PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

MAURÍCIO ARMANI LOPES

FULLY-DISENTANGLED TEXT-TO-IMAGE SYNTHESIS

Porto Alegre
2019

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

FULLY-DISENTANGLED TEXT-TO-IMAGE SYNTHESIS

MAURÍCIO ARMANI LOPES

Dissertation submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fullfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Rodrigo Coelho Barros

Porto Alegre
2019

Ficha Catalográfica

L864f Lopes, Maurício Armani

Fully-Disentangled Text-to-Image Synthesis / Maurício Armani Lopes . – 2019.
83.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rodrigo Coelho Barros.

1. Generative Models. 2. GANs. 3. Text-to-image Synthesis. 4. Disentanglement. I. Barros, Rodrigo Coelho. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Salete Maria Sartori CRB-10/1363

Maurício Armani Lopes

Fully-Disentangled Text-to-Image Synthesis

This Dissertation has been submitted in partial fulfillment of the requirements for the degree of Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on August 27, 2019.

COMMITTEE MEMBERS:

Prof. Dr. Paulo Lilles Jorge Drews Junior (FURG)

Prof. Dr. Duncan Dubugras Alcoba Ruiz (PPGCC/PUCRS)

Prof. Dr. Rodrigo Coelho Barros (PPGCC/PUCRS - Advisor)

Dedico este trabalho à minha família.

"Without education, we are in a horrible and deadly danger of taking educated people seriously."
(G.K. Chesterton)

ACKNOWLEDGMENTS

Thanks to Motorola, my advisor and you.

SÍNTESE DE IMAGEM A PARTIR DE TEXTO COM DESENTRELAÇAMENTO COMPLETO

RESUMO

Algoritmos de generalização sofrem vários problemas. Comumente, os algoritmos de aprendizado profundo tendem a se ajustar a uma quantidade limitada de dados, gerando resultados insatisfatórios para dados não vistos. No caso de modelos geradores profundos, esse problema se manifesta de forma diferente. O modelo tende a ignorar nuances da distribuição real e colapsar em atributos mais comuns. Embora em algum grau, esse seja um comportamento desejado, isso pode levar o algoritmo a gerar imagens com diversidade muito reduzida, não explorando todos os atributos fornecidos pela distribuição real. Redes generativas profundas (por exemplo, GANs) não têm custo explícito para incentivar a diversidade durante o treinamento. Além disso, as GANs foram inicialmente projetadas para gerar amostras aleatórias sem controle do usuário. Propomos um método baseado em desentrelaçamento para resolver ambos os problemas de uma só vez para algoritmos de síntese de imagem a partir de texto. Primeiro, forçamos o desentrelaçamento de conceitos não descritos por descrições textuais (por exemplo, o segundo plano). Então, usamos as representações desentrelaçadas para fornecer combinações aleatórias para o gerador. Isso orienta o gerador para um mapeamento mais completo, gerando um aumento na diversidadea partir da mesma quantidade de dados. Com representações desentrelaçadas, o framework também ganha controle sobre a geração para cada conceito. Realizamos vários experimentos e estudos de ablação para validar nossas contribuições em um conjunto de dados de um único objeto. Os resultados mostram melhorias para ambos os objetivos e nenhum efeito colateral para os frameworks tradicionais. Nossa abordagem pode ser facilmente usada em outros frameworks para aumentar o controle, a diversidade e o realismo.

Palavras-Chave: Modelos Geradores, GANs, Síntese de texto à imagem, Desentrelaçamento.

FULLY-DISENTANGLED TEXT-TO-IMAGE SYNTHESIS

ABSTRACT

Generalization algorithms suffer from several problems. Commonly, deep learning algorithms are prone to overfit a limited amount of data, generating unsatisfactory results for unseen data. In the case of deep generative models, this problem manifests itself differently. The model tends to ignore nuances from the real distribution and to collapse into the most common attributes. To some degree, this is a desired behavior, but this can lead the algorithm to generate images with very reduced diversity, not exploring all attributes provided by the real distribution. Deep generative networks (e.g. GANs) have no explicit objective to encourage diversity during training. Furthermore, GANs were initially designed to generate random realistic samples with no control for the user. We propose a disentanglement-base method to tackle both problems at once for text-to-image synthesis frameworks. First, we force the disentanglement of concepts not described by textual descriptions (e.g. background). Then, we use the learned disentangled representations to provide random combinations for the generator. This guides the generator to a more complete mapping, hence increasing diversity using the same amount of data. With disentangled representations, the framework also gains control over synthesis for each concept. We conduct several experiments and ablation studies to validate our contributions on a single-object dataset. Results show improvements for both goals and no side-effects for the traditional text-to-image frameworks. Our approach can easily be used on top of other frameworks to increase control, diversity, and realism.

Keywords: Generative Models, GANs, Text-to-image Synthesis, Disentanglement.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ACRONYMS

GAN – Generative Adversarial Network

CGAN – Conditional Generative Adversarial Network

VAE – Variational Autoencoder

CA – Conditioning Augmentation

GRU – Gated Recurrent Unit

BIGRU – Bidirectional Gated Recurrent Unit

IS – Inceptin Score

FID – Fréchet Inception Distance

CONTENTS

1. INTRODUCTION

Image synthesis has improved significantly in the past years due to the rise of Deep Learning algorithms, more specifically, Generative Adversarial Networks [19] (GANs). The GAN framework combines high-level feature representation of data, provided by deep neural networks, and an adversarial training procedure to close the gap between complex distributions. With the use of GANs, any unknown distributions (e.g. images of birds) can be mapped from a known distribution (e.g. Gaussian) without relying on any additional supervision but a limited amount of samples from the target distribution. Without an explicit mathematical representation, new data indistinguishable from the real can be synthesized.

Initially, GANs were designed to synthesize images from random noise with no control of what should be drawn. To overcome this issue, Mirza et al. [42] suggested the use of conditions to guide generation. The framework adds on top of the original GAN an auxiliary categorical representation of classes or attributes. Following this framework, Reed et al. [49] proposed the use of text descriptions, which became the benchmark framework for a new field on deep learning research: text-to-image synthesis.

Text-to-image synthesis is the task of generating images from natural language. The natural language provides an easy way, for users, to generate or manipulate images based on their will. It is a general constraint that can guide different aspects of an image. With text descriptions, one could guide the synthesis process to have specific objects with specific attributes such as colors, textures, relative position, etc.

Furthermore, natural language is an appropriate and powerful protocol towards automation. It enhances human-computer interaction by facilitating access to computers by unsophisticated computer users, users in hands-busy situations (e.g. car driving), and users with disabilities [38]. Closing the gap between text and image ultimately means to give computers visual knowledge from our most ordinary way of communication.

However, like any machine learning algorithm, GANs undergoes some statistical problems of generalization; ignoring less common features from the real distribution and generating samples with low diversity [61]. Likewise, it is known that GANs are prone to suffer from *mode collapse*, where equal samples are generated from different points of the known distribution. Specifically, in the case of text descriptions, such issues can become more evident. It is safe to say that text description can not provide complete pixel-level information of an image. Thus, for the text-to-image synthesis, much of the content must be "imagined" taking into account the desired distribution. Text-to-image models are likely to learn common properties for these "free" attributes and are not encouraged to explore all features contained in the real data. This makes text descriptions good in guiding synthesis with a great diversity of colors and textures, but weak in providing diversity for shapes, poses, backgrounds or any other feature rarely described by natural language.

Several alternatives have been proposed to tackle these problems and to overcome general text-to-image GAN's limitations: data augmentation for text latent space smoothness [78, 54]; disentanglement [48]; multi-objective discriminators [11]; and many others. This work focuses its efforts on increasing diversity and control via the disentanglement of concepts.

Text-to-image synthesis frameworks commonly receive only two inputs: a text description and a random noise, which stands alone for diversity. The noise stores all mapped variations for a given text description, which can tie together distinct attributes based on frequently seen combinations. Disentanglement aims to decouple a specific attribute from the noise, which allows the use of non-frequent combinations to increase diversity. For instance, in the case of images of birds, where a body shape would be taken to easily fit the most common color-shape combination (e.g. small neck and yellow body), different shapes can be generated with the same color (e.g. long neck and yellow body). Moreover, once a concept is disentangled from the random noise, it offers a new controllable feature for the synthesis. If a user decides to make small changes in the bird's shape and keep the other features constant, he/she can simply marginalize the disentangled condition.

General image synthesis has three main objectives to be tackled: *realism*, which is associated to how close the learned distribution is to the real distribution; *control*, related to the capability of the user to choose what will be drawn; and *diversity*, related to the model capacity on exploring the full space of possible images to be synthesized.

This work provides a novel way of disentangling concepts for text-to-image synthesis as a proxy to tackle control and diversity. We found that disentanglement can be achieved by conditioning synthesis on a chosen concept and forcing consistency with extra supervision. This approach not only encourages a more diverse mapping but also provides control, letting the user choose any desired combination of features.

We now define image content as three distinct concepts: object style (colors and textures); object boundaries (shape and position); and object context (background) [56]. Since text descriptions mostly provide style information [49], the proposed method focuses on disentangling shape and background from the traditional GAN noise. We add two consistency modules that enforce shape and background conditioning. The framework is then able to learn mismatched combinations of attributes at training time. Also, the method learns new representations of conditions that, at inference time, can guide synthesis or be completely discarded, preserving the traditional text-to-image GAN feature of generating samples only from natural language and a random noise. The ultimate result is a more complete mapping between text descriptions and images. Now, every text description can be mapped to any combination of style, shape, and background contained in the dataset.

Additionally, we employ text representation from NLP (Natural Language Processing) word-vector models instead of the sentence-level representations used on previous work. This technique provides better generalization for words not contained on the task-specific dataset vocabulary and is more appropriate for our disentanglement method, giving a word-level sen-

sitivity to the model. Furthermore, we argue that the problem of multimodal mapping is addressed more properly by using a raw representation of both modalities (text and image).

To summarize, the contributions of this work are: (i) two modules for shape and background disentanglement on text-to-image GANs as a proxy to increase control and diversity; (ii) an adapted text-to-image synthesis architecture for the use of NLP word vectors that gives word-level sensitivity to the model for a better disentanglement, prevent bad conditioning, and increase general results on the task.

2. BACKGROUND

In this chapter, we will dive into the concepts and modules that compose the base framework used in this work. We begin by introducing the frameworks that preceded the first text-to-image synthesis GAN [49] and end with a related work section to contextualize our contributions in the current research status.

2.1 Generative Models

In Machine Learning, every task can be defined as the process of learning a target function $f : X \rightarrow Y$, or equivalent $P(Y|X)$. Generative and discriminative modeling are two distinct designs of learning algorithms for this purpose. Given data samples X and their labels Y, Mitchell [29] defines a generative model as one capable of describing how to generate random instances of X conditioned on the target attribute Y by learning the distribution of $P(X|Y)$. Classifiers such as Naive Bayes are considered generative models since they estimate $P(X|Y)$ and $P(Y)$ to ultimately learn $P(Y|X)$ for classification. On the other hand, Linear Regression is an example of a discriminative classifier, since it directly estimates $P(Y|X)$.

In the case of Naive Bayes, generative modeling is an intermediate step for classification. As opposed to that, a sub-field of deep learning focuses on the capability of synthesizing data indistinguishable from the original domain: content synthesis or content generation. Leveraging from the success of deep learning on high-level feature extraction, several studies have tried to generate new content on image data. Work like Pixel-RNN [63] and Variational Autoencoder [33] achieved promising results. However, arguably Generative Adversarial Networks (GANs), proposed by Goodfellow [19], opened a new path on content generation and established itself as the state-of-the-art in many related tasks. GANs have been widely used in tasks such as single-image super resolution [35], unsupervised learning [55], image-to-image translation [28], semantic image in-painting [75], and domain adaptation [23].

2.2 Generative Adversarial Networks

Generative adversarial networks are generative models that address the unsupervised learning problem of density estimation without explicitly defining a density function [19]. Instead, GANs tackle the problem by mapping some unknown data distribution p_x to a prior known distribution $p_z \sim \mathcal{N}(\mu, \sigma^2)$. Once the mapping is concluded, the network can generate new data indistinguishable from p_x by sampling from p_z.

The GAN framework comprises two parametric functions whose objectives, though concurrent, cooperate for the final objective. The first is the generator $G_{\theta_g}(\cdot)$, trained to learn the one-way mapping $p_z \rightarrow p_x$. The second is the discriminator $D_{\theta_d}(\cdot)$, a binary classifier

trained to distinguish if its input comes from p_x or p_g, where p_g is the distribution of samples generated by G. Ultimately, the discriminator learns to distinguish between real data from the dataset and fake data coming from the generator (see Figure 2.1).

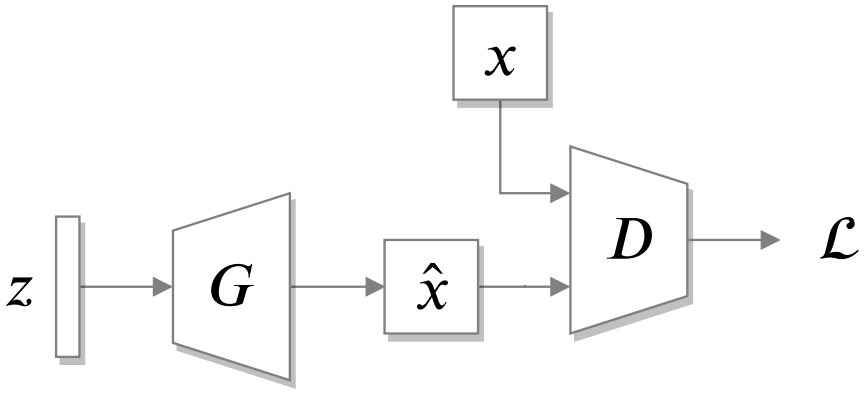


Figure 2.1: Vanilla GAN framework, where x denotes real images from the dataset and $\hat{x}$ denote fake images coming from the generator. Real and fake images are fed into the discriminator one at a time.

The training dynamics is based on the capacity of neural networks to encode high-level features and the backpropagation gradient flow from D to G. While D gets better on its task it is simultaneously providing gradients for G to generate samples closer to the distribution p_x. In practice, both G and D are trained alternately, updating θ_g or θ_d at each step. The final objective is achieved when the generated data $G_{\theta_g}(z) \sim p_g$ becomes indistinguishable from p_x, or simply $p_g = p_x$. The training objective function is formally defined as:

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x}\sim p_x(\mathbf{x})}[\log D_{\theta_d}(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z})))], \tag{2.1}$$

a zero-sum game, where the first term represents the expected classification given an input from p_x and the second the expected classification given an input from p_g.

Considering that D is a binary classifier (outputs 1 for *real* and 0 for *fake*), we want the above expression to have its maximum value when updating θ_d, which means set $D_{\theta_d}(\mathbf{x}) = 1$ and $D_{\theta_d}(G_{\theta_g}(\mathbf{z})) = 0$. Likewise, we want the minimum value by setting $D_{\theta_d}(G_{\theta_g}(\mathbf{z})) = 1$ when updating θ_g. Note that, to update θ_g, the first term does not need to be computed. In practice, for numeric stability, equation. 2.1 is split into:

$$\mathcal{L}_d = \mathbb{E}_{x\sim p_x}[\log(1 - D(x))] + \mathbb{E}_{z\sim p_z}[\log D(G(z))], \tag{2.2}$$

$$\mathcal{L}_g = \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))] \tag{2.3}$$

to be both minimized during optimization. Thus, the labels are modified to $D(\mathbf{x}) = 1$ and $D(G(\mathbf{z})) = 0$ for $\mathcal{L}_d$ and $D(G(\mathbf{z})) = 1$ for $\mathcal{L}_g$. Finally, gradients are computed via backpropagation to update θ_d and θ_g with gradient descent (see algorithm 2.1).

