C4Synth: Cross-Caption Cycle-Consistent Text-to-Image Synthesis

K J Joseph Arghya Pal Sailaja Rajanala Vineeth N Balasubramanian
IIT Hyderabad, India
cs17m18p100001@iith.ac.in

Abstract

Generating an image from its description is a challenging task worth solving because of its numerous practical applications ranging from image editing to virtual reality. All existing methods use one single caption to generate a plausible image. A single caption by itself, can be limited, and may not be able to capture the variety of concepts and behavior that may be present in the image. We propose two deep generative models that generate an image by making use of multiple captions describing it. This is achieved by ensuring 'Cross-Caption Cycle Consistency' between the multiple captions and the generated image(s). We report quantitative and qualitative results on the standard Caltech-UCSD Birds (CUB) and Oxford-102 Flowers datasets to validate the efficacy of the proposed approach.

1. Introduction

The days when imagination was constrained by a human's visualizing capabilities are gradually passing behind us. Through text to image synthesis, works such as [14, 22, 24, 25] have introduced us to the possibility of visualizing through textual descriptions. Text-to-image synthesis has found itself a foothold in many real-world applications such as virtual reality, tourism, image editing, gaming, and computer-aided design. More mathematically said, the problem is that of modeling $P(\mathbf{I}|\mathbf{t})$: $\mathbf{I}$ being the generated image, $\mathbf{t}$ the raw text (user descriptions). Conditioning $P(\mathbf{I}|\mathbf{t})$ on raw text may not essentially capture the details since the descriptions themselves could be vague. In general, the current trend to overcome this, is to employ distributed text representations to encode the word as a function of its concepts, yielding a text encoding $\phi(\mathbf{t})$. This brings conceptually similar words together and scatters the dissimilar words, giving us a rich text representation to model $P(\mathbf{I}|\phi(\mathbf{t}))$ rather than $P(\mathbf{I}|\mathbf{t})$.

However, as the saying goes: 'A picture is worth a thousand words', the information that is conveyed by the visual perception of an image is difficult to be captured by a single textual description (caption) of the image. In order to alleviate this semantic gap, standard image captioning datasets like COCO [8] and Pascal Sentences [12] provide five captions per image. We show how the use of multiple captions that contain complementary information aid in generating lucid images. It is analogous to having a painter update a canvas each time, after reading different descriptions of the end image that (s)he is painting. Captions with unseen information help the artist to add new concepts to the existing canvas. On the other hand, a caption with redundant concepts improves the rendering of the existing sketch.

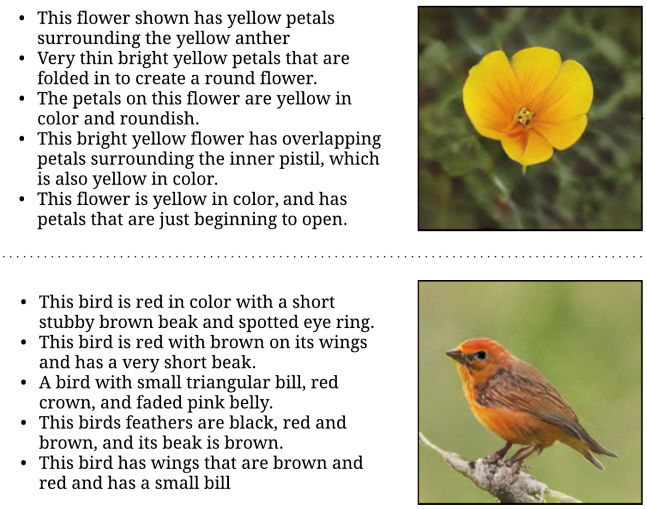


Figure 1: The figure shows two images generated by C4Synth. The corresponding captions that are used while generating images are listed on the left side. (Best viewed in color.)

