the initial version in significant ways. Firstly, we improve the ArtGAN by introducing categorical autoencoder-based discriminator. Secondly, we extend the ArtGAN to synthetically generate high resolution images using a novel magnified learning strategy. Thirdly, considerable new analyses and intuitive explanations are added to the initial results. For instance, we extend the original qualitative experiments from Wikiart [23] to CIFAR-10 [13], STL-10 [19], Oxford-102 [20], and CUB-200 [21] datasets. In addition, we included the Inception score [24] as a quantitative metric and ArtGAN obtains state-of-the-art result on CIFAR-10 dataset.

The rest of the paper is structured as follows. Related works are discussed in the next section (Section II). Section III describes the proposed method and magnified learning is detailed in Section IV. Experiments are discussed in Section V. Last but not least, conclusion is drawn in Section VI.

II. Related works

Generative models have been a fundamental interest and challenging problem in the field of computer vision and machine learning. In contrast with discriminative models which only allow sampling of the target variables conditioned on the observed quantities, generative models can be used to simulate observed distribution, and so they offer a much richer representation. Early works [25]–[27] studied the statistical properties of natural images, but are limited to texture or certain patterns (e.g.faces) only due to the difficulty in learning an effective feature representation. Recently, advances in deep models nourish a series of deep generative models [28], [29] for image generation through the Bayesian inference, typically trained by maximizing the log-likelihood. These models are able to construct decent quality images on less complicated images, such as digits and faces, but generally have intractable likelihood and require numerous approximations. The denoising autoencoders (DAE) [30] was introduced to overcome the intractable problem, but the reconstructed images are generally blur. DRAW [31] was proposed, depicted as a sequential model with attention mechanism to draw image recursively. It mimics the process of human drawing but faces challenges when it is scaling up to large complex images. PixelRNN [32] is another autoregressive approach for image generation which has received much attentions recently. Its extensions (PixelCNN [33] and PixelCNN++ [34]) are able to synthesize decent images but are computationally expensive to train[3].

A more significant breakthrough framework, Generative Adversarial Network (GAN) was introduced by Goodfellow et al. [1]. This framework bypasses the difficulty of maximum likelihood estimation by estimating the generative model via an adversarial process and has gained striking successes in natural image generation. However, GAN is well-known with its instability during training. To tackle this problem, feature matching [24] was proposed to generate descent quality images. While, instance noise [35] is also an robust trick

[3]They reported that PixelCNN++ requires approximately 5 days to converge to the reported results using 8 Maxwell TITAN X GPUs in github: https://github.com/openai/pixel-cnn.

to remedy the instability of the GAN. Several variants were proposed to address this problem by analysing the objective function of the GAN. Wasserstein GAN (WGAN) uses the Earth-Mover (EM) distance to address the vanishing gradient and saturated Jensen-Shannon distance problems while enforcing Lipschitz constraint. However, WGAN can still generate low quality images and fail to converge in many settings. An improvement [36] was proposed to overcome these problems. Although they argued that the performance is more stable at convergence, WGAN is still outperformed by DCGAN [6] in terms of convergent speed and Inception score. A similar solution was introduced in Loss-Sensitive GAN (LS-GAN) [37] with theoretical analysis on Lipschitz densities. They conceptually proved that a GAN loss functions with bounded Lipschitz constants are sufficient to match the model density to true data density. However, objects in their generated CIFAR-10 images are hardly recognizable.

Recently, another subfamily of GAN was introduced by employing an autoencoder in the discriminator. The Energy-based GAN (EBGAN) [15] is trained by replacing the discriminator with an autoencoder and it has demonstrated decent quality synthetic images up to 256×256 pixels. Denoising Feature Matching (DFM) [38] keeps the traditional GAN adversarial loss, while an additional complementary information to the generator is computed using a denoising autoencoder in the feature space learnt by the discriminator. DFM achieved state-of-the-art Inception score on CIFAR-10 in the unsupervised settings. Both works suggested a non-trivial idea that the multi-target information from the reconstruction loss helps to improve the model performance. A closely related work, Boundary Equilibrium GAN (BEGAN) [16] was proposed with a new equilibrium enforcing method. Surprisingly, it demonstrated realistic face generation but is significantly outperformed by DFM on CIFAR-10. This suggests that the traditional adversarial loss remains an important factor to generate realistic complex images.

A. Conditional Image Synthesis

While unconditional image synthesis is an important research area, many practical applications require the model to be conditioned on some prior information. This prior information has many forms, for instance a distorted image for inpainting [32], [39]; natural image for super-resolution [8] or style transfer [40]–[42]; text codes for text to image translation [10], [11]. In this work, we are particularly interested in class-conditioned image generation, as the aim of this work is to investigate how a model learns the representations of the styles, genres, and artists from the artworks.

An earlier work that employed conditional setting in GAN was the Conditional GAN (CondGAN) [5] by feeding the labels or modes to the generator and discriminator. However, such setting was demonstrated on less complex images i.e. MNIST and faces [43]. While this website[4] unofficially demonstrated generating images on CIFAR-10 using CondGAN, the objects in the generated images are hardly recognizable. This is expected because the labels were not fully utilized, as there is no error information backpropagated from the labels. A similar work was introduced in InfoGAN [44] with the discriminator in InfoGAN replaced by a multi-class classifier. It is shown that the InfoGAN is able to learn disentangled representations in an unsupervised manner but the meaning of the representations are uncontrollable during the training dtage. In addition, this work only demonstrated on the less complex images, i.e. digits and faces.

In addition to the GAN variants, PixelCNN [33], [34] also demonstrated decent results on conditional image generation but is computationally expensive for sampling. Built on the Deep Generator Network (DGN) [45], Plug and Play Generative Networks (PPGN) [46] is able to produce high quality images at high resolution. It allows different generators and condition networks to be hacked together without having to re-train the generators. However, PPGN differs to the other generative models discussed, where images are generated in **one-shot** from the latent codes in the traditional generative models. In PPGN, images are generated by optimizing the latent codes to produce images that highly activate target neuron in the condition network. The sampling procedure is formalized as an approximate Langevin Markov chain Monte Carlo sampler to ensure diversity. Like other sequential approaches, such gradient-based recursive approach may cause unwanted overhead when deployed in some real-world applications, e.g. mobile devices. Nonetheless, they showed that adversarial training is crucial to obtain high quality images.

B. Challenges of painting synthesis

Until now, most of the aforementioned works have been focusing on generating images that contain very structured targets such as digits, faces, objects, etc. On the contrary, this work is interested in generating synthetic paintings based on a desired styles, genres, and artists. Many of these categories are structured abstract or non-figurative. For instance, *Cubism* is one of the most influential visual art styles of the early twentieth century, focusing on visualizing the artists' imagination using geometric forms of cubes. Meanwhile, *Impressionism* marked the beginning of a gradual departure from *Realism*, using loose brush stroke, sketchy lines, and blotches of colors that blend together to create the feeling of impression, giving less attention to the details. On the other hand, some categories require historical knowledge. For instance, artworks painted during the Renaissance period are categorized as *Renaissance* art. Many new ideas and attitudes were introduced in this period, emphasizing on humanism, religion, perspective, balance, and proportion. Recognizing the correlations and differences between these styles are awfully difficult even for human, let alone machine. Tan et al. [47] showed that representations learning on these categories is possible but only at simple visual semantic level with the current techniques, e.g. detecting person, sky, sea, etc.

III. Proposed Method

This section describes the proposed method in detail. First, we revisit the traditional GAN [1] model. Then, we depict the formulation of the proposed ArtGAN. The overall architecture is depicted in Figure 1.