Algorithm 2.1 GAN training procedure [19]

Data: images x, number of training steps S.

for $n = 1 : S$ **do**

Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from prior $p_z \sim \mathcal{N}(\mu, \sigma^2)$.

Sample minibatch of m image samples $\{x^{(1)}, ..., x^{(m)}\}$ from prior p_x.

Update θ_d by descending stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(x)) + \log D(G(z))].$$

Update θ_g by descending stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(z)))].$$

end

In GANs there are no stop criteria. Instead, the iterations are repeated several times to guarantee convergence for the task in hand. The best model is verified by human inspection or by the use of metrics like Inception Score [62] or Fréchet Inception Distance [20]. Both metrics will be discussed in section 4.3.

2.2.1 Conditional Generative Adversarial Networks

Mirza et at. [42] proposed an extended version of the traditional GAN called Conditional GAN (CGAN). Along with the noise vector z, CGAN adds vector c as input to stand for a specific attribute or class (see Figure 2.2). Usually, c is a one-hot vector of size k (number of classes on the dataset). Real and synthesized images will be evaluated against their conditions on the discriminator during training. This procedure ultimately allows G to synthesize images of a specific class among all the others present on the dataset. Hence, the objective functions can be rewritten as:

$$\mathcal{L}_d = \mathbb{E}_{x \sim p_x}[\log(1 - D(x, c)) + \log(D(x, c'))] + \mathbb{E}_{z \sim p_z}[\log D(G(z, c), c)] \tag{2.4}$$

$$\mathcal{L}_g = \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z, c), c))]. \tag{2.5}$$

Note that D also receives c as an argument. In this case, as described in the algorithm 2.2, the discriminator is fed with matched and mismatched pairs of images and conditions, (x, c) and (x, c') respectively. This is because the discriminator D will not only be trained to distinguish between real and fake data but also to evaluate if their input pairs are matching or not. Gradients will now store information regarding the chosen conditions for G.

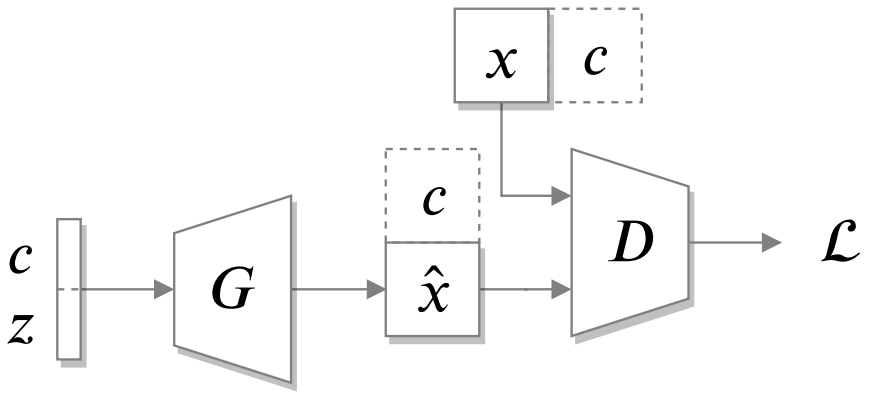


Figure 2.2: CGAN framework.

Algorithm 2.2 CGAN training procedure.

Data: images x, matched conditions c, mismatched conditions c', number of training steps S.

for $n = 1 : S$ **do**

Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from prior $p_z \sim \mathcal{N}(\mu, \sigma^2)$.

Sample minibatch of m image examples $\{x^{(1)}, ..., x^{(m)}\}$ from prior p_x.

Sample minibatch of m matched conditions $\{c^{(1)}, ..., c^{(m)}\}$.

Sample minibatch of m mismatched conditions $\{c'^{(1)}, ..., c'^{(m)}\}$.

Update θ_d by descending stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(x, c)) + \log D(x, c') + \log D(G(z, c), c)].$$

Update θ_g by descending stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(z, c), c))].$$

end

As opposed to the GAN space of fake images (Figure 2.3a), the resulted CGAN space can be seen as a clustering of classes (Figure 2.3b). Each cluster can be achieved by passing the respective condition as input along with the random noise z.

2.3 Text-to-Image Synthesis

Text-to-image synthesis is the task of generating images from a natural language description. The task has two main challenges: to close the heterogeneous and the homogeneous gap between the data. The heterogeneous gap is the gap between high-level concepts of text descriptions and the pixel-level contents of an image, while the homogeneous gap is the gap between both modalities [77] — synthetic and real image distributions —, our final objective.

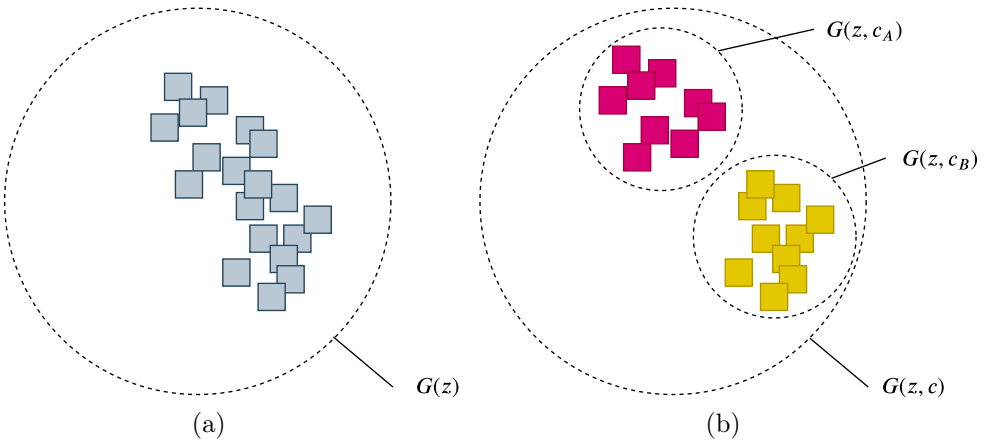


Figure 2.3: Representation of the space of fake images learned by the generator. In (a) we have the resulted mapping of a traditional GAN without categorical conditions, and (b) we have the space learned by the CGAN considering two categories, c_A and c_B.

Reed et al. [49] were the first to propose the use of text descriptions as conditions on top of the CGAN framework to tackle this problem. Differently from classes, text descriptions have high intra-variance; semantically equivalent sentences may have very distinct formatting. Besides, it is not possible to preset a fixed number of conditions. Text descriptions give us an infinite space of conditions varying in size, semantics, and formatting. To deal with such a scenario, Reed added a text encoder $\varphi(\cdot)$ to map text representations to a semantically-meaningful space of sentences (see Figure 2.4).

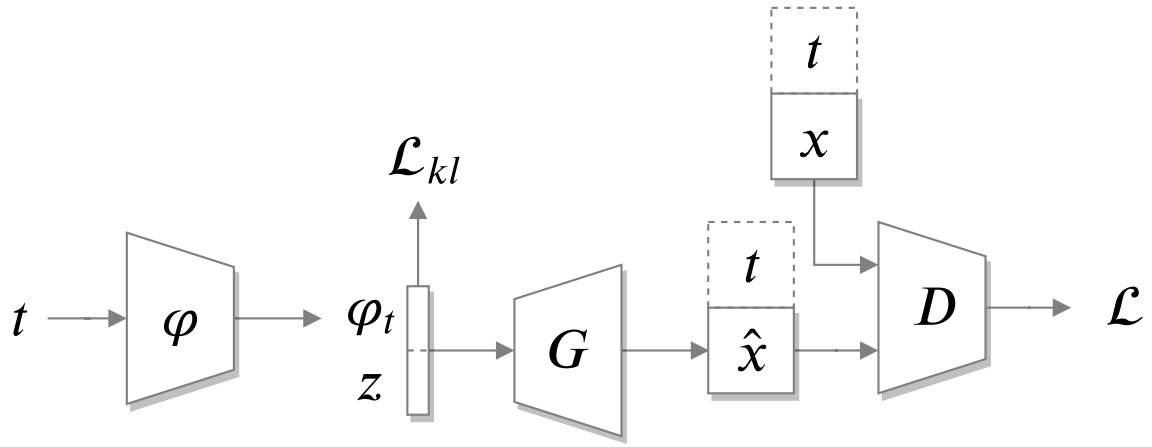


Figure 2.4: Text-to-Image GAN framework. The vector t is a raw text description representation.

This mapping is particularly challenging due to the trade-off between fine word-level control and formatting invariance. For instance, sentences with very similar formatting may describe very different features, while the exact same features can be described in a variety of ways. We want text encoding to be invariant on writing style while preserving fine detailed word-level control over synthesis.

However due to the text representation used by Reed, we believe that text encoders tend to ignore details and prioritize common features during the adversarial training. This scenario may generate a semantically-poor latent space, not exploring all benefits of natural language. Finally, the default training procedure for text-to-image synthesis is summarized in Algorithm 2.3.

Algorithm 2.3 Text-to-image synthesis training procedure.

Data: images x, matched text t, mismatch text t', number of training steps S.

for $n = 1 : S$ **do**

Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from prior $p_z \sim \mathcal{N}(\mu, \sigma^2)$.

Sample minibatch of m image examples $\{x^{(1)}, ..., x^{(m)}\}$ from prior p_x.

Sample minibatch of m matched text $\{t^{(1)}, ..., t^{(m)}\}$.

Sample minibatch of m mismatched text $\{t'^{(1)}, ..., t'^{(m)}\}$.

Update θ_d by ascending stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x, \varphi(t)) + \log(1 - D(x, \varphi(t'))) + \log(1 - D_{\theta_d}(G(\mathbf{z}, \varphi(t)), \varphi(t)))].$$

Update θ_g by descending stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D_{\theta_d}(G(\mathbf{z}, \varphi(t)), \varphi(t)))].$$

end

2.3.1 Text Representation

The first text representation module proposed by Reed was a character-level vector representation learned from the char-CNN-RNN encoder [49]. Previous work show across many different tasks that char-CNN-RNN is one of the most effective textual encoders [50]. Hence, their approach was followed by almost all subsequent work on text-to-image synthesis.

The char-CNN-RNN as it was first proposed for text-to-image synthesis was pre-trained using classes and images from the dataset. The general idea was to train the text encoder to map text representations into a space semantically-distributed in terms of classes. Roughly, this was done by encoding images and their captions to a common space and using both vector representations to predict the correct class (e.g. bird species).

We believe that this method introduces several undesired behaviors. First, since it tries to predict classes using a categorical representation, it is assuming that they are equidistant from each other. Two classes can be closer to each other (e.g. car and bus) than others (e.g. animal species). Second, there is no guarantee that this method is preserving word-level

semantics. All sentences from the same classes are treated as equivalent, which may not be true. Descriptions may have a different level of detailing of the same object, which may cause less frequent attributes to be completely ignored when approximating complete descriptions to poor ones. Lastly, in Reed's approach the GAN only sees sentence-level representations during the adversarial training, which may be harmful for diversity and coherence.

The char-CNN-RNN learns character-level vector representation by optimizing the following cost function:

$$\frac{1}{N}\sum_{n=1}^{N}\Delta(y_n, f_x(x_n)) + \Delta(y_n, f_t(t_n)) \tag{2.6}$$

where $\{(x_n, t_n, y_n) : n = 1, ..., N\}$ is the training data set of images, text, and class labels. The function Δ is the 0-1 loss, which assigns *one* and *zero* for matched and mismatched labels, respectively. The predicted labels are given by two classifiers f_x and f_t parameterized as:

$$f_x(x) = \arg\max_{y\in\gamma} \mathbb{E}_{t\sim T(y)}[\phi(x)^\intercal \varphi_c(t)] \tag{2.7}$$

$$f_t(t) = \arg\max_{y\in\gamma} \mathbb{E}_{x\sim X(y)}[\phi(x)^\intercal \varphi_c(t)] \tag{2.8}$$

where ϕ is a deep convolutional neural network, φ_c is the char-CNN-RNN text encoder, $X(y)$ is the set of images from class y, and $T(y)$ is the set of textual descriptions from class y. What these equations are doing is aligning text and images to have a higher compatibility score when they belong to the same class compared to any other. As already mentioned, the problem with this approach is that it assumes that classes are equally distant from each other, which is not necessarily true. Furthermore, two distinct classes can have very similar descriptions not just because the classes are similar, but because they can lack detailed information. This may lead the encoder to learn extra information that is not present in the textual description. Briefly, this approach can lead to several behaviors that might harm coherence and diversity.

Finally, we argue that the char-CNN-RNN procedure provides extra dataset information that would not be present in a "raw" text description. If we want to learn text-to-image correspondent representations, it is reasonable to argue that text representations should contain only textual properties (grammatical, semantical and syntactical). Also, it should be complete dataset-agnostic and be learned only from a linguistic corpus. We hypothesize that such representation will allow better generalization, preserve fine details and prevent the pointed out issues.

Popular Word Vector Models such as word2vec [40] and Glove [46] provide a vector space of words by parsing large corpora and assigning each word to a point in a semantically-meaningful space. Glove, for example, is a count-based model that learns vectors of words from their co-occurrence information, i.e., how frequently they appear together. Word2vec is

a predictive model that uses a feed-forward neural network to predict a target word from its context (Continuous Bag of Words) or a target context from a source word (Skip-Gram).

We believe that the use of word vectors learned from those methods on text-to-image synthesis is more appropriate for the task. It not only offers a more extensive vocabulary than the one seen in the training set but also truly represents the task meaning of mapping two complete independent modalities (text and image). Moreover, we believe that such representation can be beneficial for our disentanglement approach.

Finally, a crucial difference between char-CNN-RNN and this approach is that while char-CNN-RNN freezes the learned text encoder and only use sentence-level representations, the word-vector model allows the training of a text encoder with word-level sensitivity during the adversarial training. By doing so, we believe that the algorithm will learn a more tuned text encoder and leverage general results for the task.

2.3.2 Conditioning Augmentation

Since the first work on text-to-image synthesis, researchers notice that — due to the very high dimensionality of text latent space and the little amount of data available — the sparsity of the latent space would have to be tackled. At that time, Reed proposed a manifold smoothing by interpolating two sentences before feeding the network. However, it was not an ideal approach since it lost textual consistency by mixing sentences from different classes of objects. Since then, many studies have tackled this problem in a variety of ways. From the use of extra dialog information to textual data augmentation modules [54, 16, 30, 76].

Zhang et al. [78] proposed what was called Conditioning Augmentation (CA). Conditioning augmentation is a module that maps text t to a multivariate normal distribution $\mathcal{N}(0, I)$ before feeding it to the generator and discriminator networks. The intuition is that it could generate a more smooth latent space of sentences by sampling from a normal distribution.