We realize the aforementioned vision via C4Synth, a deep generative model that iteratively updates its generated image features by taking into account different captions at each step. To assimilate information from multiple captions, we use an adversarial learning setup using Generative Adversarial Networks (GANs) [4], consisting of i generator-discriminator pairs in a serial manner: each conditioned on the current caption encoding $\phi(\mathbf{t})_i$ and *history*

block B_i. We ensure that the captions and the generated image features satisfy a cyclical consistency across the set of captions available. Concretely, let $F_i : \mathbf{t}_i \rightarrow \mathbf{I}_i$ and $G_i : \mathbf{I}_i \rightarrow \mathbf{t}_i$; where $\mathbf{t}$ represents a caption, $\mathbf{I}$ represents an image, F_i transforms the i^{th} caption to the corresponding image representation and G_i does the opposite. A network that is consistent with two captions, for example, is trained such that $G_2 \circ F_2 \circ G_1 \circ F_1(\mathbf{t}) \approx \mathbf{t}$. This model takes inspiration from Cycle-GAN [28] which has demonstrated superior performance in unpaired image-to-image translation. We delve into the details of the cycle-consistency and how this is implemented through a cascaded approach, which we call Cascaded-C4Synth, in Section 4.

The scope of Cascaded-C4Synth is limited by the number of generator-discriminator pairs which are in turn dependent on the number of captions at hand and requires training multiple generator-discriminator pairs in a serial manner. However, the number of available captions can vary across the datasets. This calls for a recurrent version of Cascaded-C4Synth, which we christen as Recurrent-C4Synth, which is not bound by the number of captions. In Recurrent-C4Synth, the images are generated conditioned on a caption and a hidden-state which acts as a memory to capture dependencies among multiple captions. The architecture is explained in detail in Section 4.3.

The key contributions of this work are as follows:

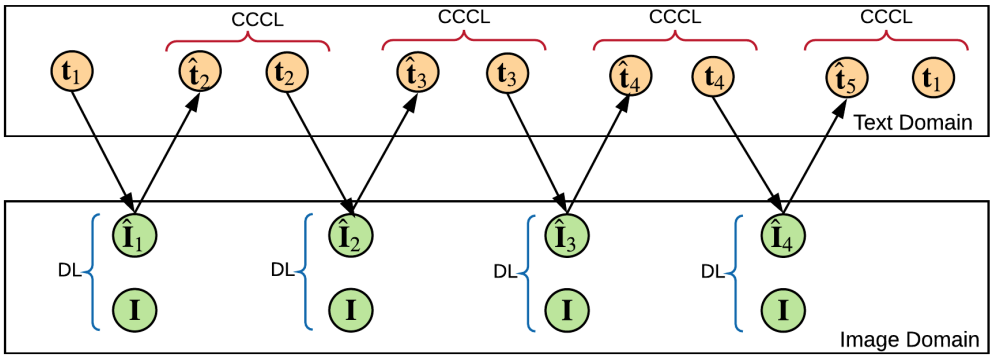
- We propose a methodology for image generation using a medley of captions. To the best of our knowledge, this is the first such effort.
- We introduce a Cross-Caption Cycle-Consistency (and hence, the name C4Synth) as a means of amalgamating information in multiple concepts to generate a single image.
- We supplement the abovementioned method by inducing a recurrent structure that removes the limitation of number of captions on the architecture.
- Our experiments (both qualitative and quantitative) on the standard Caltech-UCSD Birds (CUB) [19] and Oxford-102 Flowers [11] datasets show that both our models, Cascaded-C4Synth and Recurrent-C4Synth, generate real-like, plausible images given a set of captions per sample. As an interesting byproduct, we showcase the model's capability to generate stylized images that vary in the pose and background, however, consistent with the set of captions.