[4] http://soumith.ch/eyescream/

A. Preliminaries: Generative adversarial networks

Generative Adversarial Networks (GAN) [1] contains two networks that are trained by competing with each other. The Generator G aims to learn the true data distribution p_{data} by generating images $G(\mathbf{z})$ that are difficult to differentiate from real images $\hat{\mathbf{x}} \sim p_{data}$. Traditionally, G generates images from some noise vectors $\mathbf{z} \sim p_{noise}$ that are sampled from a distribution p_{noise} (e.g. uniform distribution). On the other hand, the Discriminator D is trained to distinguish the images generated by G from the real images. Overall, the training procedure is a two-player min-max game with the following objective function,

$$\min_{G} \max_{D} \mathbb{E}_{\hat{\mathbf{x}} \sim p_{data}}[\log D(\hat{\mathbf{x}})] + \mathbb{E}_{\mathbf{z} \sim p_{noise}}[\log(1 - D(G(\mathbf{z}))] \tag{1}$$

B. The ArtGAN

The basic structure of ArtGAN is similar to GAN, such that it consists of a discriminator and a generator that are simultaneously trained using the minmax formulation of GAN, as described in Eq. 1. The key innovation of ArtGAN is to allow feedback from the labels given to each generated image through the loss function. That is, additional label information is fed to the generator to draw a specific subject based on the information, imitating how human learns to draw. This is in contrast to CondGAN [5] that does not fully utilize the labels during training. In order to leverage the labels information, the discriminator is extended to *categorical autoencoder-based discriminator* to output $K + 1$ logistic predictions with K actual categories following the dataset used, and $K + 1^{th}$ output as the adversarial class (denoted as Fake category).

Formally, the formulation of a categorical discriminator is written as $D : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{K+1}$, where H, W, and C are the height, width, and number of channels of an image, respectively. This is somehow similar to Salimans et al. [24], except that the conditional setting is not implemented in their work. While the notations of the *conditional generator* is written as $G : (\mathbf{z}, \bar{c}) \rightarrow \mathbb{R}^{H \times W \times C}$, where $\bar{c}$ is the randomly chosen label for the generated sample in the form of one-hot vector. This allows the generator to learn better from the feedback labels information. Following Salimans et al. [24], we modify the categorical discriminator such that D becomes the standard supervised classifier with K outputs, $D : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{K}$. Let $l_k(\mathbf{x}) \in D(\mathbf{x})$ be the output of $D(\mathbf{x})$ at class k without activation function and $\mathbf{x}$ is an input image (either from real data or generator). The probability distribution over K classes is given as $p(\mathbf{c}|\mathbf{x})$, such that the predicted probability for each class k is defined as a Softmax function,

$$p(c_k|\mathbf{x}) = \frac{e^{l_k}}{\sum_{i=1}^{K} e^{l_i}} \tag{2}$$

The probability distribution function for the binary adversarial prediction $p(y|\mathbf{x})$ of the discriminator is then reformulated as

$$p(y|\mathbf{x}) = \frac{Z(\mathbf{x})}{Z(\mathbf{x}) + 1} \tag{3}$$

where $Z(\mathbf{x}) = \sum_{i=1}^{K} e^{l_i}$ and $p(y|\mathbf{x}) = 1$ infers that the image $\mathbf{x}$ is real. The benefit of such setting is that the number of parameters can be reduced to relax the over-parametrization problem without changing the output of the Softmax, conceptually. The D is then trained by minimizing the following discriminator loss function $\mathcal{L}_D$,

$$\begin{aligned}\mathcal{L}_D = &- \mathbb{E}_{\hat{\mathbf{x}}, \hat{c} \sim p_{data}} \left[\sum_{i=1}^{K} \hat{c}_i \log p(c_i|\hat{\mathbf{x}}) + \log p(y|\hat{\mathbf{x}}) \right] \\ &- \mathbb{E}_{\mathbf{z} \sim p_{noise}, \bar{c}} \left[\log(1 - p(y|G(\mathbf{z}, \bar{c}))) \right]\end{aligned} \tag{4}$$

where $\hat{c}$ is the ground truth one-hot label of the given real image $\hat{\mathbf{x}}$. The generator loss function $\mathcal{L}_G$ to be minimized for training G is defined as,

$$\mathcal{L}_G = - \mathbb{E}_{\mathbf{z} \sim p_{noise}, \bar{c}} \left[\sum_{i=1}^{K} \bar{c}_i \log p(c_i|G(\mathbf{z}, \bar{c})) + \log(p(y|G(\mathbf{z}, \bar{c}))) \right] \tag{5}$$

Inspired by recent works [15], [16], [38], we incorporates an autoencoder into the categorical discriminator in ArtGAN for additional complementary information. The core idea of using the autoencoder in the discriminator is that reconstruction-based output offers diverse targets, which produce very different gradient directions within the minibatch. This conceptually improves the efficiency and effectiveness when training a GAN model. Rather than deploying two separate computationally expensive networks (a categorical discriminator and an autoencoder separately), our networks partly share the same architecture and weights. In specific, the *encoder* in the autoencoder is shared by the categorical discriminator, as shown in Figure 1. The formulations of the categorical autoencoder-based discriminators are described in three ways. The first two types, namely as the ArtGAN-EB and the ArtGAN-AE are implemented using the pixel-level autoencoder, similar to the EBGAN [15]. However, these two models are differed by the discriminator loss functions formulation. The third type, namely the ArtGAN-DFM is an extension of the Denoising Feature Matching (DFM) [38] to a conditional setup, forming a *Conditional DFM*. The ArtGAN variants are summarized in Figure 2 and the detailed loss functions formulations for each of them are described next. Meanwhile, analysis and comparisons between these ArtGAN variants will be discussed in the experimental section.

ArtGAN-EB: EBGAN [15] is formulated according to the energy-based models by replacing the discriminator with an autoencoder, such that $D_{AE}(\cdot) = Dec(Enc(\cdot))$, where Dec and Enc are the decoder and encoder, respectively. The discriminator loss $\mathcal{L}_{Deb}$ in the EBGAN is given as,

$$\begin{aligned}\mathcal{L}_{Deb} = &\mathbb{E}_{\hat{\mathbf{x}} \sim p_{data}} \left[||D_{AE}(\hat{\mathbf{x}}) - \hat{\mathbf{x}}|| \right] \\ &+ \mathbb{E}_{\mathbf{z} \sim p_{noise}} \left[\max(0, m - ||D_{AE}(G(\mathbf{z})) - G(\mathbf{z})||) \right]\end{aligned} \tag{6}$$

Fig. 2: Different ArtGAN variants (bottom row) compared to the state-of-the-art models (top row). The discriminator in ArtGAN outputs the class predictions and the loss function is computed from the true labels, instead of taking the true labels as input as depicted in the CondGAN. Hence, the true labels can be leveraged to train the discriminator and generator. Meanwhile, ArtGAN-EB and ArtGAN-AE share the same model, which combines ArtGAN and EBGAN by sharing the encoder. However, decoder in the ArtGAN-AE is not trained using the generated samples, contrast to the ArtGAN-EB. The ArtGAN-DFM depicts the extension from DFM with conditional settings.

where $||\cdot||$ is a Euclidean norm, and m as a positive margin. While the generator loss $\mathcal{L}_{Geb}$ is formulated as,

$$\mathcal{L}_{Geb} = \mathbb{E}_{\mathbf{z}\sim p_{noise}}\Big[||D_{AE}(G(\mathbf{z})) - G(\mathbf{z})||\Big] \tag{7}$$

In order to formulate a conditional energy-based loss function, ArtGAN-EB propose a novel discriminator loss function $\mathcal{L}_{Debc}$ as,

$$\mathcal{L}_{Debc} = \mathcal{L}_D + \mathcal{L}_{Deb} \tag{8}$$

While, the new generator loss $\mathcal{L}_{Gae}$ is defined as,

$$\mathcal{L}_{Gae} = \mathcal{L}_G + \mathcal{L}_{Geb} \tag{9}$$

ArtGAN-AE: The discriminator loss is similar to $\mathcal{L}_{Debc}$, except that we do not use the generated images as adversarial samples to update the decoder. This was inspired by DFM [38] to use the autoencoder as a source of *complementary information* when updating the generator, instead of using the autoencoder as an adversarial function (as in [15]). Hence, the discriminator loss $\mathcal{L}_{Dae}$ of ArtGAN-AE is formulated as,

$$\mathcal{L}_{Dae} = \mathcal{L}_D + \mathbb{E}_{\hat{\mathbf{x}}\sim p_{data}}\Big[||D_{AE}(\hat{\mathbf{x}}) - \hat{\mathbf{x}}||\Big] \tag{10}$$

Meanwhile, ArtGAN-AE shares the same generator loss as the ArtGAN-EB.