Similar to a Variational Autoencoder (VAE), the CA module encodes t to mean μ and standard deviation σ vectors using a neural network. From it, the module can sample many sentence representations from the same sentence only by adding a random noise as described in equation 2.9:

$$z = \mu(t) + \sigma(t) \odot \epsilon, \tag{2.9}$$

where ϵ is the random noise sampled from $\mathcal{N}(0, I)$. Like VAE, sentences are mapped to $\mathcal{N}(0, I)$ by minimizing

$$KL(t) = \mathcal{D}_{kl}[\mathcal{N}(\mu(t), \sigma(t)^2) \;||\; \mathcal{N}(0, I)], \tag{2.10}$$

which is the KL divergence between the output normal distribution $\mathcal{N}(\mu, \sigma^2)$ and $\mathcal{N}(0, I)$.

GANs were always supposed to map unknown distributions from known distributions and that is exactly what the CA module does by mapping text descriptions to a normal distribution. Until this moment, this is the main approach to tackle text latent space sparsity.

2.3.3 Joint Conditional and Unconditional Distribution Approximation

Another module adopted by many studies in the field is the joint conditional and unconditional discriminator. As mentioned in Section 2.2.1, to handle conditional synthesis, discriminators are required to classify whether a pair of image and condition match. In spite of this being sufficient for the task, it has no explicit penalty for unrealistic images. Zhang et al. [79] handle this problem using the equations from GAN and CGAN jointly. However, instead of using two discriminators, they propose a single discriminator with two branches: one for conditional loss evaluation and other for unconditional loss evaluation (see Figure 2.5).

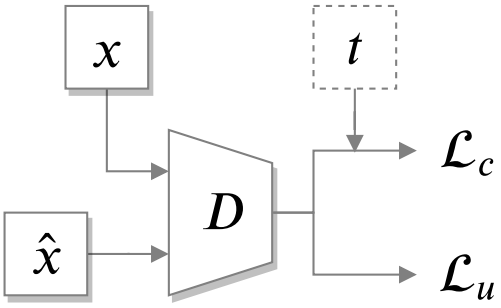


Figure 2.5: Joint Conditional and Unconditional Discriminator.

With the building blocks shown in this section, we cover the basic techniques adopted by the text-to-image synthesis research field.

2.4 Disentangled Representation Learning

In this section, we will define disentanglement and the disentangled representation learning research field. Disentangled representation learning is a recent sub-field in Deep Learning with its own vast literature (e.g. [3, 56, 59, 22, 73, 9, 21, 14, 25]). Disentangled representation learning consists of the technique of breaking down each data feature into defined variables and encoding them as separate dimensions [12]. The goal is to mimic the human perception of data. For instance, humans can easily distinguish different concepts containing in data and imagine the expected result when only one is modified. We can define disentanglement as the degree to which a representation factorizes the underlying factors of variation [18].

For any machine learning algorithms, the first stage of modeling is the feature representation of the observed raw data. It is a crucial stage that can define whether the designed algorithm will achieve its goal or not. Effective representations should highlight important

abstract factors that are relevant for the task while suppressing inconsequential or nuisance factors [34]. Disentangled feature representation is the representation of features with disjoint subsets of abstract concepts that can be manipulated independently and help algorithms towards better performance.

Disentangled representations offer several advantages — (i) Invariance: it is easier to derive representations by simply marginalizing over the corresponding dimensions; (ii) Transferability: they are more suitable for transfer learning, since the subset features can be selectively shared across various tasks; (iii) Interpretability/Inspection: a human expert may be able to interpret dimensions individually and assign specific meaning to them, helping to build trust-in machine learning systems [26]; (iv) Conditioning and intervention: they allow for interpretable conditioning or intervention for the observation of each latent subset effect. In short, the importance of learning disentangled representations has been suggested to be useful for a large variety of tasks and domains [3, 51].

Finally, these representations can also enable controllable generation of data through generative models [26] and be used as a proxy to force diversity.

2.5 Related Work

Besides the already mentioned textual representation issue, text-to-image synthesis as it was first conceived had many problems to be addressed: low resolution, low diversity, low semantic consistency and lack of control. In this section, we summarize text-to-image synthesis efforts to address these problems, as well as some recent work on disentanglement techniques for GANs.

Initially, to increase the original 64×64 low resolution, Zhang et al. [78] proposed a multi-stage generation by stacking two GANs. A first GAN sketches a primitive 64×64 image from text descriptions and a second image-to-image GAN refines and upsamples the input image until it reaches the 256×256 resolution. Addressing the same issue, authors from [80] synthesized 512×512 images introducing the hierarchical-nested adversarial objectives. Their approach can train the entire GAN in a single stage process, allowing mid-level layers to be tuned from gradients of high-resolution layers.

Aiming at semantic consistency, Qiao et al. [47] added a captioning module to generate sentences from the synthesized images and force consistency with the input description. Furthermore, Yin et al. [76] added what they called *Siamese* mechanism in the discriminator to learn consistent high-level semantics from different descriptions. The Siamese mechanism employs a contrastive loss to minimize the distance between fake images from two semantically-equivalent descriptions while maximizing the distance between any other description. This leads to better textual encoding, hence a more consistent generation. Following the StackGAN multi-stage schema, the AttnGAN [72] was the first to explore the benefit of word-level repre-

sentations on text-to-image synthesis. Their framework keeps the sentence-level proposed by Reed [49] as input for the generator but adds word-level features to attend to intermediate image representations and increase detailed refinement.

To increase control over synthesis, we see the first attempt at generating images with position and orientation constraints in [50]. The work introduces a conditional GAN named GAWWN that also receives as input either coordinates of parts of the object (e.g., beak position of a bird or head position of a person) or bounding boxes delimiting the object position. It has several additional steps to encode that information, defining each part of the object in a grid that is passed to the generator to evaluate the network at local (regions of interest) and global levels. In MC-GAN [45], the authors propose a method that modifies a background image to draw objects at specific locations. In their case, the discriminator uses a segmentation mask to determine whether the image is real or fake.

Aiming at diversity, Dash et al. [11] propose the use of an auxiliary classification branch in the discriminator. Their discriminator not only evaluates realism and text matching but also classifies the input image, diversifying generated samples and improving their structural coherence. In FusedGAN [5] the authors combine conditional and unconditional adversarial training to create a prior structural representation to be post-conditioned on textual descriptions. By doing so, they manage to implicitly disentangle structure/shape from style without requiring additional supervision. With this, their method is capable of generating samples that have a very similar structure but diverse style conditions. This approach, however, offers no control over the object structure. A manual search is required to sample the desired shape.

Following previous work on disentangled representation learning [74, 9], Singh et al. [56] use information theory and adversarial training to learn separate code representations for background, shape and style on single-object datasets. To do so, they impose two constraints to separate shape and style: (i) the diversity of shapes is smaller than the diversity of style and (ii) for each shape there is a fixed number of possible styles. The background information is learned in parallel to ground-truth background images. The ultimate result is the ability to change each concept independently and to extract these code representations from real images. These learned features are then used to cluster real images based on the three learned concepts.

3. METHOD

3.1 Problem Definition and Proposed Approach

General data synthesis faces three main goals: realism, diversity, and control. Likewise any generalization problem, data synthesis is prone to ignore less-common statistics. Specifically, when dealing with GANs — where the cost function has no explicit target for diversity — models are likely to map the same image from different noise points (*mode collapse*).

Recently, several approaches have been proposed to mitigate mode collapse [57, 79, 1] and it is fair to say that the problem has been well-addressed. However, mode collapse prevention alone does not guarantee general diversity. Even state-of-the-art approaches tend to be specialists in generating realistic images of a certain pattern while generating poor results for less common features or even ignoring them.

For text-to-image generation, it translates into generating common attributes not present on a description or even ignoring some parts because they are not normally seen with the rest of the description. Furthermore, specific modifications in the text can trigger a series of alterations in the image for matching a more common feature combination. For instance, a "long beak" description might cause the network to also generate a long neck bird, as they are often seen together. Although this might be a desirable behavior, since the model learns to model how correlated bird features are in the training distribution, there is no explicit way to prevent this from happening when the opposite is desired. It is thus unlikely that the model can properly generate combinations of image features that rarely appear together in the training-set.

This work introduces the idea of *explicitly forcing diversity during training via unmatched conditions* to tackle such a problem. The entire training procedure is done by forcing synthesis to match a specific shape, style and background passed as conditions. However, instead of using the discriminator to evaluate each condition consistency on fake images, we force consistency by penalizing pixel-level divergence.

Therefore, our method adds two modules to the regular text-to-image framework: one for shape consistency and one for background consistency (see Figure 3.1). This technique forces a complete disentanglement of the three concepts (object style, object shape, and background) at a noise level, giving better control and allowing a more diverse mapping during training.

Moreover, we opt not to utilize char-CNN-RNN sentence embeddings and, instead, we make use of word-level vectors that are completely dataset-agnostic. We hypothesize that both techniques (consistency modules and word-level descriptions) are more suitable to be used jointly to leverage disentanglement, hence diversity. In the following sections we detail each of our contributions.

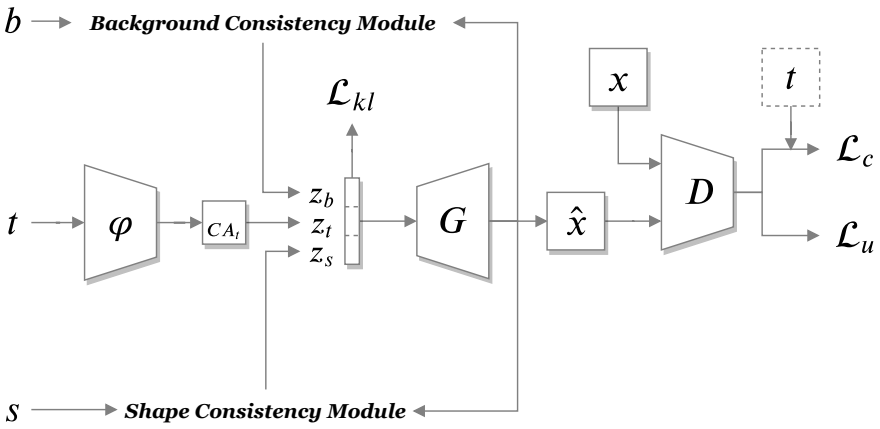


Figure 3.1: Fully disentangled text-to-image architecture. The generator receives a background condition z_b, a text condition z_t, and a shape condition z_s. We use CA modules for each condition to allow intra-diversity.

3.2 Shape Consistency Module

The shape consistency module comprises two additional neural networks, one for shape encoding and other for shape extraction (see Figure 3.2). The shape encoder $\phi(\cdot)$ receives a previously annotated binary matrix $s \in \mathbb{R}^{H \times W}$ that assigns *one* to the object and *zero* to the background of an image $x \in \mathbb{R}^{H \times W \times C}$, where C stands for color channel and $H \times W$ for height and width. The encoded shape representation $d_s \in \mathbb{R}^n$ is fed into a CA module CA_s to generate a final conditional vector representation $z_s \sim \mathcal{N}(0, I)$. The condition z_s is concatenated with other conditions (text and background) to feed the generator G. Once the generation is completed, the synthesized image $\hat{x} \in \mathbb{R}^{H \times W \times C}$ is segmented using the shape extraction network S and the result is evaluated against the input condition.

3.2.1 Shape Extraction Network

The shape extraction network is a pixel-level classifier (segmentation network) that outputs a binary mask $\hat{s} \in \mathbb{R}^{H \times W \times 1}$ from the generated image $\hat{x} \in \mathbb{R}^{H \times W \times C}$, assigning *one* for object pixels and *zero* for the background pixels.

We make use of a well-known segmentation network called U-Net [52], which is a fully convolutional neural network with skip-connections. We use this model to show that our contributions are not highly dependent on sophisticated segmentation strategies. We opted to change the standard loss used in the original paper, replacing it with the dice loss [41]:

$$\mathcal{L}_{seg} = -\frac{2 * \sum [S(\cdot) * y]}{\sum S(\cdot) + \sum y} \tag{3.1}$$

where S is the segmentation network and y is the ground-truth mask. The dice loss was shown to produce sharper edges, which is a useful property given that we force segmentations to be binary during training. We opt to pretrain S and not to update its weights during generation to prevent instability during the adversarial training.

3.2.2 Shape Consistency Loss

To guarantee that the segmentation encoding is properly guiding the network to synthesize objects in the respective shape, we include a loss that accounts for shape consistency:

$$\mathcal{L}_{s_{consist}} = ||s - \hat{s}||^2, \tag{3.2}$$

which is the mean square error between the shape s and the extracted shape $\hat{s}$ from the generated image. Since the ground truth segmentation s is binary we use a threshold of 0.5 to make $\hat{s}$ binary as well. Values above the threshold are set to 1 and values below are set to 0.

It is important to note that, since the CA module force z_s to be in a known distribution $\mathcal{N}(0, I)$, the shape consistency module can be completely discarded at inference time. The user is allowed to generate images from any random noise $z_s \sim \mathcal{N}(0, I)$ that will be automatically mapped to a shape in the learned shape latent space.

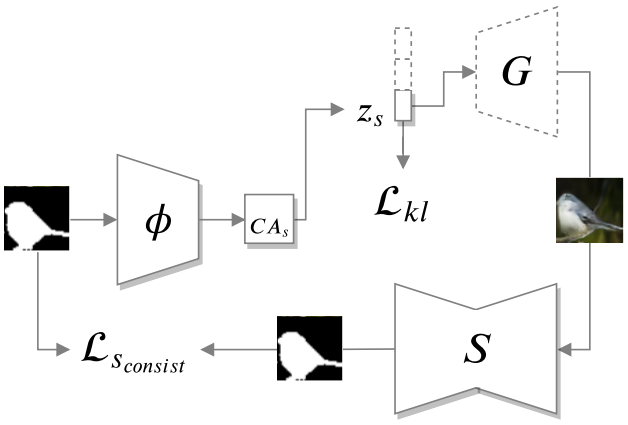


Figure 3.2: Shape Consistency Module.

3.3 Background Consistency Module

In the same way of shape disentanglement, we propose the use of a background consistency module (see Figure 3.3). However, the idea is not to provide a pure background image as a condition but to take advantage of the already used images and learn to extract the background information automatically. Therefore, we encode an image $b \in \mathbb{R}^{H \times W \times C}$ from which we want to extract the background information using an encoder $\psi(\cdot)$. The encoder outputs a

background encoded vector $d_b \in \mathbb{R}^n$ to be fed into a CA module CA_b. The CA_b outputs a final background condition vector $z_b \sim \mathcal{N}(0, I)$. Thereafter, a synthetic image $\hat{x}$ is generated from the background condition z_b to be evaluated against the original background image b.

Note that, till this moment there is no incentive for the background encoder to ignore the foreground (e.g. the bird). Simply applying an image-to-image consistency to the background would force the generation of the object itself. That would cause an overlap between the other conditions of style and shape. To avoid such a scenario, we propose a consistency module with a logical conjunction.