2. Related Work

Text to Image Synthesis: In the realm of GANs, text-to-image synthesis is qualified as a conditional Generative Adversarial Network (cGAN) that transforms human-written text descriptions to the pixel space. The admission of a text into pixel space was realized using deep symmetric structured joint embeddings followed by a cGAN in Reed's *et al*. seminal work [14]. This was the first end-to-end differentiable architecture from character-level to pixel-level generation and showed efficacy in generating real-like images. Subsequently, StackGAN [25] and its follow-up work, StackGAN++ [24], increased the spatial resolution of the generated image by adopting a two-stage process. In StackGAN, low-resolution images (64×64) generated by the first stage are used to condition the second stage along with the caption embedding to generate higher resolution (256×256) images, with significant improvement in quality. Conditional augmentation was introduced to ensure continuity on the latent space of text embedding while maintaining the same richness in the newly induced text space. This is ensured by sampling from a Gaussian distribution whose mean vector and covariance matrix is a function of the caption. Most recently, to be able to consider appropriate parts of a given text description, AttnGAN [22] makes use of an attention mechanism, along with a multi-stage GAN, for improved image generation. A hierarchical nested network is proposed in [26] to assist the generator in capturing complex image statistics.

Despite the aforementioned few efforts, it is worth noting that all the methods so far in the literature use only one caption to generate images, despite the availability of multiple captions in datasets today. Our proposed method iteratively improves the image quality by distilling concepts from multiple captions. Our extensive experimentation stands a testimony to the claim that utilization of multiple captions isn't merely an aggregation of object mentions, but a harmony of complex structure of objects and their relative relations. To strengthen our claim, we quote one such work [16] that loosely corresponds to our idea. The authors improve the image quality by taking into account the dialogues (questions and answers) about an image along with the captions. Though the work shows impressive improvement, the process of answer collection is not similar to multi-captioning and imposes an extra overhead to the system thereby, aggravating the supervision budget for intricate images. This separates our work from their effort.

Cycle Consistency: Cycle-consistent adversarial networks, i.e. CycleGAN [28], has shown impressive results in unpaired image-to-image translation. CycleGAN learns two mappings, $G : A \rightarrow B$ and $F : B \rightarrow A$ using two generators G and F. A and B can be unpaired images from any two domains. For learning the mapping, they introduce a cycle-consistency loss that checks if $F(G(A)) \approx A$ and $G(F(B)) \approx B$. Standard discriminator loss ensures that the images generated by G and F are plausible. Several methods like [23, 27, 9, 6] with similar ideas has extended the CycleGAN idea more recently in literature. All of them consider only pairwise cycle consistency to ac-



(a) *Legend* : $\{\mathbf{t}_i\}$: True Captions; $\{\hat{\mathbf{t}}_i\}$: Generated Captions; $\{\hat{\mathbf{I}}_i\}$: Generated Images; $\mathbf{I}$: True Image; CCCL: Cross-Caption Cycle Consistency Loss; DL: Discriminator Loss.

Figure 2: The figure shows how Cross-Caption Cycle Consistency is maintained across four captions $(\mathbf{t}_1, \cdots, \mathbf{t}_4)$. A generator G converts $\mathbf{t}_i$ to an image $\hat{\mathbf{I}}_i$. Discriminator at each step forces $\hat{\mathbf{I}}_i$ to be realistic. A Cross-Caption Cycle Consistency Network (CCCN) converts $\hat{\mathbf{I}}_i$ back to a caption $(\hat{\mathbf{t}}_{i+1})$. The Cross Caption Consistency Loss (CCCL) forces it to be close to $\mathbf{t}_{i+1}$. In the last step, $\hat{\mathbf{t}}_5$ is ensured to be consistent with the initial caption $\mathbf{t}_1$, hence completing a cycle.

complish real-world applications such as sketch-to-image generation, real image-to-anime character generation, etc. Our proposed approach takes the idea one step ahead and imposes a transitive consistency across multiple captions. We call this Cross-Caption Cycle Consistency, which is explained in Section 4.1.

Recurrent GAN Architectures: Recurrent GAN was proposed to model data with a temporal component. In particular, Vondrick *et al*. [18] uses this idea to generate small realistic video clips and, Ghosh *et al*. [3] depict the use of a Recurrent GAN architecture to make predictions on abstract reasoning tasks by conditioning on the previous context or the history. More relevant examples come from the efforts in [5, 2], which display the potential of recurrent GAN architectures in generating better quality images. The generative process spans across time, building the image step by step. [2] utilizes this time lapse to enhance an attribute of the object at a time. We exploit this recurrent nature to continually improve upon the history while generating an image. Unlike previous efforts, we differ in how we model and use the recurrent aspect of the model, and how we update the hidden state of the recurrent model. To the best of our knowledge, the proposed architecture is the first to use a recurrent formulation for the text-to-image synthesis problem.