ArtGAN-DFM: In DFM [38], an additional denoising autoencoder (or denoiser) $r(\cdot)$ is employed to update the generator. The denoiser is trained separately from the discriminator. In specific, the denoiser is trained on the discriminator's hidden state when evaluated on the training data. Formally, D is updated according to Eq. 1. Given that $\Phi(\cdot)$ is a hidden state from $D(\cdot)$, the denoiser is trained by minimizing the following loss function $\mathcal{L}_r$,

$$\mathcal{L}_r = \mathbb{E}_{\hat{\mathbf{x}}\sim p_{data}}\Big[||\Phi(\hat{\mathbf{x}}) - r(\Phi(\hat{\mathbf{x}}))||\Big] \tag{11}$$

Then, the generator is trained with the loss function $\mathcal{L}_{Gdfm}$,

$$\begin{aligned}\mathcal{L}_{Gdfm} =& \mathbb{E}_{\mathbf{z}\sim p_{noise}}\Big[\lambda_{denoise}||\Phi(G(\mathbf{z})) - r(\Phi(G(\mathbf{z})))|| \\ &- \lambda_{adv}\log D(G(\mathbf{z}))\Big]\end{aligned} \tag{12}$$

The authors [38] suggested to fix $\lambda_{adv} = 1$ and set $\lambda_{denoise} = 0.03/n_h$, where n_h is the number of discriminator hidden units fed to the denoiser as input. Here, the modification is straightforward using the categorical discriminator as the discriminator network. Hence, the discriminator loss is same as Eq. 4, and the denoiser loss remains unchanged (Eq. 11). While, the generator loss $\mathcal{L}_{Gdfmc}$ for the conditional DFM is defined as,

$$\mathcal{L}_{Gdfmc} = \mathbb{E}_{\mathbf{z}\sim p_{noise}}\Big[\lambda_{denoise}||\Phi(G(\mathbf{z})) - r(\Phi(G(\mathbf{z})))||\Big] + \mathcal{L}_G \tag{13}$$

IV. Magnified Learning

In order to synthetically generate a high quality image, we proposed a novel magnified learning. The motivation behind the proposed magnified learning is to generate a set of pixel values to vote for a single pixel. In this context, the proposed magnified learning is defined as *a process of training a generative model by generating samples at higher dimensional feature space, while evaluate the samples at original dimension (i.e. same as the true data).*

In specific, suppose a generator in the traditional GAN trained on CIFAR-10 usually generates 32×32 pixels images, $G : \mathbf{z} \rightarrow \mathbb{R}^{32\times32\times C}$. By using magnified learning, the generator is formulated to generate 64×64 pixels images, $G : \mathbf{z} \rightarrow \mathbb{R}^{64\times64\times C}$. This can be done by adding an extra upsampling layer between the existing layers in the generator. Meanwhile, the input image size of the discriminator remains the same as to the original size, such that D :

Fig. 3: **Magnified learning** using *overlapped average pooling*. Pixels in the same block from B (e.g. $[B_1, \cdots, B_9]$) will vote for one pixel in A (e.g. A_1) through averaging. During training, the same gradient (i.e. from A_1) will be backpropagated to all pixels in the block (i.e. $[B_1, \cdots, B_9]$), which helps the pixels in B to learn better correlations between themselves. Similarly, the neighbouring pixels in A will also be correlated due to the *overlapped* pooling operation.

$\mathbb{R}^{32\times32\times C} \rightarrow \mathbb{R}^{K}$. When evaluating the generated samples with the discriminator, the samples are downsampled, such that $\pi : \mathbb{R}^{64\times64\times C} \rightarrow \mathbb{R}^{32\times32\times C}$ where $\pi(\cdot)$ is a downsampling operation.

In this work, overlapped average pooling is used as the downsampling operation. This pooling operation can be viewed as a form of voting system, as shown in Figure 3. An advantage of this design is that the correlations between the neighbouring pixels can be learnt better. This helps the model to learn to draw finer details, resulting in improved image quality. Overlapping the pooling operations discourages the generator to compute pixel values that collapse within the same block, i.e. pixels of the same pooling block have exactly same value. Overall, by using overlapped average pooling in magnified learning, the generator is regularized with two seemingly contradictory constraints: 1) the pixels within the same pooling block should have similar values so that the generated image looks smooth across same color (i.e. the blue coloured sky looks smooth); 2) these pixels must not be naively computed to produce exactly same value that could cause excessive artefacts in the images.

In addition, the magnified learning also supports generating images at higher resolution without the need of using dataset that contains high resolution images. Unlike PPGN [46] which requires a computationally expensive recurrent sampling procedure to generate one image, ArtGAN with magnified learning, namely as the ArtGAN-M is able to generate high quality images at high resolution in *one-shot*. We argue that one-shot procedure is more desired when a model is deployed in a real-world application as not all embedded systems has high computational power, e.g. a mobile device.

V. Experimental results

A. Experimental settings

This section describes the settings that are shared by all experiments, unless stated otherwise. All networks are trained with the Adam optimizer [48] with initial learning rate of 0.0002, $\beta_1 = 0.5$, and minibatch size of 100. The learning rate is decreased once by a factor of 10 after iteration $30,000$. Input noise vector $\mathbf{z}$ is a 100-dimensional multivariate random variable sampled using an i.i.d. uniform distributed random generator $U(-1, 1)$. Instance noise [35] is implemented in all discriminators for better training stability. During magnified learning, all samples are generated at twice as high as the resolution of the non-magnified learning counterparts. For fair comparisons, we run one gradient descent step for each player in each iteration. This usually works better than running more steps of one player than the other [49]. In practice, it is very difficult to determine how many more steps to run as the performance is usually inconsistent with the same setting on different datasets. The rest of the settings will be described in other sections. The experiments were conducted using Tensorflow [50] with one Titan X (Maxwell) GPU.

In order to assess the performance of the proposed models, Inception score is adopted [24] for quantitative measurement. Intuitively, Inception score measures the *objectness* by minimizing entropy per-sample posterior (i.e. each sample is classified with high certainty), as well as the *class diversity* by maximizing the entropy aggregate posterior (i.e. the classifier used in Inception score identifies wide variety of classes among the samples). However, we do need to note that the *class diversity* metric becomes meaningless in the conditional setting as the conditional generative models will almost always generate visually different images for different modes. In addition, the *class diversity* metric can be misleading, i.e. it can be maximized (higher is better) and fooled when the generated samples have uniform distribution across all classes prediction. Hence, we split the measurements (*objectness* and *class diversity* metrics) when we report the scores for a better performance assessments.

Since Inception score is measured mainly based on the objectness of an image, therefore it is unsuitable for assessing the model performance on the artworks. Meanwhile, evaluation of generative model based on the state-of-the-art log-likelihood estimates can be misleading [51]. Hence, the comparative studies are first conducted using the *objectness* metric from Inception score on CIFAR-10 [13] and STL-10 [19] datasets. The experiments report state-of-the-art results on these datasets based on the *objectness* and Inception scores. Then, Wikiart dataset [23], [47] is used to train the best model found for artworks synthesis based on the genres, artists, and styles. In addition, we also trained the model on Oxford-102 [20] and CUB-200 [21] for additional performance assessments.