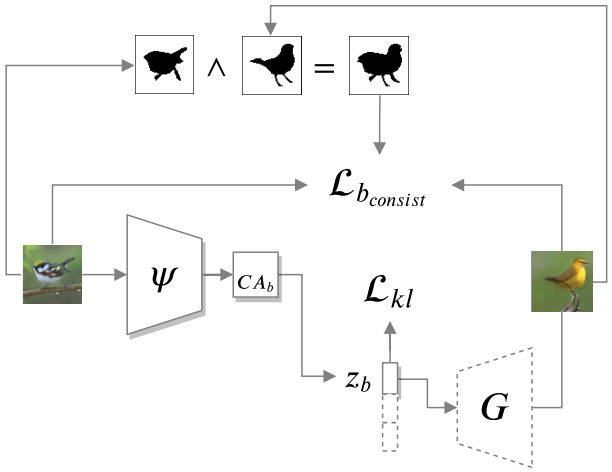


Figure 3.3: Background Consistency Module.

3.3.1 Background Consistency with Logical Conjunction

For the background consistency module to account only for background information, we propose the use of $L1$ error between background pixels. We guarantee consistency only between background information employing a pixel-level logical conjunction operation on both b (background image) and $\hat{x}$ (generated image). The pixel-level logical conjunction operation is exemplified in Figure 3.4. The background cost function is finally given by the equation 3.3.

$$\mathcal{L}_{b_{consist}} = |b \odot u - \hat{x} \odot u|, \tag{3.3}$$

where s_b is the segmentation of the image b, $\hat{s}$ is the segmentation of the generated image $\hat{x}$ and

$$u = g(s_b) \wedge g(\hat{s}) \tag{3.4}$$

is the background mask. The function $g(x) = (x-1)\cdot(-1)$ only inverts the segmentation mask to account for background instead of object pixels.

To avoid redundant computation, we use only mini-batch information in such a way that the segmentation $\hat{s}$ used for shape consistency is reused for background consistency. Once

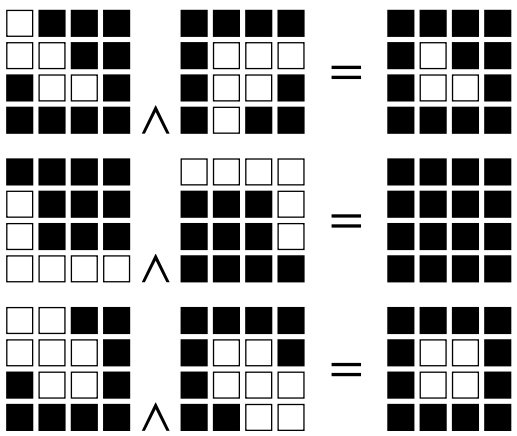


Figure 3.4: Pixel-level logical conjunction.

the encoder ψ perceives that it has to ignore object regions, it starts to encode only background information to minimize Equation 3.3. As it will be shown, the encoder ψ learns to distinguish between background pixels and foreground pixels and not simply learns to encode regions where the objects appear less often. This is an interesting feature and can be employed to different tasks to guarantee an encoding guided by high-level features and not only by regions. Finally, the background encoder also encodes a background condition $z_b \sim \mathcal{N}(0, I)$ that can be randomly sampled from the learned background latent space.

3.4 Domain-Agnostic Text Representation

As described in Section 2.3.1, traditional text representation approaches for text-to-image synthesis use a pretrained text encoder and let the entire adversarial training run only with sentence-level information. Such an approach makes the algorithm fully dependent on the capacity of Equations 2.6, 2.7, and 2.8 to generate semantically-meaningful space of sentences. Furthermore, those equations do not encourage the space to learn intra-class information. We believe that this method results in a text representation that might limit word-level changes during synthesis and harm diversity. To avoid such undesired effects we took advantage of a method proposed by Nam et al. [43] that uses a word-level discriminator with language priors from semantic word embeddings. Such an approach does not consider any previous dataset information to train word vectors. Instead, it uses a well-known method to train word vectors on large corpora: fastText [6].

3.4.1 FastText

FastText is a well-known classifier used for text classification and representation learning. To learn the semantic space of words, fastText parses an entire text corpus. The network is trained to predict context words from a center word with fixed window size. For instance, in

the sentence “this is a brown bird”, the input would be “a”, whereas the output would be “this”, “is”, “brown”, and “bird”, assuming a window size of 5.

FastText uses n-gram of characters to represent words. For instance, taking the word “artificial” with n=3, the fastText representation would be <ar, art, rti, tif, ifi, fic, ici, ial, al> - angular brackets indicate the beginning and the end of the word. This approach can provide vector representation for words that are not in the model dictionary and allows the embeddings to understand suffixes and prefixes. We use word vector learned with this method parsing the entire English Wikipedia corpora.

3.4.2 Text Encoder

The final sentence representation provided by the use of language priors is denoted as $t \in \mathbb{R}^{l_{max} \times d_t}$, where l_{max} is a fixed max sentence length expected by our network and d_t is the word embedding dimension given by fastText. The text encoder φ is a bidirectional GRU with mean reduction along with the temporal steps, resulting in a final sentence vector $\varphi_t \in \mathbb{R}^{n_t}$. Similar to previous steps on shape and background modules, we adopt the CA method for a smooth text representation. The final text condition is then given by $z_t \sim \mathcal{N}(0, I)$. Finally, z_t can be concatenated with the previous z_b and z_s conditions to get the final disentangled noise.

3.4.3 Text-Adaptive Discriminator

The text-adaptive discriminator was proposed by Nam et al. [43] for image manipulation from text descriptions. For such a task, it was crucial to provide training signals related to specific words. We follow the same approach to give word-level sensitivity to our network and increase coherence between words and visual attributes. In the text-adaptive discriminator, each word is independently evaluated against visual attributes to provide fine-grained feedback to the generator.

The procedure goes as follows. In the textual side, the discriminator encodes the sentence t using its own BiGRU. Recurrent networks provide T hidden layers for T sized sentences. In the case of BiGRU, which has forward and backward steps, the network provides two hidden layers for each i-th word that is averaged to get the final word feature representation w_i. For the visual attribute side, the image x is encoded with multiple convolutional steps. For each step j an intermediate feature is taken and global average pooling is applied to get a 1D image vector v_j. Then, for each w_i, a 1D sigmoid local discriminator f_{w_i} is created to determine whether a visual attribute v_j is related to a word w_i. Formally, f_{w_i} is described as:

$$f_{w_i}(v) = \sigma(W(w_i) \cdot v + b(w_i)) \tag{3.5}$$

where $W(w_i)$ and $b(w_i)$ are the weight and the bias dependent on w_i. The fucntion σ is the sigmoid.

The final classification decision is made with the aid of a word-level attention module. However, differently from [72], the method does not use explicit spatial attention on images. It simply learns to weight each classification score - associated with a word - by attending words based on their own. This method reduces the impact of less important words and highlight key words important to specific visual attributes. The score weight is computed by a softmax layer written as:

$$\alpha_i = \frac{\exp\left(u^\intercal w_i\right)}{\sum_i \exp\left(u^\intercal w_i\right)}, \tag{3.6}$$

where u is the temporal average of words w_i. The final weight α_i is then applied to the final classification score for each word. Finally, the word-level text adaptive-discriminator can be written as:

$$D_w(x,t) = \prod_{i=1}^{T} [\sum_j \beta_{ij} f_{w_i}(v_j)]^{\alpha_i} \tag{3.7}$$

where β_{ij} is a softmax weight that determines the importance of the visual attribute from the convolutional layer j for each word w_i. It is important to note that this approach does not change the adversarial training objective. It only replaces the traditional classifier D with this multi-scale text-adaptive classifier D_w.

3.5 Final Objective

Due to train instability, Nam et al. [43] removed the conditional discriminator for synthetic images and pointed out that their resulting objective was enough to produce real and coherent images. We follow the same approach and remove this objective from the traditional conditional adversarial loss of Eq. 2.4. Finally, the final objective function of our method can be rewritten as:

$$\begin{aligned} \mathcal{L}_d &= \mathbb{E}_{x,t\sim p_x,p_t}[\log\left(1 - D_w(x)\right) + \lambda_d(\log\left(1 - D_w(x,t)\right) + \log D_w(x,t'))] \\ &+ \mathbb{E}_{t,s,b\sim p_t,p_s,p_x}[\log D_w(G(t,s',b''))] \end{aligned} \tag{3.8}$$

for the discriminator, where s is the segmentation mask and b is an image from p_x passed as a background condition for the generator. Note that the discriminator never sees segmentation or the background data, which simplifies the overall loss function. Shape and background information are completely contained in the consistency modules. Finally, for the generator,

the final loss is defined as:

$$\begin{aligned}\mathcal{L}_g = \mathbb{E}_{t,s,b\sim p_t,p_s,p_x}[\log D_w(1 - G(t, s', b'')) + \lambda_g \log(1 - D_w(G(t, s', b''), t'))] \\ +\mathcal{L}_{kl} + \gamma_s \mathcal{L}_{s_{consist}} + \gamma_b \mathcal{L}_{b_{consist}},\end{aligned} \tag{3.9}$$

where

$$\mathcal{L}_{kl} = \lambda_t KL(\varphi(t)) + \lambda_s KL(\phi(s)) + \lambda_b KL(\psi(b)). \tag{3.10}$$

Note that text, segmentation mask, and background image are sampled independently, which means that they most likely do not match the same example or class. This is an important aspect of our method, ensuring that the generator is capable of synthesizing samples with respect to the textual description but with a wide variety of shapes and backgrounds.

4. EXPERIMENTAL SETUP

We conducted our experiments to evaluate the three main objectives of deep generative models: control, diversity, and realism. All objectives were quantitatively evaluated using metrics proposed by the research community as well as self-developed heuristics to evaluate the particularities of our method. For better understanding and to provide support to our findings we show several qualitative results for each experiment. All experiments were conducted in the CUB dataset [65].

This chapter is divided into three sections. First, we will detail the dataset used in our experiments. Second, we will detail all the implementation parameters used. Third, an explanation of all metrics adopted for evaluation.

4.1 Dataset and Preprocessing

All of our experiments were conducted in the Caltech-UCSD Birds-200-2011 dataset. The CUB dataset has a total of 11,788 images of 200 classes (bird species) extracted from Wikipedia articles and filtered by users of Mechanical Turk [71]. The classes are equally distributed with approximately 60 images per class. Each image is annotated with 1 bounding box, 15 part locations, 312 attribute labels, 1 segmentation masks, and 10 textual descriptions.

Since images on the CUB dataset have different sizes and aspect ratios, we follow the same preprocessing adopted by [79] and crop all images and segmentations based on their bounding boxes locations. This guarantees a fixed size and aspect ratio for the neural network processing and also provide a better framing for the task. The holdout split is divided into 150 classes for training and 50 classes for test to guarantee synthetic images only based on unseen conditions.

4.2 Training and Architecture Details

We made the code available at Github [1]. The Adam [32] optimizer is used. We first train the segmentation network for 600 epochs with a learning rate of 0.0001. We freeze the segmentation network and proceed to the GAN training. The discriminator and the generator are trained for 600 epochs with a learning rate of 0.0002 with a decay by half every 100 epochs. We did not perform exhaustive hyper-parameter tuning. We use random crop and horizontal flip as data augmentation. All images are $64 \times 64 \times 3$ tensors ($H = 64$, $W = 64$, $C = 3$). The mini-batch used was 64. All conditions are down-sampled to $n = 1024$ before the CA network, where they are all down-sampled to 128 (z_t, z_s and z_b). We follow TAGAN [43]

[1] https://github.com/mauricioarmani/text-to-image-synthesis-with-shape-background-disentanglement

and use $l_{max} = 50$ (max sentence length) and $d_t = 300$ (word-vector size). The recurrent network φ used to encode the sentences is a BiGRU [10] with $n_t = 512$ (hidden layer). The hyperparameters used were $\lambda_t = 0.5$, $\lambda_s = 0.5$, $\lambda_b = 0.5$, $\gamma_s = 10$, $\gamma_b = 10$, $\lambda_g = 10$ and $\lambda_d = 10$.

The Generator G comprises residual blocks with 3x3-conv layers, Batchnorm [27] and ReLU activation. After every residual block, the image is up-sampled with nearest-neighbor interpolation. We use the discriminator proposed in [43], which evaluates sentences at word-level and comprises several down-sampling operations with 3x3-Conv layers, batch-normalization, and Leaky ReLU activations. Our segmentation network is the U-Net [52].

4.3 Metrics

Despite the improvements in the image synthesis field, the absence of meaningful evaluation metrics becomes challenging to make progress towards improved models. As a result, the generative modeling community has developed various ad-hoc evaluation criteria [2]. Recent work have introduced new metrics [19, 44, 48, 80]. However, not all of these metrics are useful for evaluating GANs [60]. Notably, metrics as Inception Score (IS) [53] and Fréchet Inception Distance (FID) [20] became the most common metrics to evaluate realism and diversity of generative models.

To evaluate all different aspects of our method, we could not rely solely on traditional GAN evaluation metrics. Since our main objective is to provide a method for control and diversity via disentanglement of concepts, we use several metrics to evaluate consistency (control), as well as diversity and realism. We also propose a metric to evaluate the overlap of the disentangled concepts.

4.3.1 Inception Score

A well-performing approach to evaluate GANs is the Inception Score (IS), which has a good correlation with human judgment [53]. IS uses the well known Inception-ResNet [58] image classifier to evaluate the distribution of generated classes. To account for diversity, the assumption is that a diverse model should provide samples equally distributed among the classes in the dataset. In other words, the prior distribution $p(y)$ should be the closest as possible of a homogeneous distribution. To account for realism, the assumption is that the conditional probability $p(y|x)$ should be highly predictable. Given an image x and a specific class y_i, $p(y|x)$ should be closest as possible of 100% for $y = y_i$ and 0% for any other class or $y \neq y_i$. Finally, this means that we want $p(y|x)$ and $p(y)$ to be highly divergent. Thus, IS is given as a single scalar value by computing the KL-Divergence between $p(y)$ and $p(y|x)$.

To properly calculate IS, we construct an estimator of the marginal class distribution $\hat{p}(y)$ by sampling fake images,

$$\hat{p}(y) = \frac{1}{N}\sum_{i=1}^{N} p(y|x^{(i)}), \tag{4.1}$$

where N is the number of sample images taken from the model and $p(y|x^{(i)})$ the softmax layer modeled by a Inception-Resnet pretrained on the ImageNet dataset [13] and fine-tuned for the 50 classes of the CUB test-set. Then, KL-divergence can be computed by

$$IS(G) = \exp(\frac{1}{N}\sum_{i=1}^{N} \mathcal{D}_{kl}[p(y|x^{(i)}) \;||\; \hat{p}(y))]). \tag{4.2}$$

It is recommended applying the above estimator 10 times with $N = 5000$ and then taking the mean and standard deviation of the resulting scores [2]. For our experiments we use $N = 5000$, being approximately 100 samples from each of the 50 classes from the test-set. We can rewrite IS using the $\mathcal{D}_{kl}$ formulation for discrete probability distribution as:

$$IS(G) = \exp(-\frac{1}{N}\sum_{i=1}^{N} p(y|x^{(i)})(\log p(y|x^{(i)}) - \log \hat{p}(y)). \tag{4.3}$$

where $p(y|x^{(i)}) \in \mathbb{R}^{1\times C}$ - being C the number of classes -, and $\hat{p}(y) \in \mathbb{R}^{1\times C}$. Supposing the perfect model, $p(y|x^{(i)})$ would be a one-hot vector $\forall i$, hence $\hat{p}(y)$ would be a vector of $1/C$ for each class (uniform distribution).