3. Preliminaries

3.1. Generative Adversarial Networks

GANs are generative models that sidestep the difficulty in approximating intractable probabilistic computations associated with maximum likelihood estimation and related strategies by matching the generative model (G) with an adversary (D), which learns to discriminate whether the samples are coming from the model distribution ($p_{data}(\mathbf{x})$) or the data distribution ($p_{\mathbf{z}}(\mathbf{z})$). G and D play the following min-max game with the value function $V(D, G)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

In the proposed architecture, we make use of a conditional GAN [10] in which both the generator and the discriminator are conditioned on a variable $\phi(\mathbf{t})$ yielding $G(\mathbf{z}|\phi(\mathbf{t}))$ and $D(\mathbf{x}|\phi(\mathbf{t}))$, where $\phi(\mathbf{t})$ is a vector representation of the caption.

3.2. Text embedding

The text embedding of the caption that we use to condition the GAN, would yield best results if it could bear a semantic correspondence with the actual image that it represents. One method for such a text encoding is Structured Joint Embeddings (SJE) initially proposed by Akata *et al*. [1] and further improved by Reed *et al*. [13]. They learn a text encoder, $\varphi(\mathbf{t})$, which transforms the caption $\mathbf{t}$ in such a way that its inner product with the corresponding image embedding, $\theta(v)$, will be higher if they belong to the same class and lower otherwise. For a training set $\{(v_n, t_n, y_n : n = 1, \cdots, N)\}$, where v_n, t_n and y_n corresponds to image, text and the class label, $\varphi(\mathbf{t})$ is learned by optimizing the following structured loss:

$$\frac{1}{N}\sum_{n=1}^{N} \Delta(y_n, f_v(v_n)) + \Delta(y_n, f_t(t_n)) \text{ where}$$

$$f_v(v) = \arg\max_{y\in Y} \mathbb{E}_{t\sim T(y)}[\theta(v)^T \varphi(t)] \text{ and}$$

$$f_t(t) = \arg\max_{y\in Y} \mathbb{E}_{v\sim V(y)}[\theta(v)^T \varphi(t)]$$

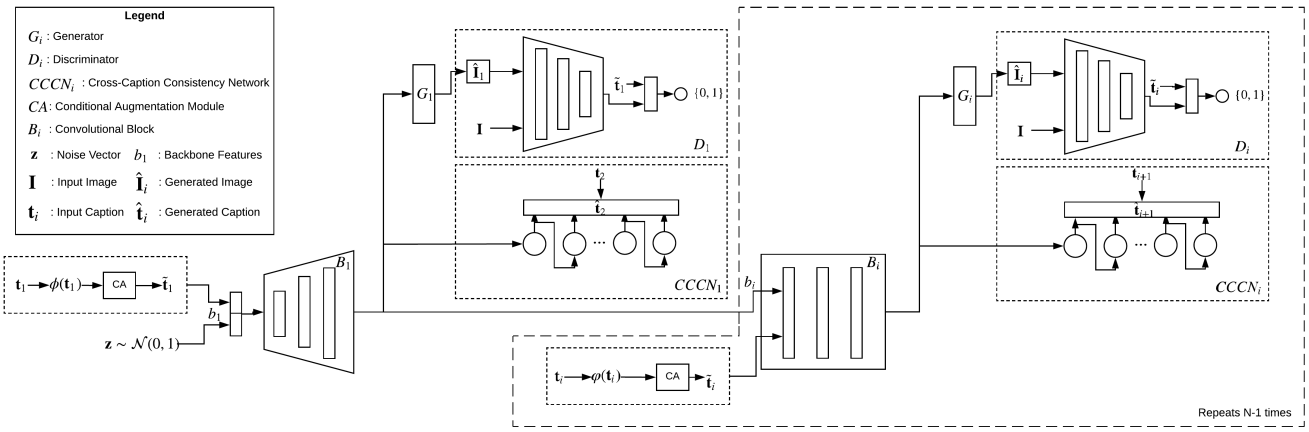


Figure 3: Figure depicts the cascaded architecture of Cascaded-C4Synth. A series of generators conditioned on N captions one by one and previously generated image through a non-linear mapping (convolutional block B_i). Presently, the value N is set to be 3.