We used similar design to BEGAN [16] by employing nearest neighbour upsampling instead of strided deconvolution layer in the generator as suggested by Odena et al. [52] in order to avoid checkerboard artifacts. Between the upsampling layers are at least one layer of convolutional layer. The discriminator has the same design as to the traditional GAN with multiple layers of strided convolutional layers. Batch normalization and leaky ReLU are used for both the discriminator and generator. Due to page limit, detailed network descriptions and additional generated samples are available in the appendix. The list of models to be evaluated are as follows:

1) ArtGAN - Baseline model.

2) ArtGAN-EB - The first type of categorical autoencoder-based discriminator.
3) ArtGAN-AE - The second type of categorical autoencoder-based discriminator.
4) ArtGAN-DFM - The third type of categorical autoencoder-based discriminator.
5) ArtGAN-M - ArtGAN with **magnified learning**.
6) ArtGAN-D - ArtGAN with deeper architecture (more layers and number of parameters). This is implemented to verify that network size is not the main factor that contributes to the improvements observed in the experiments when using magnified learning.
7) ArtGAN-AEM - ArtGAN-AE with **magnified learning**.
8) ArtGAN-AEMT - Huang et al. [53] employed a trick by updating more steps for the generator per each discriminator update step. Although it is hard to determine number of steps, their setup seems to work well for CIFAR-10. Hence, the same setting is employed in our CIFAR-10 experiment.

B. Evaluation and quantitative metric

Evaluation of a generative model is extremely difficult as it is still not clear how to quantitatively evaluate a generative model. This is due to the difficulty in estimating the intractable log-likelihood in many models [51]. The most widely used log-likelihood estimator is the Parzen window estimates [54]. However, Theis et al. [51] convincingly argued that this estimator can be quite misleading for high-dimensional data. Recently, Salimans et al. [24] proposed a different way to assess image quality by using the Inception score (higher is better). The formulation of the Inception score is defined as:

$$\begin{aligned} I(\{\mathbf{x}\}_1^N) &= \exp(\mathbb{E}[D_{KL}(p(y|\mathbf{x})||p(y))]) \\ &\approx \exp(-\mathbb{E}[H(p(y|\mathbf{x}))] + \mathbb{E}[H(\mathbb{E}_{\mathbf{x}}(p(y|\mathbf{x})))]) \quad (14) \end{aligned}$$

where $H(\cdot)$ is the Shannon entropy and $D_{KL}(\cdot)$ is the KullbackLeibler divergence. As aforementioned, this metric measures the *objectness* in the first term (lower is better) and *class diversity* in the second term (higher is better) of the samples can be misleading when the *class diversity* metric is fooled. One of this case can be seen in the experiments when we compare ArtGAN (baseline) and ArtGAN-EB in Table I. Although ArtGAN-EB performed better than ArtGAN with higher Inception score (8.26 by ArtGAN-EB compared to 8.21 by ArtGAN-DFM), it has worse *objectness* score (33.51 by ArtGAN-EB compared to 33.24 by ArtGAN). It is clear that the *class diversity* score in the ArtGAN-EB unreliably affected its Inception score as higher *class diversity* score can also imply that the objects in the generated images are hard to recognize. This is especially true when the model has the worst score in the *objectness* metric. Nonetheless, Inception score is still a preferred metric due to the lack of a better alternative for quantitative measurement. Hence, this work adopts Inception score but the performance assessment is done mainly based on the *objectness* score since it is a more reliable metric.

In addition, the generated images will be displayed for visual inspection as human evaluation is always more accurate when accessing the image quality, though can be subjective at times. Furthermore, latent space interpolation on the proposed ArtGAN-AEM is performed to "probe" the structure of the latent space $\mathbf{z}$. The smooth transitions between samples when the latent space is interpolated usually indicates how well the generative models understand the structure of the images.

(a) ArtGAN ($\mathbf{32 \times 32}$) (b) ArtGAN-AE ($\mathbf{32 \times 32}$) (c) ArtGAN-M ($\mathbf{64 \times 64}$) (d) ArtGAN-AEM ($\mathbf{64 \times 64}$)

Fig. 4: Generated CIFAR-10 images. From top to bottom: (1) Airplane, (2) Automobile, (3) Bird, (4) Cat, (5) Deer, (6) Dog, (7) Frog, (8) Horse, (9) Ship, (10) Truck.

C. CIFAR-10

CIFAR-10 [13] is a small, well-studied dataset consisting 32×32 pixels RGB images. It is split into 50,000 training images and 10,000 test images from 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

The models are trained on full image size, i.e. 32×32 pixels, while the generator generates 64×64 pixels images during magnified learning. The models are trained for 70,000 iterations and saved every 1,000 iterations. As shown by Gulrajani et al. [36], the Inception scores of the generative models continue to oscillate with non-negligible amplitude at convergence. Hence, best models found based on the *objectness* score are reported in Table I along with the state-of-the-art results. It can be noticed that comparison to other state-of-the-art generative models, ArtGAN-AEMT obtains state-of-the-art result with a score of 8.81 ± 0.14, outperforming two lastest methods - SGAN [53] (8.59 ± 0.12) and AC-GAN [9] (8.25 ± 0.07). Qualitatively, the proposed models are able to produce many samples with high visual fidelity, especially when magnified learning is employed as shown in Figure 4. Particularly the details are drawn finer, e.g. cats are more recognizable with better ears shape, most of the frogs are drawn with clear contour, etc.

TABLE I: Inception scores on CIFAR-10 evaluated at 32×32 pixels. Scores are reported in the form of *mean score*$\pm$*std.* In the proposed methods, other scores are reported in the form of *objectness*(*class diversity*).

Model	Scores
Unlabelled	
Infusion training [55]	4.62 ± 0.06
ALI [56] (as reported in [38])	5.34 ± 0.05
BEGAN [16]	5.62
GMAN [57]	6.00 ± 0.19
EGAN-Ent-VI [58]	7.07 ± 0.10
LR-GAN [59]	7.17 ± 0.07
Denoising feature matching [38]	7.72 ± 0.13
Labelled	
SteinGAN [60]	6.35
DCGAN (as reported [60])	6.58
Improved GAN [24]	8.09 ± 0.07
AC-GAN [9]	8.25 ± 0.07
SGAN [53]	8.59 ± 0.12
Proposed methods	
ArtGAN (baseline)	8.21 ± 0.08 33.24 (272.90)
ArtGAN-EB	8.26 ± 0.10 33.51 (276.60)
ArtGAN-AE	8.43 ± 0.09 31.09 (262.04)
ArtGAN-DFM	8.25 ± 0.09 33.34 (274.99)
ArtGAN-M	8.50 ± 0.06 30.19 (256.62)
ArtGAN-D	8.29 ± 0.10 33.30 (276.15)
ArtGAN-AEM	8.53 ± 0.09 **30.07** (256.42)
ArtGAN-AEMT	$\mathbf{8.81} \pm 0.14$ 30.65(269.83)
Real data	11.24 ± 0.12 24.32 (271.76)

D. STL- 10

STL-10 [19] is a dataset inspired by CIFAR-10 with higher image resolution of 96×96 pixels. However, it contains fewer labelled training examples and a very large set of unlabelled examples. Although STL-10 is primarily used to develop unsupervised features learning, we stay focus on our goal in this paper and use STL-10 for conditional image synthesis in a supervised fashion. In particular, we only use the labelled examples during training, which contains only 5,000 samples from 10 classes: airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck. As such, it makes STL-10 a more challenging dataset than CIFAR-10.

During training, we randomly cropped 84×84 pixels from the 96×96 pixels images, then the images are resized and trained at 64×64 resolution. Note that different areas are cropped in different iterations. The models trained using magnified learning will generate samples at 128×128 pixels resolution. All models are trained for 50,000 iterations. Similar to CIFAR-10, models are saved every 1,000 iterations and the best models are reported. The Inception scores are reported in Table II, while the generated samples are shown in Figure 5. It is noticed that the synthetic images at higher resolution (i.e. 128×128) are clearer and sharper without much artefacts. No mode collapse is observed in this part of the experiments.