4.3.2 Fréchet Inception Distance

Similarly to IS, Fréchet Inception Distance [20] (FID) relies on the use of the Inception-Resnet classifier to compute model statistics. However, instead of computing the classification distribution over classes, FID extract image features from an intermediate layer and compare results from real and fake data.

The coding units are assumed to follow a multivariate Gaussian distribution with mean μ and covariance Σ. The difference between real and fake data is measured by the Fréchet distance [17] also known as Wasserstein-2 distance [64].

The Fréchet distance between the Gaussian $\mathcal{N}(\mu_r, \Sigma_r)$ obtained from real images and the Gaussian $\mathcal{N}(\mu_g, \Sigma_g)$ obtained from fake images is computed as follows:

$$FID(G) = \|\mu_r - \mu_g\| + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}), \tag{4.4}$$

where $X_r \sim \mathcal{N}(\mu_r, \Sigma_r)$ and $X_g \sim \mathcal{N}(\mu_g, \Sigma_g)$ are the 2048-dimensional activations of the Inception-Resnet pool3 layer for real and generated samples respectively. X_g comprises a total of 30000 images while X_r is computed from the entire training-set (8835 images). Thus, μ and Σ are all 2048-dimension vectors.

4.3.3 Multi-Scale Structural Similarity

The Multi-Scale Structural Similarity (MS-SSIM) has been widely used as a proxy to evaluate diversity on general image synthesis tasks [37, 66, 11, 80, 67, 44].

The single-scale SSIM measure [67] computes the similarity between two images, here denoted as x_1 and x_2. Differently from mean squared error (MSE), SSIM tries to emulate the human perception of an image discounting aspects considered not important for human perception [7]. The main problem with metrics like MSE is that the evaluation is made at pixel-level. Humans tend to see images more holistically, not evaluating the pixel-wise differences but capturing the overall relation between them. To do so, SSIM computes three quantities: luminance (I), contrast (C), and structure (S), given by:

$$I(x_1, x_2) = \frac{2\mu_1\mu_2 + C_1}{\mu_1^2 + \mu_2^2 + C_1} \quad C(x_1, x_2) = \frac{2\sigma_1\sigma_2 + C_2}{\sigma_1^2 + \sigma_2^2 + C_2} \quad S(x_1, x_2) = \frac{\sigma_{12} + C_3}{\sigma_1\sigma_2 + C_3} \tag{4.5}$$

where μ_1, μ_2, σ_1 and σ_2 are the mean and standard deviations of pixel intensity in a local image patch of either image x_1 or x_2. The variable σ_{12} denotes the correlation coefficient between corresponding patches of x_1 and x_2. C_1, C_2 and C_3 are only constants added for numerical stability. The three quantities are combined to compute the SSIM score as follows:

$$\text{SSIM}(x_1, x_2) = I(x_1, x_2)^\alpha C(x_1, x_2)^\beta S(x_1, x_2)^\gamma \tag{4.6}$$

where α, β and γ are parameters used to adjust the relative importance of the three components. We follow [80] and set $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$. This results in a specific form of SSIM:

$$\text{SSIM}(x_1, x_2) = \frac{(2\mu_{x_1}\mu_{x_2} + C_1)(2\sigma_{x_1x_2} + C_2)}{(\mu_{x_1}^2 + \mu_{x_2}^2 + C_1)(\sigma_{x_1}^2 + \sigma_{x_2}^2 + C_2)}. \tag{4.7}$$

Finally, the MS-SSIM score is a variant of SSIM for multiple scales. Both images are iteratively downsampled by a factor of 2. Considering that at each step images are downsampled by a factor of 2^{j-1} till the coarsest scale, M, we can formalize the MS-SSIM as follows:

$$\text{MS-SSIM}(x_1, x_2) = I_M(x_1, x_2)^{\alpha_M} \prod_{j=1}^{M} C_j(x_1, x_2)^{\beta_j} S_j(x_1, x_2)^{\gamma_j} \tag{4.8}$$

Luminosity I is only computed for the smaller scale [67]. All parameters were considered following previous work on text-to-image synthesis.

MS-SSIM and SSIM satisfy the following conditions:

1. Symmetry: $S(x_1, x_2) = S(x_2, x_1)$;
2. Boundedness: $S(x_1, x_2) \leq 1$;

3. Unique maximum: $S(x_1, x_2) = 1$ if and only $x_1 = x_2$;

The final MS-SSIM score of a given image pair is obtained computing the average MS-SSIM for all patches. Finally, to account for diversity, the MS-SSIM is computed for all possible image pairs in a batch of 10000 random samples and averaged [80]. This metric will also be used to evaluate background consistency in Section 5.1.3, however, with only one scale.

4.3.4 Visual-semantic similarity

In order to evaluate consistency between the input text descriptions and the generated images, we adopt the method proposed by Zhang et al. [80], Visual-Semantic (VS) similarity score. VS score trains two functions f_v and f_t to align paired image features and text descriptions into a common space $\mathbb{R}^{512}$. The alignment is obtained by minimizing the following pairwise ranking loss [68, 69, 70]:

$$\begin{aligned} &\sum_v \sum_t \max\{0, \delta - c(f_v(v), f_t(t)) + c(f_v(v), f_t(t'))\} \\ &+ \sum_t \sum_v \max\{0, \delta - c(f_t(t), f_v(v)) + c(f_t(t), f_v(v'))\}, \end{aligned} \tag{4.9}$$

where v is the 2048-dimensional feature vector extracted from Inception-Resnet and t is the word-level text description. The vectors t' and v' denote mismatched text and image features, respectively. The image feature encoder f_v is a fully connected layer that maps $v \in \mathbb{R}^{2048}$ to $\mathbb{R}^{512}$. Originally, the text encoder f_t is a fully connected layer that receives the sentence-vector representation used in traditional text-to-image synthesis framework. Since we are using word-level representations, we had to adapt the method and use a GRU with 512 hidden units followed by a fully connected layer. The final representation is a 512-dimensional vector. Finally, $c(\cdot)$ is the cosine similarity between both representations. The final VS score is computed by averaging the results of 30000 samples.

4.3.5 Intersection Over Union

In this work, the Intersection over Union (IoU) will be used to compute the consistency between input shape condition $s \in \mathbb{R}^{64\times64\times1}$ and the synthesized image shape $\hat{s} \in \mathbb{R}^{64\times64\times1}$. Therefore, we employ the IoU at a pixel-level based on input and output segmentations. The output segmentation is extracted by segmenting the synthesized image in the same way it is done during training. We define the IoU in terms of its confusion matrix as follows:

$$\text{IoU}(\text{s}, \hat{\text{s}}) = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FP_i + FN_i}. \tag{4.10}$$

where TP is the number of pixels where the predicted mask correctly assigns as an object; FP the number os pixels wrongly assigned as an object, and FN the number of pixels wrongly assigned as non-object. N is the total number of text descriptions on the test split, 2953.

To evaluate the segmentation network's accuracy, we calculate the IoU over the 2953 real image samples and their ground truth masks. The calculated IoU was 86.87%. Thus, it is important to point out that our measure has a minimum error of approximately 13%.

4.3.6 Disentanglement Measure

The disentangled representation learning literature has made it clear the importance of quantifying disentanglement and how there are no straightforward solution [21]. The field still lacks a clear metric for quantitative evaluation and most of the work still relies solely on qualitative analysis [18]. Recently, several metrics have been proposed [31, 21, 8, 18] However, we found that most of them rely on classification predictability of specific attributes on synthetic datasets [21, 31, 34, 39, 24] and are strictly developed to evaluate disentanglement for annotated labels.

Therefore, we did not find a metric that could easily be adapted to our scenario of supervised disentanglement of non-categorical labels. To overcome this issue, we developed a simple method to evaluate the overlap between disentangled concepts.

The idea is to evaluate how much a condition (e.g. text description) affects pixel statistics from the foreground and the background on synthetic images. In the case of text descriptions, we expect high changes in the foreground and low changes in the background. Table 4.1 shows the expected result for all three conditions.

Table 4.1: Expected disentanglement score table.

Free	Foreground Changes	Background Changes
Text	**high**	low
Background	low	**high**
Shape	low	low

Scores are computed by varying a single condition while keeping others fixed. For each generated image, we use the segmentation network S to classify whether a pixel belongs to the foreground or the background. Next, we consider foreground and background distributions as Gaussian distributions. Then, we compute the mean and standard deviation for each region (foreground and background). Finally, the foreground change score is computed by taking the average divergence between foreground distributions of all synthesized images. Likewise, the background change score is computed from all generated background distributions.

The overall process goes as follows. First, we take N instances from our test-set, which means N text, shape, and background conditions ($\{(z_t^{(n)}, z_s^{(n)}, z_b^{(n)}) : n = 1, ..., N\}$). Second, we generate N^3 samples, one for every possible input combination. We then evaluate the disentanglement of one of our three conditions for both the foreground and background. Take, for instance, a scenario where we are measuring the text disentanglement with $N = 20$. For each one of the 400 combinations of shape and background ($\{(z_s^{(n)}, z_b^{(n)}) : n = 1, ..., N\}$), we calculate the mean and standard deviation of the foreground and background of each image (20 different z_t for each (z_s, z_b) combination). Then, we compute the mean univariate KL-divergence for each mean and standard deviation, representing the amount of object and background changes for a single (z_s, z_b) combination. We then compute the average for every combination (again, 400) and store them. We execute the procedure M times and report the mean and standard deviation of the multiple executions as the final *disentanglement score*.

5. RESULTS AND DISCUSSIONS

In this chapter, we present the results obtained and their implications. The results are organized as follows: First, we evaluate control by computing consistency measures for shape, text, and background conditions; second, we evaluate diversity individually using the proposed literature metrics; finally, we evaluate the impact on image quality by comparing scores to previous work and evaluating latent space smoothness. Furthermore, we conducted several ablation studies to evaluate each proposed solution as an individual module and it effects. All results are evaluated quantitatively and qualitatively.

5.1 Control

To evaluate how well-conditioned our framework is, we start by computing the consistency between input information and synthesized information. Each concept (shape, text, and background) has its approach to evaluate consistency. Shape consistency can be easily evaluated in a pixel-wise manner using the same segmentation network used during training. However, note that text and background information cannot be simply pixel-wised compared. We do not want the background to be strictly equal to the input condition. That would mean a possible overlap of the background on the object and vice-versa. Text condition obviously cannot be directly compared with the output result. Text input and the synthesized style does not belong to the same modality and depends on a more sophisticated heuristic to evaluate consistency.

5.1.1 Shape Consistency

We use IoU described in Section 4.3.5 to measure the shape consistency. The procedure is simple. We randomly choose 30000 input triplets (t, s, b) from the test-set. Note that each condition is individually sampled, so the combination is more likely not to be aligned. We generate 30000 images from those inputs. Next, we use the segmentation network S used during training to extract the segmentation mask $\hat{s}$. Then, we compute the IoU between fake segmentations $\hat{s}$ and real segmentation s. The result obtained is an average of 77.29%. Remembering that the segmentation network already has an error of 86.87%, so we can compute the final error by taking the difference and subtracting from 100%. The final shape consistency score is 90.42%.

We can see qualitatively in Figure 5.1 that the model easily follows the input shape. For each row, background and text conditions are fixed and only the shape conditions are being modified. It was not only expected but also desired that the shape was not hard-conditioned.

That could harm fine-grained style details, such as "curved beak" and "long feathers", and even cause training instability.

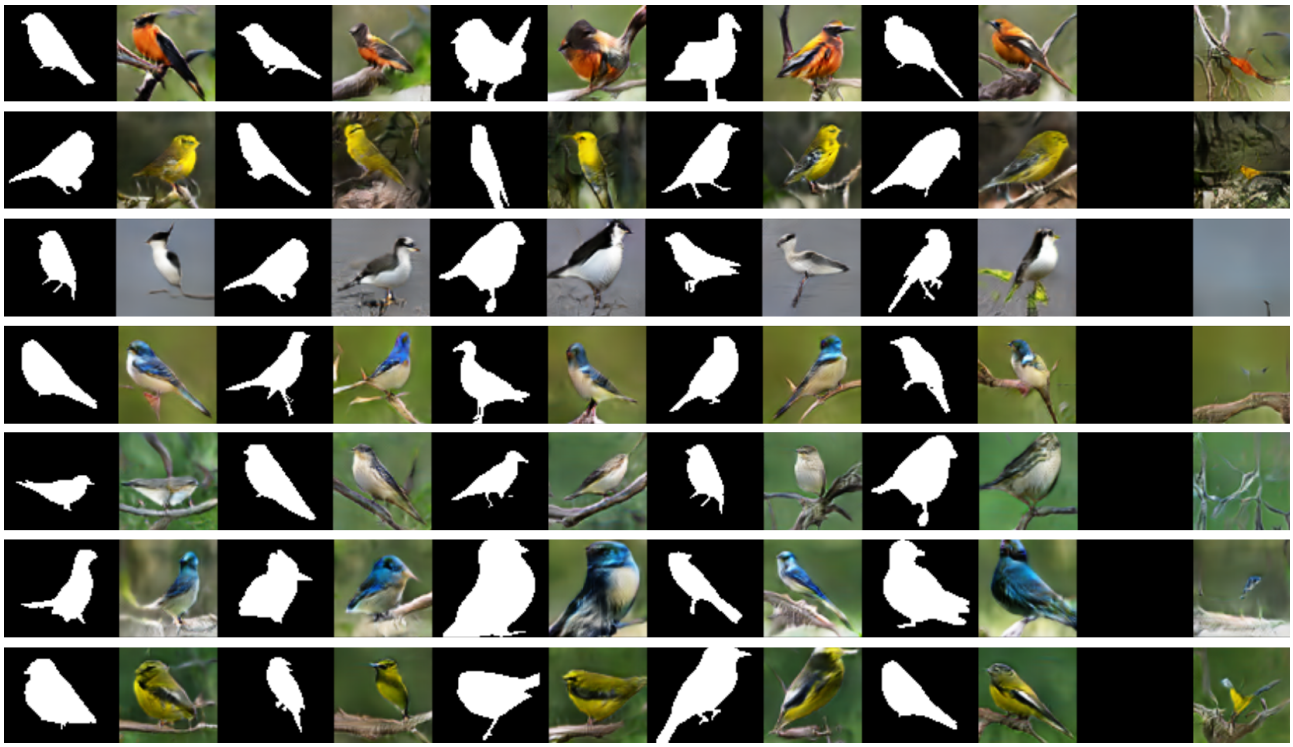


Figure 5.1: Qualitative results for shape consistency. For each row, background and text description are fixed. Better seen in color.

Note that, in the absence of object pixels (last column), the model synthesizes a background image and completely ignores the textual information. This highlights two desired features of our model: good shape/background disentanglement and the learning of a mapping between shape and text. In other words, the model understands that style information is only relevant for the object, which also shows a good disentanglement between style and background. We can see that in some cases the model barely tries to draw a bird with the style condition.