After the network is trained [1], we use $\varphi(\mathbf{t})$ to encode the captions. Similar method has been used in previous methods for text to image generation [14, 25, 24, 16, 17]. $\varphi(\mathbf{t})$ is a high dimensional vector. To transform it to a lower dimensional conditioning latent variable, Han *et al*. [25] proposed the 'Conditional Augmentation' technique. Here, the latent vector is randomly sampled from an independent Gaussian distribution whose mean vector and covariance matrix is parameterized by $\varphi(\mathbf{t})$. We request the reader to refer to [25] for more information.

4. Methodology

The main contribution of our work is to formulate a framework to generate images by utilizing information from multiple captions. This is achieved by ensuring Cross-Caption Cycle Consistency. The generic idea of Cross-Caption Cycle Consistency is explained in Section 4.1. We devise two network architectures that maintain this consistency. The first one is a straightforward instantiation of the idea, where multiple generators progressively generate images by consuming captions one by one. This method is explained in Section 4.2. A serious limitation of this approach is that the network architecture restricts the number of captions that can be used to generate an image. This leads us to formulate a recurrent version of the method, where a single generator recursively consumes any number of captions. This elegant method is explained in Section 4.3.

4.1. Cross-Caption Cycle Consistency

Cross-Caption Cycle Consistency is achieved by ensuring that the generated image is consistent with a set of captions describing the same image. Figure 2 gives a simplified overview of the process. Let us take an example of synthesizing an image by distilling information from four captions. In the first iteration, a generator network (G) takes noise and the first caption, $\mathbf{t}_1$, as its input, to generate an image, $\hat{\mathbf{I}}_1$, which is passed to the discriminator network (D), which verifies whether it is real or not. As in a usual GAN setup, generator tries to create better looking images so that it can fool the discriminator. The generated image features are passed on to a 'Cross-Caption Cycle Consistency Network' (CCCN) which will learn to generate a caption for the image. While training, the Cross-Caption Cycle Consistency Loss ensures that the generated caption is similar to the second caption, $\mathbf{t}_2$.

In the next iteration, $\hat{\mathbf{I}}_1$ and $\mathbf{t}_2$ is fed to the generator to generate $\hat{\mathbf{I}}_2$. While D urges G to make $\hat{\mathbf{I}}_2$ similar to the real image $\mathbf{I}$, the CCCN ensures that the learned image representation is consistent for generating the next caption in sequence. This repeats until when $\hat{\mathbf{I}}_4$ gets generated. Here, the CCCN will ensure that the generated caption is similar to the first caption, $\mathbf{t}_1$. Hence we complete a cycle: $\mathbf{t}_1 \rightarrow \mathbf{t}_2 \rightarrow \mathbf{t}_3 \rightarrow \mathbf{t}_4 \rightarrow \mathbf{t}_1$, while generating $\hat{\mathbf{I}}_1 \cdots \hat{\mathbf{I}}_4$ in-between. $\hat{\mathbf{I}}_4$ contains the concepts from all the captions and hence is much richer in quality.