TABLE II: Inception scores on STL-10 evaluated at 64×64 pixels. Readers may refer to Table I for scores descriptions.

Model	Scores
ArtGAN (baseline)	9.72 ± 0.14 31.03 (301.63)
ArtGAN-EB	9.73 ± 0.12 30.22 (293.89)
ArtGAN-AE	9.65 ± 0.08 31.04 (299.50)
ArtGAN-DFM	9.63 ± 0.09 31.25 (300.89)
ArtGAN-M	$\mathbf{10.12} \pm 0.09$ 29.05 (293.90)
ArtGAN-D	9.87 ± 0.09 31.03 (306.39)
ArtGAN-AEM	10.07 ± 0.09 **28.18**(283.81)
Real data	15.48 ± 0.76 15.04 (232.17)

E. More ablation studies

In order to further understand the effect of different model components, we conduct extensive ablation studies by comparing the performances of the proposed models on CIFAR-10 (Table I) and STL-10 (Table II) datasets. Note that the performances are evaluated based on the *objectness* metric, unless specified otherwise. Below we summarize some of our findings.

First, the performance of magnified learning can be assessed by comparing the ArtGAN-M and the ArtGAN-D. Note that the ArtGAN-D is not overfit since its performance is similar to the baseline (ArtGAN). In addition, the ArtGAN-D has more number of parameters than the ArtGAN-M. Hence, it shows that the additional parameter numbers are not the main factor that contribute to the improvement in the ArtGAN-M. These results also show that magnified learning improves the generated image quality with significantly better *objectness* and Inception score when compared to the ArtGAN baseline.

Secondly, the ArtGAN-DFM performed slightly poorer than the baseline. In the ArtGAN-DFM, the features fed to the denoiser are extracted from the discriminator which is still in training. Hence, we speculate that measuring the loss using these underdeveloped features might cause instability when training the denoiser and generator. Therefore, it is encouraged to compute the losses by leveraging the true data directly.

Thirdly, performance inconsistency can be seen in the ArtGAN-EB, where it performed better on one dataset but worse on the other. This suggests that additional adversarial loss does not always complement a model. This is because the underdeveloped adversarial samples may provide noisy information that can hamper the training process. Hence, extra adversarial loss can aggravate this problem. Without the aforementioned problems, the ArtGAN-AE exhibited more consistent performances with either better or comparable scores. Readers should also be noted that we tried to train another model using only the Energy-based adversarial loss (traditional adversarial loss is removed). We found that this model failed to learn, producing collapsed and meaningless images. This hints that traditional adversarial loss is still a better choice for adversarial training in our settings. Hence,

(a) ArtGAN ($\mathbf{64 \times 64}$) (b) ArtGAN-AE ($\mathbf{64 \times 64}$) (c) ArtGAN-M ($\mathbf{128 \times 128}$) (d) ArtGAN-AEM ($\mathbf{128 \times 128}$)

Fig. 5: Generated STL-10 images. From top to bottom: (1) Airplane, (2) Bird, (3) Car, (4) Cat, (5) Deer, (6) Dog, (7) Horse, (8) Monkey, (9) Ship, (10) Truck.

Fig. 6: Sample generated images on CUB-200 birds. Left (red box): Real samples; Middle (blue box): Generated samples (64×64); Right: Generated samples (128×128).

the ArtGAN-AE model is used for the rest of the experiments.

Finally, ArtGAN-AEM, which is the ArtGAN-AE with magnified learning, achieved the best results with consistent and significant improvements. However, it has lower *objectness* score to the ArtGAN-AEM. This shows the unreliability of the Inception score.

F. CUB-200

Caltech-UCSD Birds-200-2011 (CUB-200) [21] is an image dataset with 200 bird species and a total of 11,788 images. It also has the annotations for 15 part locations, 312 binary attributes, and one bounding box per image. However, these attributes are not used in this work as the design of the proposed model is not suitable for multi-label problem in its current state. Hence, the ArtGAN-AEM is trained based on the 200 bird species classes.

The images are pre-processed in the similar way as to the experiments on Oxford-102, i.e. model is trained at 64×64 resolution after cropping and resizing. In order to avoid mode collapse as to experimenting the Oxford-102, the model follows the same settings by randomly choosing 20 classes in each iteration with 5 samples per class. Generated images sample are shown in Figure 6. Similar to the Oxford-102 dataset, the discriminator has a poor performance on bird species classification ($\sim 20\%$ accuracy). Interestingly, the figures show that the proposed model is still able to recognize and draw the characteristics of different bird species, e.g. colors, shape, and body size. However, the body structures of the birds are not well-learnt.

G. Oxford-102

Oxford-102 [20] consists of 102 flower categories, with around 40 and 258 images in each class. The images have large scale, pose, and light variations and some categories are very similar to each other. The model was trained for 30,000 iterations with learning rate reduced after iteration 15,000. The images were saved at resolution of 256×256 and randomly cropped to 224×224. Then, the cropped images are resized to 64×64 for training.

Two experiments were conducted where the first experiment trained with batch size of 102. In the generator, one sample is drawn for each class during the training. We found out that the image quality is high but it experienced mode collapse, i.e. generated images look almost exactly the same within a class. In the second experiment, 20 classes are randomly

Fig. 7: Sample generated images on Oxford-102 flowers. Left (red box): Real samples; Middle (blue box): Generated samples (64×64); Right: Generated samples (128×128).

chosen in each iteration and hence, 5 samples are drawn for each class during the training. This solves the mode collapse problem, which suggests that more adversarial images should be sampled for each class in the same iteration to learn more diverse correlations between the latent codes and the image space. Sample generated images can be visualized in Figure 7. Although the discriminator performed poorly on classifying the flower species ($\sim 50\%$ accuracy), the figures show that ArtGAN-AEM is able to generate high quality flower images that look natural with distinctive species-typical features, i.e. color and shape.

H. WikiArt

Wikiart is a fine-art paintings dataset first introduced by Saleh et al. [23]. The paintings were obtained from the wikiart.org website. Currently, Wikiart is the largest public dataset available that contains around 80,000 annotated paintings for genres, artists and styles classification tasks. However, not all paintings are used in all tasks. To be specific, all paintings are used for 27 *styles* classification. But, there are only 60,000 paintings annotated for 10 *genres*, and only around 20,000 paintings are annotated for 23 *artists*. We released an extended version of Wikiart dataset by randomly splitting the dataset to training and test sets, and this version will be used in our experiments.

The paintings were stored in 256×256 resolution and randomly cropped to 224×224 resolution, then resized to 64×64 resolution for training. Three different ArtGAN-AEM models were trained for different tasks (*styles*, *genres*, and *artists*) for 50,000 iterations. The results are reported using the last updated models (i.e. models at iteration 50,000). In general, ArtGAN-AEM is able to learn artistic representations and generate high quality paintings. Detailed discussions are as follows:

1) ***Genre****:* Generated paintings based on genre are visualized in Figure 8. Out of the three tasks, categorizing genres is the easiest task [47]. Hence, it is expected that ArtGAN-AEM is able to draw many meaningful paintings based on the genre. For instance, anyone should be able to differentiate *abstract paintings*, *cityscape*, *landscape*, and *portraits* from other classes. The synthesized paintings show that ArtGAN-AEM is able to recognize and draw high quality paintings on these genres. An interesting observation can also be seen in *genre painting*. Not to be confused with "genre", "genre paintings" is the pictorial representation of scenes or events from everyday life, such as markets, parties, etc. Hence, a group of people can usually be seen in this kind of paintings. Figure 8 shows that ArtGAN-AEM is able to draw several human-like figures in a few synthetic paintings. The model may not be able to understand the true meaning of *genre paintings*, but it shows that the model is able to find some similarities in genre paintings at semantic level.

Fig. 8: Generated *genres* images at $\mathbf{128} \times \mathbf{128}$ pixels. From top to bottom: (1) Abstract, (2) Cityscape, (3) Genre, (4) Illustration, (5) Landscape, (6) Nude, (7) Portrait, (8) Religious, (9) Sketch and study, (10) Still life.