5.1.2 Text Consistency

Evaluating text consistency is not a trivial task due to the multimodal nature of this mapping. Most of the related approaches do not evaluate text consistency quantitatively. We decide to use the VS-similarity score described in Section 4.3.4. Since our framework does not use the traditional text representation of previous work, we cannot directly compare results with previous work. That said, we decide to evaluate only the effect of the consistency modules against a baseline model. The baseline receives only the text condition t and the noise vector z - which stands alone for diversity - and does not have any consistency module as ours. However, the text representation is the same used in our framework (word-vectors from fastText). This

way, we can compare the impact of shape and background consistency modules on the VS-similarity. The results can be seen in Table 5.1.

Table 5.1: Visual-semantic similarity for our model and the baseline. The "Aligned" column refers to alignment regarding the text condition.

Model	Segmentation	Aligned	Background	Aligned	VS
Ours	$z_s \sim \phi(x \sim X)$	Yes	$z_b \sim \psi(x \sim X)$	No	**0.146** $\pm$ 0.183
Baseline	-	-	-	-	0.140 $\pm$ 0.184

Note that, since our model receives a shape condition that may contain conflicting information regarding the text, we compute the VS guaranteeing alignment between shape and text description. The text and shape condition conflict will better be discussed in Section 5.1.5 Here, the background condition is chosen randomly.

As an additional verification, we decide to compute VS changing the input configuration. Table 5.2 shows the impact of aligning shape, aligning background, aligning shape and background and with no alignment at all. It is important to point out that all conditions come from real samples and are not being sampled from the ussian distribution.

Table 5.2: Visual-semantic similarity for our model with different input configurations.

Segmentation	Aligned	Background	Aligned	VS
$z_s \sim \phi(x \sim X)$	No	$z_b \sim \psi(x \sim X)$	No	0.135 $\pm$ 0.183
$z_s \sim \phi(x \sim X)$	Yes	$z_b \sim \psi(x \sim X)$	No	0.146 $\pm$ 0.183
$z_s \sim \phi(x \sim X)$	No	$z_b \sim \psi(x \sim X)$	Yes	0.166 $\pm$ 0.184
$z_s \sim \phi(x \sim X)$	Yes	$z_b \sim \psi(x \sim X)$	Yes	**0.177** $\pm$ 0.184

Note that, by aligning the background, the similarity increases significantly. We believe that the image encoder, used to compute VS, may have learned to consider background information to increase its performance. This is a reasonable assumption since birds species are closely related to their habitat. For instance, yellow-colored birds are never seen in a water background but are commonly seen in a more greenish context. On the other hand, the model with no alignment may have several samples with conflicting information. Thus, we believe only the model with shape alignment holds a fair comparison with the baseline.

We show some qualitative results for text consistency in Figure 5.2. All samples were generated by randomly choosing the three conditions. The results are quite impressive. The network can model fine details such as "*pointy bill*" and "*white wing bars*". Moreover, the model does not only consider the main features but the whole description - even when the description is long and detailed. We can see some examples of rare features such as "*pink color*" and "*red shade around eyes*" being modeled. This kind of generalization is only achievable by the use of word-vector models. Word-vector models have a meaningful latent space of colors. For example, colors such as pink, magenta, and purple tend to be close in the semantic space learned by fastText. Thus, colors that are rarely seen during training (e.g. magenta is seen only one time) can be drawn. In Figure 5.3 we can see an example of the color magenta being modeled by our framework against the StackGAN result.

This bird appears quite average in length and side, it has a white breast and throat while it's wings and crown are brown, it also has an orange beak.		*A tiny grey bird, with a lot of head, small, black, pointy bill, yellow crown, black cheek patch, white superciliary, malar stripe, breast, sides, and flanks, and yellow tipped coverts.*		*The head of the bird is black while the body of the bird is golden yellow.*	
A bird with a white belly, gray-green head, and short black and pink beak.		*This medium bird has a long neck, grey and white countershading, and a set of dark feathers atop its crown with a narrow pointed orange bill.*		*This bird has a white belly and head with a pointy black beak.*	
This colorful specimen has a rust colored belly, crown, and nape accented with blue primaries and secondaries, and tail.		*A small bird with a black head and black nape, with black covering the rest of its body, and bill is short and pointed.*		*An entirely black bird with a small, sharp beak and small black eyes.*	
The bird has black crown and nape, gray throat, breast, belly and abdomen, the tarsus and feet are light brown.		*The bird has a white belly and a yellow beak with a brown body.*		*A mostly yellow bird with brown feathers in its wings and head.*	
This dark colored bird has a brown belly and breast and a brown bill.		*Medium sized bird with a matte gray body, pale yellow tipped tail feather and wing edges, red shade around eyes, and small red spot on wings.*		*A small brightly colored red bird, with a white speckled belly.*	
A bird with a white breast, short yellow bill and gray feet.		*A yellow breasted and yellow crowned bird, with a black throat and black primaries with white wingbars.*		*A small bird with brown feathers and short but sharp beak.*	
The bird has a white throat, a spiked crown and a long bill.		*This birds head is small compared to its body, its body is brown, and its foot and bill are black.*		*A colorful bird with a bright yellow body, a black crown and throat, orange bill, and black primaries and secondaries.*	

Figure 5.2: Generated samples. Each bird was generated from the sentence on its left and random shape and background conditions. Better seen in color.

Figure 5.3: Depiction of magenta coloured samples. The upper example is the real image being described. Our model is capable of reproducing a rare feature (magenta) while StackGAN [79] converges to a common bird (brown-colored). Better seen in color.

5.1.3 Background Consistency

To compute background consistency, we decide to use the SSIM score defined in Section 4.3. Since we are only interested in evaluating background, we follow the same approach of pixel-wise conjunction used during training. We compute the SSIM using the resulted mask.

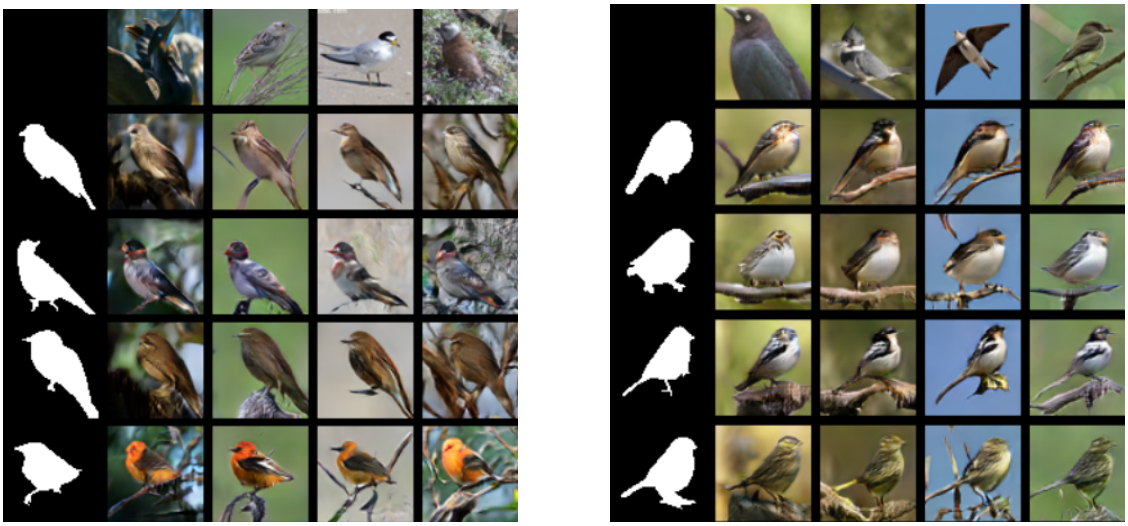
As described earlier, SSIM computes mean and standard deviation in patches (usually 11×11 patches). Computationally, this is done via 2D convolutions in the image with a 11×11 gaussian filter. In order to evaluate only background pixels, we have to exclude all patches with at least one foreground pixel. Then, we could not compute the SSIM for smaller scales (MS-SSIM) - almost all patches had at least 1 foreground pixel for 32×32 or smaller scales. The modified formula of SSIM is given by:

$$\text{SSIM}(x_1, x_2) = \frac{1}{N_b} \sum_{i=1}^{N_b} \frac{(2\mu_{x_1}\mu_{x_2} + C_1)(2\sigma_{x_1x_2} + C_2)}{(\mu_{x_1}^2 + \mu_{x_2}^2 + C_1)(\sigma_{x_1}^2 + \sigma_{x_2}^2 + C_2)} \cdot mask \tag{5.1}$$

where μ and σ are the resulting $54 \times 54 \times 3$ tensors from the 11×11 filter convolution. The $mask$ is computed by convolving x_1 and x_2 segmentations with a 11×11 uniform filter and calculating the pixel-wise logical conjunction.

Finally, we compute the average SSIM from 10000 samples. The resulted SSIM for out model's background consistency is 0.253 ± 0.196 for unaligned backgrounds and 0.311 ± 0.234 for aligned backgrounds. In Figure 5.4 we show two qualitative results for background consistency with unaligned background.

Qualitative results shows us that background loses most of the details when encoded as a condition. Background condition catches the overall color and some variations that resembles tree branches, rocks and others.



(a) (b)

Figure 5.4: Background consistency qualitative results. In both images: first row indicates real images used as background condition; first column the shape conditions. Each line has a fixed text description, which was omitted. Better seen in color.

Besides that, a very interesting result is that the object in the background image is completely ignored and does not affect the generated image. The pixel-wise logical conjunction, used during training, was able to provide the supervision necessary for the background encoder to automatically differentiate background from the foreground. From our knowledge, this is the first time such a technique is employed and it is one of the most important contributions of our work for the deep learning community.

5.1.4 Disentanglement

In this section, we show the results obtained applying our disentanglement score, described in Section 4.3.6. Table 5.3 shows the impact in the foreground and the background by varying the three concepts independently. Higher values mean more changes. We denote the varying concept as "free" in the first column.

Unfortunately, it is not possible to compute a "style change" directly. That would mean an automatic way to uncouple style changes from structural changes made by the free concept. Therefore, our metric gives a general object change, which computes all changes made inside the object mask. That said, we expect text descriptions to change the object style, but keep its structure such as the wing and the beak position. Quantitatively, that means text to have fewer changes in the object than the background in the background itself. Also, we expect the shape to make small changes in the object - possibly the structural changes - and almost no changes in the background. Fortunately, this is exactly what we observe in the results.

Table 5.3: Disentanglement score with $N = 20$ and $M = 20$.

Free	Foreground Changes	Background Changes
Text	**0.48** ± 0.15	0.09 ± 0.04
Background	0.12 ± 0.03	**2.65** ± 0.88
Shape	0.08 ± 0.01	0.05 ± 0.01

(a) (b)

Figure 5.5: Samples for visual inspection of the overlap between style and background. In (a) each row has the same shape and background conditions, but the text description is changing. In (b) each row has the same text and shape condition, but the background is changing. Better seen in color.

Note how each free concept is conditioning each part of the image correctly. However, we can see that background and text have the biggest overlap between them. This is observable during the visual inspection and was an expected behavior since the bird species has a close relationship to its context. Our method was not capable of adjusting this strong statistical sign received during training.

In Figure 5.5 we can see some examples of the overlap between text and background. Figure 5.5a shows some results of background changes varying only text descriptions and Figure 5.5b shows the results of object changes varying only the background.

Lastly, to verify the impact of our word-level representation on disentanglement, we compute the disentanglement measure of our model trained with char-CNN-RNN. This ablation model has all the consistency modules for shape and background disentanglement, however, the word-level discriminator is exchanged by the traditional sentence-level discriminator.

Table 5.4: Disentanglement score with $N = 20$ and $M = 20$ using char-CNN-RNN.

Free	Foreground Changes	Background Changes
Text	0.08 ± 0.03	0.01 ± 0.01
Background	**0.34** ± 0.05	**2.65** ± 1.35
Shape	0.30 ± 0.05	0.08 ± 0.02

Table 5.4 shows that the char-CNN-RNN sentence-level vector strongly harms the disentanglement of concepts. We can see that foreground features are mostly conditioned on background and shape conditions, while the text is being completely ignored. This result confirms our claim that word-level representation plays an indispensable role in disentanglement jointly with the consistency modules.

5.1.5 Shape vs Text

In this section, we show an additional study to evaluate text and shape conflict. We compare the similarity between generated images and text with aligned and unaligned shape conditions from Table 5.2. The expected behavior is that, if there was no conflicting information between shape and text, the similarity should be the same. However, it is known that some of the descriptions have conflicting shape information such as "long beak" and "small bird". As expected, VS drops 6.2% by the use of unaligned shapes when background is aligned (first two rows) and 7.5% when background is not aligned (last two rows).

Finally, to close the control evaluation section, we show in Figure 5.6 some results with the three conditions simultaneously.

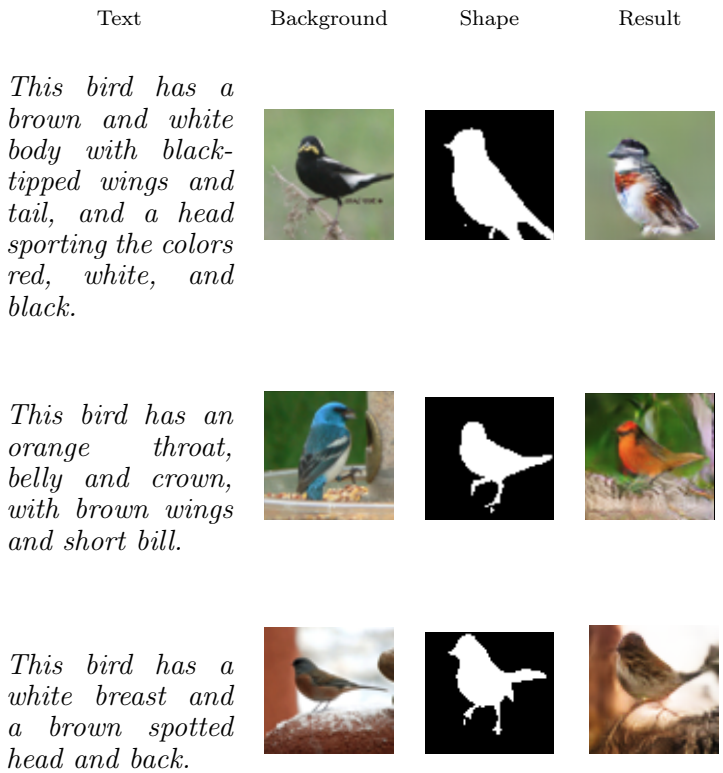


Figure 5.6: Results with the three conditions. Better seen in color.

5.2 Diversity

We evaluate our model's diversity by comparing IS, FID and MS-SSIM against two ablation studies and a state-of-the-art framework. The first ablation was obtained by removing both consistency modules and keeping a traditional text-to-image synthesis framework with word-vector representations (the same called Baseline in Section 5.1.2). The second is an ablation where the word-level representation was exchanged by the sentence-level representation of char-CNN-RNN, keeping the consistency modules. All models with shape and background consistency modules receive shape conditions from the test-set aligned with the text description, while background conditions are chosen randomly from a normal distribution. This is due to the conflict information described in Section 5.1.5 and also due to a lack of smoothness in the learned segmentation latent space, which decreases significantly all metrics. This problem will be better addressed in Section 5.3.2, where we evaluate quantitatively and qualitatively the segmentation latent space.