2) ***Artist****:* Figure 9 shows the synthetic paintings based on artist. Learning visual representations in this task is not impossible as artists usually have their own preferences when deciding what to draw, what kind of styles to use, etc. Hence, many visual similarities can be found between artworks of the same artist. First example can be seen in the paintings of *Nicholas Roerich*. He is a Russian who settled in Himachal Pradesh, India (a mountainous state) for a long time. Hence, many of his famous masterpieces depict the beauty of the mountains with expressive colors and fluid brushwork, as shown in the synthesized paintings. Meanwhile, the figures also clearly show *Gustave Dore*'s primary approach in engraving, etching, and lithography, which result in greyish artworks. However, the model generated many colourless paintings when conditioned on *Vincent van Gogh*. After some investigations, we found an interesting fact that more than half of his artworks were annotated as *sketch and study* genre. By including all his artworks, most *Van Gogh*s palette consisted mainly of sombre

Fig. 9: Generated *artists* images at $\mathbf{128} \times \mathbf{128}$ pixels. From left to right: (1) Albrecht Durer, (2) Boris Kustodiev, (3) Camille Pissarro, (4) Childe Hassam , (5) Claude Monet, (6) Edgar Degas, (7) Eugene Boudin, (8) Gustave Dore, (9) Ilya Repin, (10) Ivan Aivazovsky, (11) Ivan Shishkin, (12) John Singer Sargent, (13) Marc Chagall, (14) Martiros Saryan, (15) Nicholas Roerich, (16) Pablo Picasso, (17) Paul Cezanne, (18) Pierre Auguste Renoir, (19) Pyotr Konchalovsky, (20) Raphael Kirchner, (21) Rembrandt, (22) Salvador Dali, (23) Vincent van Gogh.

earth tones, particularly dark brown, and showed no sign of the vivid colours that distinguish from his later work, e.g the famous The Starry Night masterpiece. This explains the behaviour of the trained model, but is not desired as the striking colour, emphatic brushwork, and the contoured forms of his work that powerfully influenced the current of Expressionism in modern art is not well-learnt by the model. *Eugene Boudin* is a marine painter and has always favoured rendering the sea and along its shores in his artworks. Meanwhile, *Ivan Shishkin* became famous for his forest landscapes. These preferences can be seen in the synthesized paintings.

3) ***Style****:* Synthetic paintings based on style is shown in Figure 11. Out of the three task, styles classification is seem to be the most difficult task. In addition to the difficulty in recognizing *Renaissance* art as explained in Section II, differentiating *Baroque* and *Rococo* is also challenging as they are historically related. They are generally differentiated by the "feelings" they give to their viewers. *Baroque* art often depicts violence, darkness, and the nudes more plump than in *Rococo* works. During mid-1700s, artists gradually moved away from *Baroque* into the modern *Rococo* style. *Rococo* art was often light-hearted, pastoral, and a rosy-tinted view of the world. A subjective observation can be seen in Figure 11 such that *Baroque* synthetic arts are drawn with darker color than the *Rococo* counterparts. This suggests that ArtGAN-AEM do learnt some characteristics in these styles through the color intensity. Meanwhile, *Ukiyo-e* is a type of Japanese art flourished from the 17th through 19th centuries. Generally, *Ukiyo-e* is produced using the woodblock printing for mass production and a large portion of these paintings appear to be yellowish due to the paper material. Such characteristic can be seen in the generated Ukiyo-e style paintings.

I. Latent space interpolation

In this section, we would like to show that ArtGAN is not simply memorizing training data, but can truly generate novel images. Walking on the manifold of the latent space $\mathbf{z}$ can examines the signs of memorization, i.e. sharp image transitions along the latent space indicates high probability that the model is memorizing the true data space. This is an undesired property as it also implies that the relation between the latent codes and image space is not well learnt. Figure

Fig. 10: Interpolations over the latent space $\mathbf{z}$ in Wikiart, CIFAR-10, STL-10, Oxford-102 and CUB-200 datasets. Samples show smooth transitions and each image looks plausible.

Fig. 11: Generated *styles* images at $\mathbf{128} \times \mathbf{128}$ pixels. From top to bottom, **Left**: (1) Abstract Expressionism, (2) Action painting, (3) Analytical Cubism, (4) Art Nouveau, (5) Baroque, (6) Color Field Painting, (7) Contemporary Realism, (8) Cubism, (9) Early Renaissance; **Middle**: (10) Expressionism, (11) Fauvism, (12) High Renaissance, (13) Impressionism, (14) Mannerism Late Renaissance. (15) Minimalism, (16) Naive Art Primitivism, (17) New Realism, (18) Northern Renaissance; **Right**: (19) Pointillism, (20) Pop Art, (21) Post Impressionism, (22) Realism, (23) Rococo, (24) Romanticism, (25) Symbolism, (26) Synthetic Cubism, (27) Ukiyo-e.

10 shows that the samples generated have smooth semantic changes and look plausible. This indicates that the model is not memorizing and has learnt relevant, interesting, and rich visual representations.

VI. Conclusion

In this work, we proposed a novel GAN variant called **ArtGAN** which leverage the labels information for better representation learning and image quality. In addition, its extension called ArtGAN-AEM was introduced by employing the newly proposed categorical autoencoder-based discriminator and magnified learning. Empirical experiments showed that the proposed ArtGAN-AEM achieved the state-of-the-art results on CIFAR-10 and STL-10. Furthermore, the figures visualized in this paper showed the superiority of the proposed ArtGAN-AEM in generating high quality and plausibly looking images. More importantly, the generated paintings showed that ArtGAN-AEM is able to learn artistic representations from the Wikiart paintings that are usually non-figurative and structured abstract. ArtGAN-AEM is able to generate many high quality fine-art paintings based on the given *style*, *genre*, or *artist*. For future work, we are looking forward to extend this work for other interesting applications, such as natural to artistic image translation based on a desired semantic-level mode, e.g. *style*.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[5] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[7] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," *arXiv preprint arXiv:1511.06390*, 2015.

[8] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *NIPS*, 2015, pp. 1486–1494.

[9] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," *arXiv preprint arXiv:1610.09585*, 2016.

[10] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *ICML*, vol. 3, 2016.

[11] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *arXiv preprint arXiv:1612.03242*, 2016.

[12] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998.

[13] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[15] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[16] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[17] Y. Wang, L. Zhang, and J. van de Weijer, "Ensembles of generative adversarial networks," *arXiv preprint arXiv:1612.00991*, 2016.

[18] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," *arXiv preprint arXiv:1704.00109*, 2017.

[19] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[20] 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.

[21] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[22] W. R. Tan, C. S. Chan, H. Aguirre, and K. Tanaka, "Artgan: Artwork synthesis with conditional categorial gans," in *ICIP*, 2017.

[23] B. Saleh and A. Elgammal, "Large-scale classification of fine-art paintings: Learning the right metric on the right feature," *arXiv preprint arXiv:1505.00855*, 2015.

[24] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *NIPS*, 2016, pp. 2226–2234.

[25] V. Mnih, G. E. Hinton *et al.*, "Generating more realistic images using gated mrf's," in *NIPS*, 2010, pp. 2002–2010.

[26] N. Le Roux, N. Heess, J. Shotton, and J. Winn, "Learning a generative model of images by factoring appearance and shape," *Neural Computation*, vol. 23, no. 3, pp. 593–650, 2011.

[27] H. Shim, "Probabilistic approach to realistic face synthesis with a single uncalibrated image," *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3784–3793, 2012.

[28] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[29] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *NIPS*, 2014, pp. 3581–3589.

[30] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *NIPS*, 2013, pp. 899–907.

[31] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[32] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.

[33] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves *et al.*, "Conditional image generation with pixelcnn decoders," in *NIPS*, 2016, pp. 4790–4798.