The HDGAN proposed by Zhang [80] will be used for comparison. IS and FID are omitted due to the high resolution of the HDGAN framework (512×512), which has a high correlation with IS results [80].

Table 5.5: Diversity evaluation. IS is the Inception Score, higher is better; FID is the Fréchet Inception Distance, less is better; MS-SSIM is the Multi-Scale Structural Similarity, in this case less is better.

Model	Text	C. Modules	Seg.	Aligned	Bkg.	IS (+)	FID (-)	MS-SSIM (-)
Ours	sentence	Yes	$z_s \sim \phi(x \sim X)$	Yes	$z_b \sim \mathcal{N}(0,1)$	2.99 ± 0.07	20.53	0.178
Ours (Baseline)	word	No	-	-	-	3.48 ± 0.09	24.38	**0.173**
Ours	word	Yes	$z_s \sim \phi(x \sim X)$	Yes	$z_b \sim \mathcal{N}(0,1)$	**3.55** ± 0.09	**17.08**	0.174
HDGAN [80]	sentence	-	-	-	-	-	-	0.215

Note from Table 5.5 that the consistency modules had an improvement of only 2.01% for IS, while FID had an improvement of 29.94%. This is because IS only measures diversity regarding the 50 test-set classes - intra-class diversity is ignored. Our method was not designed to increase diversity in a class sense, but to improve general variability. On the other hand, FID extracts statistics from high-level features of images and computes the Fréchet distance between real and fake data. In other words, FID considers diversity and realism based on high-level features of the training set, from where the statistics are computed. Individually, we can see that the use of word-vectors also had an important role to increase IS and FID.

On MS-SSIM we had good yet different results. We can see that our work has a much better performance than HDGAN. Photo-realistic frameworks, such as HDGAN, focus its efforts on generating high-quality images. We argue that high-quality images are achieved not only by increasing resolution but by leveraging from common statistics. Since MS-SSIM does not consider realism, our model easily surpasses HDGAN results. However, differently from FID results, MS-SSIM does not indicate an improvement by adding consistency modules.

We believe that the major difference is that FID and IS rely on the use of a deep classifier, which is trained to encode the object for classification. MS-SSIM instead evaluates the entire image equally. We can conclude that our method increases object diversity, but has no significant impact on background diversity.

5.3 Realism

After showing our improvements in control and diversity, we evaluate if the proposed method penalizes realism. Following the literature, we compute IS and FID and compare to previous approaches on text-to-image synthesis. Next, we evaluate the learned latent space smoothness quantitatively and qualitatively through 1D, 2D, and 3D interpolations.

5.3.1 Scores

It is known that IS is directly correlated with the synthesized resolution [80]. Thus, we compare our work only with previous 64×64 resolution models: GAN-INT-CLS [49], StackGAN-I [78] (first stage only), and FusedGAN [5]. We show results with different input configurations for our method to investigate the best and worst case (see Table 5.6).

Table 5.6: Realism evaluation against 64×64 resolution frameworks.

Model	Text	C. Modules	Seg.	Aligned	Bkg.	Aligned	IS (+)	FID (-)
GAN-INT-CLS [49]	-	-	-	-	-	-	2.88 ± 0.04	68.79
StackGAN-I [78]	-	-	-	-	-	-	2.95 ± 0.02	-
FusedGAN [5]	-	-	-	-	-	-	3.00 ± 0.03	-
Ours	word	Yes	$z_s \sim \mathcal{N}(0,1)$	-	$z_b \sim \mathcal{N}(0,1)$	-	3.22 ± 0.08	33.74
Ours	word	Yes	$z_s \sim \phi(x \sim X)$	No	$z_b \sim \psi(x \sim X)$	No	3.35 ± 0.09	17.09
Ours	word	Yes	$z_s \sim \phi(x \sim X)$	Yes	$z_b \sim \psi(x \sim X)$	Yes	**3.63** ± 0.10	**16.33**

As we can see, our model still achieves good performance on IS and FID. Previous methods on 64×64 resolution hit a maximum of 3.00 on IS, while our model - even with conditions sampled from the Gaussian distribution - have a minimum of 3.22. By aligning all conditions and sampling from the test-set, our model achieves the highest performance. Aligned conditions guarantee no conflicting information between shape and text descriptions, and also does not force less common background/object combination (e.g. yellow bird in the water).

Comparatively, the method that uses background and shape conditions sampled from the normal distribution has a significant loss on performance. We believe this is mostly due to the lack of smoothness in the latent space of shapes. This problem will be betteer discussed in the following section.

5.3.2 Latent Space Smoothness

It is difficult to imagine how segmentations should be interpolated to always provide a realistic bird shape. That would mean that every point in the normal distribution is an intermediate realistic position of a bird. For instance, between a bird facing left and a bird facing right, we should have infinite intermediate realistic positions - including a bird facing forward (or backward). Now, imagine this between every possible position the bird might be and every different shape the bird might have.

We hypothesize that, if the model's latent space is smooth enough, results from either segmentation data from the test set or sampling from the known distribution should be similar. Given the amount of data available, it is more likely that some regions are being filled with unrealistic shapes that may harm the generation.

Table 5.7: Quantitative results for different input configurations for the generator.

Segmentation	Aligned	Background	Aligned	IS (+)	FID (-)
$z_s \sim \phi(x \sim X)$	No	$z_b \sim \psi(x \sim X)$	No	**3.35** $\pm$ 0.09	**17.09**
$z_s \sim \mathcal{N}(0,1)$	-	$z_b \sim \psi(x \sim X)$	No	3.29 $\pm$ 0.08	33.89
$z_s \sim \mathcal{N}(0,1)$	-	$z_b \sim \mathcal{N}(0,1)$	-	3.22 $\pm$ 0.08	33.74

Table 5.7 show results for conditions from the test-set and the normal distribution. It confirms our expectations. Both metrics are harmed by sampling shape from the normal distribution. However, we can see that sampling background from the normal distribution does not affect the generation significantly. Background latent space is mostly composed of color variations, which do not have to be realistic for high IS and FID scores. Figure 5.7 shows some samples that were harmed by bad shape conditioning from the normal distribution.

Figure 5.7: Samples generated with bad shape conditioning from the normal distribution. Better seen in color.

Notably, images synthesized by bad shape conditions have bad results. Unfortunately, given the quantitative results, those are very common in the latent space. We believe that

this problem can be addressed by increasing λ_s and giving more relevance to the segmentation mapping during training.

5.3.3 Interpolations

A good method to evaluate the degree of overfitting in a generative model is to explore the model's latent space by interpolation [44]. Overfitted models tend to have discrete transitions in the interpolated images and regions that do not correspond to meaningful images [4, 48, 15]. Furthermore, interpolation also gives us a good visual tool to evaluate smoothness qualitatively. In this section, we show several examples not only for evaluation but also to explore our model's interesting features of 1D, 2D, and 3D interpolation. All interpolations were obtained using shape, text, and background conditions from the test-set.

Figures 5.8, 5.9 and 5.10 depict 1D interpolations for style, shape and background, respectively. Figures 5.11a and 5.11b show 2D interpolations on the shape and background dimensions. Figures 5.12a and 5.12b show 2D interpolations on the style and background dimensions. Figure 5.13a and 5.13b show 2D interpolations on the style and shape dimensions. Finally, Figure 5.14 depicts a 3D interpolation through style, shape, and background dimensions.

Figure 5.8: 1D text description (style) interpolation. Shape and background are kept constant for each row. Better seen in color.

Figure 5.9: 1D shape interpolation. Text and background are kept constant for each row. Better seen in color.

Figure 5.10: 1D background interpolation. Shape and text are kept constant for each row. Better seen in color.

(a) (b)

Figure 5.11: 2D interpolation. Shape (x axis) and background (y axis). Text is kept constant.

(a) (b)

Figure 5.12: 2D interpolation. Style (x axis) and background (y axis). Shape condition is kept constant.

(a) (b)

Figure 5.13: 2D interpolation. Style (x axis) and shape (y axis). Background condition is kept constant.

Figure 5.14: 3D interpolation. On the left, a 2D interpolation with style (x axis) and shape (y axis). For better visualization, we choose the diagonal images to show the third dimension (background) interpolation - on the right.

Apart from the general good results, Figure 5.11a shows how the model may interpolate two shapes with the opposite orientation. The model does not interpolate with intermediate positions, but with a merging shape that looks more like a two-headed bird. On the other hand, in Figure 5.15 we show a good interpolation result of our model in a similar scenario.

Figure 5.15: Interpolation between flipped shapes.

5.4 Ablation Studies

In this section we investigate the impact of our contributions individually. We conduct ablation studies for shape consistency module, background consistency module, fastText word-vectors, and the KL loss used to map segmentations to a normal distribution.

The consistency modules ablations are performed by exchanging the input with a traditional noise $z \sim \mathcal{N}(0, 1)$ and removing the consistency loss setting the hyperparameters to zero. This way, the diversity is not harmed by the absence of input conditions. The fastText ablation is performed by using sentence-level representations from the char-CNN-RNN and the traditional sentence-level discriminator D.

Lastly, the segmentation KL loss ablation is made by setting the hyperparameter to zero. By doing so, segmentation masks are not mapped to a known distribution and there is no guarantee of good results by randomly sampling z_s instead of using a test-set segmentation.

Table 5.8 summarizes all the obtained results for IS, FID, VS, MS-SSIM and IoU. We show additional IS and FID results from sampling shape and background conditions from the normal distribution, denoted as IS-z and FID-z, respectively. MS-SSIM scores follows the protocol used in Section 5.2, with shape conditions from the test-set and background from a standard normal distribution.

Table 5.8: Ablation for segmentation consistency (γ_s), background consistency (γ_b), segmentation KL-divergence (λ_s) and fastText word vectors (Text). All results using conditions from the test-set without alignment, except for MS-SSIM, where we keep the protocol used in Section 5.2.

γ_s	γ_b	λ_s	Text	IS (+)	IS-z (+)	FID (-)	FID-z (-)	VS (+)	MS-SSIM (-)	IoU (+)
10	10	0.5	word	3.35 ± 0.09	3.22 ± 0.08	**17.09**	33.74	0.135 ± 0.183	**0.174**	77.2%
0	10	0.5	word	3.42 ± 0.09	**3.49** ± 0.09	23.40	**25.73**	0.135 ± 0.183	0.177	-
10	0	0.5	word	3.45 ± 0.08	3.28 ± 0.07	17.47	33.57	**0.147** ± 0.186	0.181	76.57%
10	10	0	word	**3.48** ± 0.10	1.93 ± 0.04	17.33	146.74	0.132 ± 0.183	0.298	**82.56%**
10	10	0.5	sentence	3.03 ± 0.08	2.68 ± 0.06	19.15	37.57	-	0.178	77.78%

We can conclude that the shape consistency module plays an important role in diversity. FID scores are highly improved by the use of shape consistency. The shape consistency module forces the shape disentanglement hence the use of mismatched conditions during training, which approximates fake images distribution towards the real data distribution. Also, we observe that text consistency had no impact.

By the background shape consistency removal, we do not see significant changes in IS and FID. However, we can see how text consistency is improved in the absence of background disentanglement. As seen in previous sections, VS is improved by the use of an aligned back-

ground. Although expected, this behavior is not appropriate for a text consistency metric. Text descriptions do not describe background information on the CUB dataset. Thus, we cannot take further conclusions about text coherence on background consistency ablation.

As expected, the removal of KL loss for segmentation harms the use of normal distribution sampling for the shape condition. IS-z and FID-z drop performance to the lowest level we have seen so far. However, we can see that IoU is improved to 82.56%, almost the segmentation network error. Therefore, KL harms the shape consistency to provide the random sampling feature.

The use of word-vectors from fastText has shown to boost general results of IS and FID. Unfortunately, it is not trivial compare semantic consistency across methods with different text representations. Hence, we could not evaluate quantitatively the impact of fastText word-vectors on semantic consistency.

Finally, MS-SSIM results show that the most significant impact is the removal of the shape KL loss. This result confirms that CA modules also improve diversity, as pointed out by Zhang [79]. Furthermore, we confirm that the best result is achieved by the use of all proposed techniques.

6. CONCLUSION

In this work, we introduced the base concepts and the main issues of text-to-image synthesis as well as state-of-the-art efforts on the research community to mitigate them. We summarized general synthesis goals as realism, diversity, and control; and presented a novel disentanglement-based approach to tackle diversity and control in the text-to-image scenario.

Furthermore, we raised some criticism about current approaches for text representation and how it may lead to undesired behaviors for the text-to-image models. Based on that, we proposed the use of a more proper textual representation, which was demonstrated to be also beneficial for our disentanglement technique.

The overall idea of our method was to use disentanglement as a proxy to increase diversity. With the use of extra shape supervision, we could easily disentangle the image into multiple concepts and force uncommon combinations during training. By doing so, the model would have to learn a more complete mapping using the same amount of data. Also, once a concept is disentangled, it can be used to control generation during inference. Our method learns how to encode the information (e.g. background) in such a way that the user can easily choose the overall aspects of the image: from small detailed object structure to background colors.

We built an experimental setup based on current research metrics as well as a self-made measure to evaluate all aspects of our method. First, we evaluated the generation consistency against the three disentangled conditions — style (text), shape, and background — and investigate dataset peculiarities such as text and segmentation conflicts. Second, we measured the degree of disentanglement of our model and the influence of traditional sentence-level representation. Third, we evaluated the impact of our approach on diversity and compared it to state-of-the-art results. Next, we checked if the proposed techniques did not harm realism to increase diversity and control. Finally, we evaluated each component independently conducting several ablation studies for the shape consistency module, the background consistency module, the fastText word-vector representation and segmentation KL loss (used to map conditions to a normal distribution).

Based on the results presented in Section 5.1, we can say that our framework offers fine control over style, pose and background. Via natural language, a user can dictate detailed aspects of the object such as colors and textures. With a 2D binary mask, the user can choose a specific pose, shape or location for the object. Lastly, passing a real image, the algorithm is able to extract background information and copy for a synthesized object. Some improvements for the background are still necessary, but the current results are promising. It is important to point out that none of the above conditions are strictly necessary to generate a synthetic image during inference. Our method preserves all text-to-image features.

Regarding diversity, we could conclude in Section 5.2 that our method highly improves object diversity. Mostly due to the use of the shape consistency module and the word-level

text representation. The fastText representation jointly with the word-level discriminator have shown to be helpful to increase diversity but also played a fundamental role in disentanglement.

Additionally, we investigated in Section 5.3 the impact of our contributions on realism by comparing results with related approaches. Even with the loss of performance by the use of normal distribution sampling, our method still has the best results on FID and IS. Unfortunately, there is no quantitative way of measuring realism independently and we support our claim that our framework does not harm realism based on the degree of improvement on FID and IS.

We investigated the mentioned loss on performance by evaluating quantitatively the latent space smoothness. The hypothesis that a not smooth latent space would cause a drop in performance by sampling from the normal distribution was confirmed. Our method was not able to interpolate the segmentation latent space properly. Most of the shape conditions generate unrealistic images when sampled from a normal distribution and not from a real mask. We believe this issue can be better addressed by a more rigorous search for hyperparameters. Unfortunately, this is computationally demanding, an thus, we leave it for future work.