[34] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *arXiv preprint arXiv:1701.05517*, 2017.

[35] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár, "Amortised map inference for image super-resolution," *arXiv preprint arXiv:1610.04490*, 2016.

[36] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[37] G.-J. Qi, "Loss-sensitive generative adversarial networks on lipschitz densities," *arXiv preprint arXiv:1701.06264*, 2017.

[38] D. Warde-Farley and Y. Bengio, "Improving generative adversarial networks with denoising feature matching," *ICLR submissions*, vol. 8, 2017.

[39] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016, pp. 2536–2544.

[40] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016.

[41] N. Wang, X. Gao, L. Sun, and J. Li, "Bayesian face sketch synthesis," *IEEE Transactions on Image Processing*, vol. 26, no. 3, pp. 1264–1274, 2017.

[42] M. Elad and P. Milanfar, "Style transfer via texture synthesis," *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2338–2351, 2017.

[43] J. Gauthier, "Conditional generative adversarial nets for convolutional face generation," *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, vol. 2014, no. 5, p. 2, 2014.

[44] C. Xi, D. Yan, H. Rein, S. John, S. Ilya, and A. Pieter, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *arXiv preprint arXiv:1606.03657*, 2016.

[45] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *NIPS*, 2016, pp. 3387–3395.

[46] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune, "Plug & play generative networks: Conditional iterative generation of images in latent space," *arXiv preprint arXiv:1612.00005*, 2016.

[47] W. R. Tan, C. S. Chan, H. E. Aguirre, and K. Tanaka, "Ceci n'est pas une pipe: A deep convolutional network for fine-art paintings classification," in *ICIP*. IEEE, 2016, pp. 3703–3707.

[48] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[49] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[50] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[51] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," *arXiv preprint arXiv:1511.01844*, 2015.

[52] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[53] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, "Stacked generative adversarial networks," *arXiv preprint arXiv:1612.04357*, 2016.

[54] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[55] F. Bordes, S. Honari, and P. Vincent, "Learning to generate samples from noise through infusion training," *arXiv preprint arXiv:1703.06975*, 2017.

[56] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville, "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016.

[57] I. Durugkar, I. Gemp, and S. Mahadevan, "Generative multi-adversarial networks," *arXiv preprint arXiv:1611.01673*, 2016.

[58] Z. Dai, A. Almahairi, P. Bachman, E. Hovy, and A. Courville, "Calibrating energy-based generative adversarial networks," *arXiv preprint arXiv:1702.01691*, 2017.

[59] J. Yang, A. Kannan, D. Batra, and D. Parikh, "Lr-gan: Layered recursive generative adversarial networks for image generation," *arXiv preprint arXiv:1703.01560*, 2017.

[60] D. Wang and Q. Liu, "Learning to draw samples: With application to amortized mle for generative adversarial learning," *arXiv preprint arXiv:1611.01722*, 2016.

VII. Appendix

This provides additional details which are not covered in the main document. First, we present the pseudocode that is used to train the ArtGAN. Then, we list the detailed model configurations of the Generator and Discriminator used in our work to facilitate future reimplementation of our work. Finally, we show more qualitative results for experiments on Wikiart, CIFAR-10, STL-10, CUB-200, and Oxford-102 datasets.

A. Algorithm

Algorithm 1 illustrates the training process in our ArtGAN models. The notations are consistent with the submission. In addition, we denote $\mathbf{K} = \{1, \ldots, K\}$ as the set of indices of the classes. Then, the one-hot vector of a sample $\bar{c}_k$ is randomly sampled, where $k \in \mathbf{K}$ and value at position k is set to **one** while the rest of the elements are set to **zero**. Given n samples in a minibatch, $\mathbf{y} = \{y_1, \ldots, y_n\}$ is a vector of the computed adversarial outputs. While, $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$ is a set of class prediction.

Algorithm 1 Pseudocode for training ArtGAN

Require: Minibatch size, n, learning rate, λ, and $\mathbf{z}$ vector size, d
Require: Randomly initialize θ_D and θ_G
while condition not met **do**
  Sample $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_n] \sim \mathcal{N}(0,1)^{n \times d}$
  Randomly set $\bar{\mathbf{C}} = [\bar{c}_{k_1}, \ldots, \bar{c}_{k_n}]$
  Sample minibatch $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n]$
  $\mathbf{C}, \mathbf{y} = D(\hat{\mathbf{X}})$
  $\bar{\mathbf{X}} = G(\mathbf{Z}, \bar{\mathbf{C}})$
  $\mathbf{C}', \mathbf{y}' = D(\bar{\mathbf{X}})$
  if use magnified learning **then**
    $\mathbf{R} = Dec(\Phi(\hat{\mathbf{X}}))$
    $\mathbf{R}' = Dec(\Phi(\bar{\mathbf{X}}))$
    $\theta_D = \theta_D - \lambda \frac{\partial \mathcal{L}_{Dae}}{\partial \theta_D}$, $\mathcal{L}_{Dae} \leftarrow \mathbf{y}, \mathbf{C}, \bar{\mathbf{C}}, \mathbf{C}', \mathbf{y}', \mathbf{R}$
    $\theta_G = \theta_G - \lambda \frac{\partial \mathcal{L}_{Gae}}{\partial \theta_G}$, $\mathcal{L}_{Gae} \leftarrow \bar{\mathbf{C}}, \mathbf{C}', \mathbf{y}', \mathbf{R}'$
  else
    $\theta_D = \theta_D - \lambda \frac{\partial \mathcal{L}_D}{\partial \theta_D}$, $\mathcal{L}_D \leftarrow \mathbf{y}, \mathbf{C}, \bar{\mathbf{C}}, \mathbf{C}', \mathbf{y}'$
    $\theta_G = \theta_G - \lambda \frac{\partial \mathcal{L}_G}{\partial \theta_G}$, $\mathcal{L}_G \leftarrow \bar{\mathbf{C}}, \mathbf{C}', \mathbf{y}'$
  end if
end while

B. Network Architectures

1) CIFAR-10: This section describes the network architectures used on CIFAR-10. Table III shows the architecture for the discriminator. Table IV shows the architectures for the generators.

2) STL-10: Table V and Table VI show the network architectures of the discriminator and generator used on STL-10.

3) Wikiart: All tasks in Wikiart (i.e. *genres*, *styles*, and *artists*) used the same architectures described in Table VII and Table VIII.

4) Oxford-102 flowers and CUB-200 birds: Oxford-102 and CUB-200 datasets share the same network architectures as described in Table IX and Table X.

TABLE III: Network architectures of the discriminator used on CIFAR-10, which contains a classifier and a decoder.

Classifier	Decoder
conv(96, 3, 1) convBN(96, 3, 2) Dropout(0.2)	convBN(256, 3, 1)
convBN(192, 3, 1) convBN(192, 3, 2) Dropout(0.2)	NNupsample(16) convBN(128, 3, 1) convBN(128, 3, 1)
convBN(256, 3, 1) convBN(256, 1, 1) convBN(512, 1, 1)	NNupsample(32) convBN(64, 3, 1) conv(3, 3, 1)
fc(10)	

TABLE IV: Network architectures for the generators (with and without magnified learning) used on CIFAR-10.

Generator	Generator with Magnified Learning
fcBN($512 \times 4 \times 4$) NNupsample(8) convBN(256, 3, 1)	fcBN($512 \times 4 \times 4$) NNupsample(8) convBN(256, 3, 1)
NNupsample(16) convBN(128, 3, 1) convBN(128, 3, 1)	NNupsample(16) convBN(128, 3, 1) convBN(128, 3, 1)
NNupsample(32) convBN(64, 3, 1) convBN(3, 3, 1)	NNupsample(32) convBN(64, 3, 1) convBN(64, 3, 1)
	NNupsample(64) convBN(32, 3, 1) conv(3, 3, 1)

TABLE V: Network architectures for discriminator (containing a classifier and a decoder) used on STL-10.