Finally, our future work also includes: increase resolution, which can easily be made following previous approaches such as stacking multiple GANs [79, 78]; adapt the framework for multi-object datasets, such as MS-COCO [36]; and lastly, expand some of our contributions for image manipulation from textual description, which has already shown promising results.

REFERENCES

[1] Arjovsky, M.; Chintala, S.; Bottou, L. "Wasserstein gan", *Stat*, vol. 1050, December 2017, pp. 9.

[2] Barratt, S.; Sharma, R. "A note on the inception score", *arXiv preprint*, vol. 1801.01973, June 2018, pp. 9.

[3] Bengio, Y.; Courville, A.; Vincent, P. "Representation learning: A review and new perspectives", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35–8, August 2013, pp. 1798–1828.

[4] Bengio, Y.; Mesnil, G.; Dauphin, Y.; Rifai, S. "Better mixing via deep representations". In: International Conference on Machine Learning, 2013, pp. 552–560.

[5] Bodla, N.; Hua, G.; Chellappa, R. "Semi-supervised fusedgan for conditional image generation". In: European Conference on Computer Vision, 2018, pp. 669–683.

[6] Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. "Enriching word vectors with subword information", *Transactions of the Association for Computational Linguistics*, vol. 5, June 2017, pp. 135–146.

[7] Borji, A. "Pros and cons of gan evaluation measures", *Computer Vision and Image Understanding*, vol. 179, February 2019, pp. 41–65.

[8] Chen, T. Q.; Li, X.; Grosse, R. B.; Duvenaud, D. K. "Isolating sources of disentanglement in variational autoencoders". In: Neural Information Processing Systems, 2018, pp. 2610–2620.

[9] Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; Abbeel, P. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In: Neural Information Processing Systems, 2016, pp. 2172–2180.

[10] Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. "Learning phrase representations using rnn encoder–decoder for statistical machine translation". In: Empirical Methods in Natural Language Processing, 2014, pp. 1724–1734.

[11] Dash, A.; Gamboa, J. C. B.; Ahmed, S.; Liwicki, M.; Afzal, M. Z. "Tac-gan-text conditioned auxiliary classifier generative adversarial network", *arXiv preprint*, vol. 1703.06412, March 2017, pp. 9.

[12] DeepAI. "Disentangled representation learning". Source: https://deepai.org/machine-learning-glossary-and-terms/disentangled-representation-learning, 2018.

[13] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. “Imagenet: A large-scale hierarchical image database”. In: Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[14] Denton, E. L.; et al.. “Unsupervised learning of disentangled representations from video”. In: Neural Information Processing Systems, 2017, pp. 4414–4423.

[15] Dinh, L.; Sohl-Dickstein, J.; Bengio, S. “Density estimation using real nvp”. In: International Conference on Learning Representations, 2016, pp. 29.

[16] Dong, H.; Zhang, J.; McIlwraith, D.; Guo, Y. “I2t2i: Learning text to image synthesis with textual data augmentation”. In: International Conference on Image Processing, 2017, pp. 2015–2019.

[17] Dowson, D.; Landau, B. “The fréchet distance between multivariate normal distributions”, *Journal of Multivariate Analysis*, vol. 12–3, March 1982, pp. 450–455.

[18] Eastwood, C.; Williams, C. K. I. “A framework for the quantitative evaluation of disentangled representations”. In: International Conference on Learning Representations, 2018, pp. 15.

[19] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. “Generative adversarial nets”. In: Neural Information Processing Systems, 2014, pp. 2672–2680.

[20] Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; Hochreiter, S. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: Neural Information Processing Systems, 2017, pp. 6626–6637.

[21] Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; Lerchner, A. “beta-vae: Learning basic visual concepts with a constrained variational framework.” In: International Conference on Learning Representations, 2017, pp. 6.

[22] Hinton, G. E.; Krizhevsky, A.; Wang, S. D. “Transforming auto-encoders”. In: International Conference on Artificial Neural Networks, 2011, pp. 44–51.

[23] Hoffman, J.; Tzeng, E.; Park, T.; Zhu, J.-Y.; Isola, P.; Saenko, K.; Efros, A. A.; Darrell, T. “Cycada: Cycle consistent adversarial domain adaptation”. In: International Conference on Machine Learning, 2018, pp. 15.

[24] Hong, S.; Yang, D.; Choi, J.; Lee, H. “Inferring semantic layout for hierarchical text-to-image synthesis”. In: Computer Vision and Pattern Recognition, 2018, pp. 7986–7994.

[25] Hu, Q.; Szabó, A.; Portenier, T.; Favaro, P.; Zwicker, M. “Disentangling factors of variation by mixing them”. In: Computer Vision and Pattern Recognition, 2018, pp. 3399–3407.

[26] IBM. “Disentanglement of latent factors of variation with deep learning”. Source: https://www.ibm.com/blogs/research/2018/05/disentanglement-deep-learning/, 2018.

[27] Ioffe, S.; Szegedy, C. “Batch normalization: accelerating deep network training by reducing internal covariate shift”. In: International Conference on Machine Learning, 2015, pp. 448–456.

[28] Isola, P.; Zhu, J.-Y.; Zhou, T.; Efros, A. A. “Image-to-image translation with conditional adversarial networks”. In: Computer Vision and Pattern Recognition, 2017, pp. 1125–1134.

[29] Jordan, M. I.; Mitchell, T. M. “Machine learning: Trends, perspectives, and prospects”, *Science*, vol. 349–6245, July 2015, pp. 255–260.

[30] Joseph, K.; Pal, A.; Rajanala, S.; Balasubramanian, V. N. “C4synth: Cross-caption cycle-consistent text-to-image synthesis”. In: Winter Conference on Applications of Computer Vision, 2019, pp. 358–366.

[31] Kim, H.; Mnih, A. “Disentangling by factorising”. In: International Conference on Machine Learning, 2018, pp. 2654–2663.

[32] Kingma, D. P.; Ba, J. “Adam: A method for stochastic optimization”. In: International Conference for Learning Representations, 2014, pp. 15.

[33] Kingma, D. P.; Welling, M. “Auto-encoding variational bayes”, *Stat*, vol. 1050, May 2014, pp. 1.

[34] Kumar, A.; Sattigeri, P.; Balakrishnan, A. “Variational inference of disentangled latent concepts from unlabeled observations”. In: International Conference on Learning Representations, 2017, pp. 16.

[35] Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al.. “Photo-realistic single image super-resolution using a generative adversarial network”. In: Computer Vision and Pattern Recognition, 2017, pp. 4681–4690.

[36] Lin, T.; Maire, M.; Belongie, S. J.; Bourdev, L. D.; Girshick, R. B.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C. L. “Microsoft COCO: common objects in context”. In: European Conference on Computer Vision, 2014, pp. 740–755.

[37] Ma, K.; Wu, Q.; Wang, Z.; Duanmu, Z.; Yong, H.; Li, H.; Zhang, L. “Group mad competition-a new methodology to compare objective image quality models”. In: Computer Vision and Pattern Recognition, 2016, pp. 1664–1673.

[38] Manaris, B. “Natural language processing: A human-computer interaction perspective”. In: *Advances in Computers*, Elsevier, 1998, vol. 47, chap. 1, pp. 1–66.

[39] Matthey, L.; Higgins, I.; Hassabis, D.; Lerchner, A. "dsprites: Disentanglement testing sprites dataset". Source: https://github.com/deepmind/dsprites-dataset/, 2017.

[40] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. "Efficient estimation of word representations in vector space", *arXiv preprint*, vol. 1301.3781, September 2013, pp. 12.

[41] Milletari, F.; Navab, N.; Ahmadi, S.-A. "V-net: Fully convolutional neural networks for volumetric medical image segmentation". In: International Conference on 3D Vision, 2016, pp. 565–571.

[42] Mirza, M.; Osindero, S. "Conditional generative adversarial nets", *arXiv*, vol. 1411.1784, November 2014, pp. 7.

[43] Nam, S.; Kim, Y.; Kim, S. J. "Text-adaptive generative adversarial networks: Manipulating images with natural language". In: Neural Information Processing Systems, 2018, pp. 42–51.

[44] Odena, A.; Olah, C.; Shlens, J. "Conditional image synthesis with auxiliary classifier gans". In: International Conference on Machine Learning, 2017, pp. 2642–2651.

[45] Park, H.; Yoo, Y.; Kwak, N. "Mc-gan: Multi-conditional generative adversarial network for image synthesis". In: The British Machine Vision Conference, 2018, pp. 13.

[46] Pennington, J.; Socher, R.; Manning, C. "Glove: Global vectors for word representation". In: Empirical Methods in Natural Language processing, 2014, pp. 1532–1543.

[47] Qiao, T.; Zhang, J.; Xu, D.; Tao, D. "Mirrorgan: Learning text-to-image generation by redescription". In: Computer Vision and Pattern Recognition, 2019, pp. 1505–1514.

[48] Radford, A.; Metz, L.; Chintala, S. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: International Conference on Learning Representations, 2015, pp. 16.

[49] Reed, S.; Akata, Z.; Yan, X.; Logeswaran, L.; Schiele, B.; Lee, H. "Generative adversarial text to image synthesis". In: International Conference on Machine Learning, 2016, pp. 1060–1069.

[50] Reed, S. E.; Akata, Z.; Mohan, S.; Tenka, S.; Schiele, B.; Lee, H. "Learning what and where to draw". In: Neural Information Processing Systems, 2016, pp. 217–225.

[51] Ridgeway, K. "A survey of inductive biases for factorial representation-learning", *arXiv preprint*, vol. 1612.05299, December 2016, pp. 36.

[52] Ronneberger, O.; Fischer, P.; Brox, T. "U-net: Convolutional networks for biomedical image segmentation". In: International Conference on Medical image computing and computer-assisted intervention, 2015, pp. 234–241.

[53] Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. "Improved techniques for training gans". In: Neural Information Processing Systems, 2016, pp. 2234–2242.

[54] Sharma, S.; Suhubdy, D.; Michalski, V.; Kahou, S. E.; Bengio, Y. "Chatpainter: Improving text to image generation using dialogue". In: Computer Vision and Pattern Recognition, 2018, pp. 7.

[55] Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; Webb, R. "Learning from simulated and unsupervised images through adversarial training". In: Computer Vision and Pattern Recognition, 2017, pp. 2107–2116.

[56] Singh, K. K.; Ojha, U.; Lee, Y. J. "Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery". In: Computer Vision and Pattern Recognition, 2019, pp. 6490–6499.

[57] Srivastava, A.; Valkov, L.; Russell, C.; Gutmann, M. U.; Sutton, C. "Veegan: Reducing mode collapse in gans using implicit variational learning". In: Neural Information Processing Systems, 2017, pp. 3308–3318.

[58] Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. A. "Inception-v4, inception-resnet and the impact of residual connections on learning". In: AAAI Conference on Artificial Intelligence, 2017, pp. 4278–4284.

[59] Tenenbaum, J. B.; Freeman, W. T. "Separating style and content with bilinear models", *Neural Computation*, vol. 12–6, June 2000, pp. 1247–1283.

[60] Theis, L.; van den Oord, A.; Bethge, M. "A note on the evaluation of generative models", *Stat*, vol. 1050, April 2015, pp. 5.

[61] Tolstikhin, I. O.; Gelly, S.; Bousquet, O.; Simon-Gabriel, C.-J.; Schölkopf, B. "Adagan: Boosting generative models". In: Neural Information Processing Systems, 2017, pp. 5424–5433.

[62] Trusov, R. "Inception score — evaluating the realism of your gan". Source: https://sudomake.ai/inception-score-explained/, 2018.

[63] Van Oord, A.; Kalchbrenner, N.; Kavukcuoglu, K. "Pixel recurrent neural networks". In: International Conference on Machine Learning, 2016, pp. 1747–1756.

[64] Vaserstein, L. N. "Markov processes over denumerable products of spaces, describing large systems of automata", *Problemy Peredachi Informatsii*, vol. 5–3, January 1969, pp. 64–72.

[65] Wah, C.; Branson, S.; Welinder, P.; Perona, P.; Belongie, S. "The caltech-ucsd birds-200-2011 dataset", Technical Report CNS-TR-2011-001, California Institute of Technology, 2011, 15p.

[66] Wang, X.; Gupta, A. “Generative image modeling using style and structure adversarial networks”. In: European Conference on Computer Vision, 2016, pp. 318–335.

[67] Wang, Z.; Bovik, A. C.; Sheikh, H. R.; Simoncelli, E. P.; et al.. “Image quality assessment: from error visibility to structural similarity”, *IEEE Transactions on Image Processing*, vol. 13–4, April 2004, pp. 600–612.

[68] Wehrmann, J.; Barros, R. C. “Bidirectional retrieval made simple”. In: Computer Vision and Pattern Recognition, 2018, pp. 7718–7726.

[69] Wehrmann, J.; Lopes, M. A.; More, M. D.; Barros, R. C. “Fast self-attentive multimodal retrieval”. In: Winter Conference on Applications of Computer Vision, 2018, pp. 1871–1878.

[70] Wehrmann, J.; Lopes, M. A.; Souza, D. M.; Barros, R. C. “Language-invariant embeddings for cross-modal retrieval”. In: International Conference on Computer Vision, 2019, pp. 9.

[71] Welinder, P.; Branson, S.; Perona, P.; Belongie, S. J. “The multidimensional wisdom of crowds”. In: Neural Information Processing Systems, 2010, pp. 2424–2432.

[72] Xu, T.; Zhang, P.; Huang, Q.; Zhang, H.; Gan, Z.; Huang, X.; He, X. “Attngan: Fine-grained text to image generation with attentional generative adversarial networks”. In: Computer Vision and Pattern Recognition, 2018, pp. 1316–1324.

[73] Yan, X.; Yang, J.; Sohn, K.; Lee, H. “Attribute2image: Conditional image generation from visual attributes”. In: European Conference on Computer Vision, 2016, pp. 776–791.

[74] Yang, J.; Kannan, A.; Batra, D.; Parikh, D. “Lr-gan: Layered recursive generative adversarial networks for image generation”. In: International Conference on Learning Representations, 2017, pp. 21.

[75] Yeh, R.; Chen, C.; Lim, T. Y.; Hasegawa-Johnson, M.; Do, M. N. “Semantic image inpainting with perceptual and contextual losses”, *arXiv preprint*, vol. 1607.07539, July 2016, pp. 3.

[76] Yin, G.; Liu, B.; Sheng, L.; Yu, N.; Wang, X.; Shao, J. “Semantics disentangling for text-to-image generation”. In: Computer Vision and Pattern Recognition, 2019, pp. 2327–2336.

[77] Yuan, M.; Peng, Y. “Text-to-image synthesis via symmetrical distillation networks”. In: ACM Multimedia Conference on Multimedia Conference, 2018, pp. 1407–1415.

[78] Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Huang, X.; Wang, X.; Metaxas, D. “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks”. In: International Conference on Computer Vision, 2017, pp. 5907–5915.

[79] Zhang, H.; Xu, T.; Li, H.; Zhang, S.; Wang, X.; Huang, X.; Metaxas, D. N. "Stackgan++: Realistic image synthesis with stacked generative adversarial networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41–8, August 2018, pp. 1947–1962.

[80] Zhang, Z.; Xie, Y.; Yang, L. "Photographic text-to-image synthesis with a hierarchically-nested adversarial network". In: Computer Vision and Pattern Recognition, 2018, pp. 6199–6208.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br