Classifier	Decoder
conv(64, 3, 1) convBN(64, 3, 2) Dropout(0.2)	convBN(512, 3, 1)
convBN(128, 3, 1) convBN(128, 3, 2) Dropout(0.2)	NNupsample(16) convBN(256, 3, 1)
convBN(256, 3, 1) convBN(256, 3, 2) Dropout(0.2)	NNupsample(32) convBN(128, 3, 1) convBN(128, 3, 1)
convBN(512, 3, 1) convBN(512, 3, 1)	NNupsample(64) convBN(64, 3, 1) conv(3, 3, 1)
fc(10)	

TABLE VI: Network architectures for the generators (with and without magnified learning) used on STL-10.

Generator	Generator with Magnified Learning
fcBN($512 \times 4 \times 4$) NNupsample(8) convBN(512, 3, 1)	fcBN($512 \times 4 \times 4$) NNupsample(8) convBN(512, 3, 1)
NNupsample(16) convBN(256, 3, 1)	NNupsample(16) convBN(256, 3, 1)
NNupsample(32) convBN(128, 3, 1) convBN(128, 3, 1)	NNupsample(32) convBN(128, 3, 1) convBN(128, 3, 1)
NNupsample(64) convBN(64, 3, 1) convBN(3, 3, 1)	NNupsample(64 convBN(64, 3, 1) convBN(64, 3, 1)
	NNupsample(128) convBN(32, 3, 1) conv(3, 3, 1)

TABLE VII: Network architectures for discriminator (containing a classifier and a decoder) used on Wikiart.

Classifier	Decoder
conv(128, 3, 2) Dropout(0.2)	convBN(512, 3, 1)
convBN(256, 3, 2) Dropout(0.2)	NNupsample(8) convBN(256, 3, 1)
convBN(512, 3, 2) convBN(512, 3, 1) Dropout(0.2)	NNupsample(16) convBN(128, 3, 1)
convBN(1024, 3, 2)	NNupsample(32) convBN(64, 3, 1)
fc(10)	NNupsample(64) convBN(32, 3, 1) conv(3, 3, 1)

TABLE VIII: Network architectures for the generators (with and without magnified learning) used on Wikiart.

Generator	Generator with Magnified Learning
fcBN($512 \times 4 \times 4$) NNupsample(8) convBN(512, 3, 1)	fcBN($512 \times 4 \times 4$) NNupsample(8) convBN(512, 3, 1)
NNupsample(16) convBN(256, 3, 1)	NNupsample(16) convBN(256, 3, 1)
NNupsample(32) convBN(128, 3, 1)	NNupsample(32) convBN(128, 3, 1)
NNupsample(64) convBN(64, 3, 1) convBN(3, 3, 1)	NNupsample(64 convBN(64, 3, 1) convBN(64, 3, 1)
	NNupsample(128) convBN(32, 3, 1) conv(3, 3, 1)

TABLE IX: Network architectures for discriminator (containing a classifier and a decoder) used on Oxford-102 and CUB-200.

Classifier	Decoder
conv(64, 3, 2) Dropout(0.2)	convBN(512, 3, 1)
convBN(128, 3, 2) Dropout(0.2)	NNupsample(8) convBN(256, 3, 1)
convBN(256, 3, 2) convBN(256, 3, 1) Dropout(0.2)	NNupsample(16) convBN(128, 3, 1)
convBN(512, 3, 2) convBN(512, 3, 1)	NNupsample(32) convBN(64, 3, 1)
fc(K)	NNupsample(64) conv(32, 3, 1) conv(3, 3, 1)

TABLE X: Network architectures for the generators (with and without magnified learning) used on Oxford-102 and CUB-200.

Generator	Generator with Magnified Learning
fcBN($512 \times 4 \times 4$) NNupsample(8) convBN(512, 3, 1)	fcBN($512 \times 4 \times 4$) NNupsample(8) convBN(512, 3, 1)
NNupsample(16) convBN(256, 3, 1)	NNupsample(16) convBN(256, 3, 1)
NNupsample(32) convBN(128, 3, 1)	NNupsample(32) convBN(128, 3, 1)
NNupsample(64) convBN(64, 3, 1) convBN(3, 3, 1)	NNupsample(64 convBN(64, 3, 1) convBN(64, 3, 1)
	NNupsample(128) convBN(32, 3, 1) conv(3, 3, 1)

C. More generated samples

1) Wikiart: More generated fine-art paintings are visualized in Figure 12, Figure 13, and Figure 14 at high resolution (128×128 pixels).

2) CIFAR-10: Figure 15 shows generated images at 64×64 resolution trained on CIFAR-10.

3) STL-10: Figure 16 shows generated images at resolution of 128×128 pixels trained on STL-10.

4) CUB-200 birds: Figure 17 shows more generated CUB-200 images. Each sample represents one of the 200 bird species.

5) Oxford-102 flowers: Figure 18 shows more generated flower images on Oxford-102 at high resolution (128×128 pixels). Each sample represents one flowers species, with a total of 102 types of flowers generated.

Fig. 12: Generated *genres* images at $\mathbf{128} \times \mathbf{128}$ pixels. From top to bottom: (1) Abstract painting, (2) Cityscape, (3) Genre painting, (4) Illustration, (5) Landscape, (6) Nude painting, (7) Portrait, (8) Religious painting, (9) Sketch and study, (10) Still life.

Fig. 13: Generated *artists* images at $\mathbf{128} \times \mathbf{128}$ pixels. (**Left**) From top to bottom: (1) Albrecht Durer, (2) Boris Kustodiev, (3) Camille Pissarro, (4) Childe Hassam , (5) Claude Monet, (6) Edgar Degas, (7) Eugene Boudin, (8) Gustave Dore, (9) Ilya Repin, (10) Ivan Aivazovsky, (11) Ivan Shishkin, (12) John Singer Sargent. (**Right**) From top to bottom: (13) Marc Chagall, (14) Martiros Saryan, (15) Nicholas Roerich, (16) Pablo Picasso, (17) Paul Cezanne, (18) Pierre Auguste Renoir, (19) Pyotr Konchalovsky, (20) Raphael Kirchner, (21) Rembrandt, (22) Salvador Dali, (23) Vincent van Gogh.

Fig. 14: Generated *styles* images at $\mathbf{128} \times \mathbf{128}$ pixels. (**Left**) From top to bottom: (1) Abstract Expressionism, (2) Action painting, (3) Analytical Cubism, (4) Art Nouveau, (5) Baroque, (6) Color Field Painting, (7) Contemporary Realism, (8) Cubism, (9) Early Renaissance. (**Middle**) From top to bottom: (10) Expressionism, (11) Fauvism, (12) High Renaissance, (13) Impressionism, (14) Mannerism Late Renaissance, (15) Minimalism, (16) Naive Art Primitivism, (17) New Realism, (18) Northern Renaissance. (**Right**) From top to bottom: (19) Pointillism, (20) Pop Art, (21) Post Impressionism, (22) Realism, (23) Rococo, (24) Romanticism, (25) Symbolism, (26) Synthetic Cubism, (27) Ukiyo-e.

Fig. 15: Generated CIFAR-10 images at $\mathbf{64} \times \mathbf{64}$ pixels. From top to bottom: (1) Airplane, (2) Automobile, (3) Bird, (4) Cat, (5) Deer, (6) Dog, (7) Frog, (8) Horse, (9) Ship, (10) Truck.

Fig. 16: Generated STL-10 images at $\mathbf{128} \times \mathbf{128}$ pixels. From top to bottom: (1) Airplane, (2) Bird, (3) Car, (4) Cat, (5) Deer, (6) Dog, (7) Horse, (8) Monkey, (9) Ship, (10) Truck.

Fig. 17: More generated images on CUB-200 birds at $\mathbf{128} \times \mathbf{128}$ pixels.

Fig. 18: More generated images on Oxford-102 flowers at $\mathbf{128} \times \mathbf{128}$ pixels.