4.2. Cascaded-C4Synth

In our first approach, we consider Cross-caption Cycle Consistent image generation as a cascaded process where a series of generators consumes multiple captions one by one, to generate images. The image that is generated at each step is a function of the previous image and the caption supplied at the current stage. This enables each stage to build up on the intermediate images generated in the previous stage, by utilizing the additional concepts from the new captions seen in the current stage. At each stage, a separate discriminator and CCCN is used. The discriminator is tasked to identify whether the generated image is fake or real while the CCCN

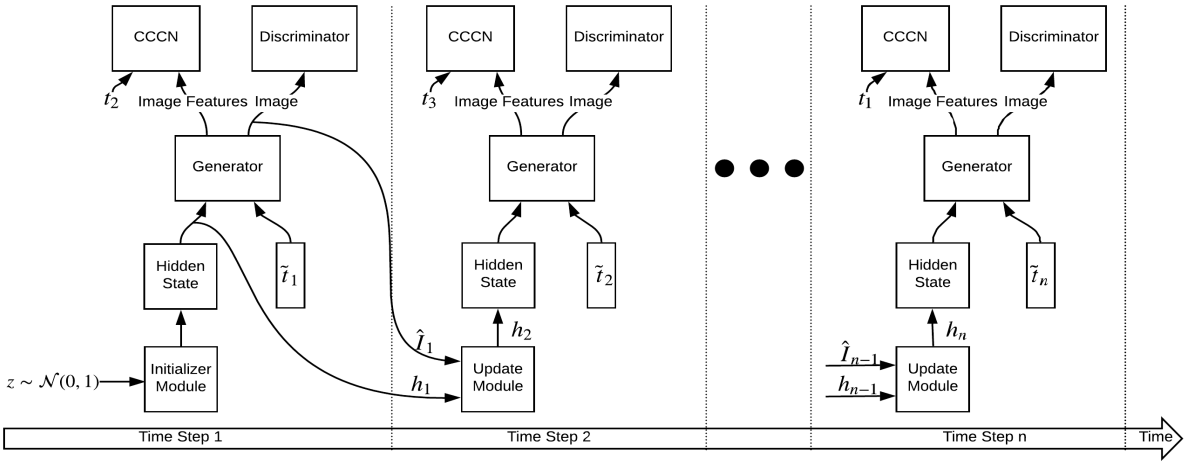


Figure 4: Architecture of Recurrent-C4Synth. The figure shows the network unrolled in time. h_i refers to the hidden state at time step i. t_i is the caption and $\hat{t}_i$ is the vector representation of t_i at time step i.

translates the image to its corresponding caption and checks how much close it is to the next caption in succession.

The architecture is presented in Figure 3. A set of convolutional blocks (denoted by B_i, in the figure) builds up the backbone of the network. The first layer of each B_i consumes a caption. Each generator (G_i) and CCCN ($CCCN_i$) branches off from the last layer of each B_i, while a new B_i attaches itself to grow the backbone. The number of B_i's is fixed while designing the architecture and restricts the number of captions that can be used to generate an image. The main components of the architecture are explained below.

4.2.1 Backbone Network

A vector representation ($\tilde{t}_i$) for each caption (t_i) is generated using Structured Joint Embedding ($\phi(t_i)$) followed by Conditional Augmentation module. A brief description of the text encoding is presented in Section 3.2. $\tilde{t}_i$ is a vector of 128 dimension. In the first convolutional block, B_1, $\tilde{t}_1$ is combined with a 100 dimensional noise vector (z), sampled from a standard normal distribution. The combined vector is passed through fully connected layers and then reshaped into $4 \times 4 \times 16N_g$ tensor. Four up-sampling layers, up-samples the tensor to $64 \times 64 \times 8N_g$ tensor. This tensor is passed on to the first generator (G_1) and the first CCCN ($CCCN_1$).

Further convolutional blocks, B_i, are added to B_1 as follows. The new caption encoding $\tilde{t}_i$, is spatially replicated at each location of the backbone features (b_i) coming from the previous convolutional block (B_i), followed by a 3×3 convolution. These features are passed thorough a residual block and an up-sampling layer. This helps to increase the spatial resolution of the feature maps in each B_i. Hence, the output of B_2 is a tensor of size $128 \times 128 \times 4N_g$. The generator and CCCN branches off from this feature map as before, and a new convolutional block (B_3) gets added. In our experiments, we used three B_is, due to GPU memory limitations. N_g is set to 32.

4.2.2 Generator

Each generator (G_i) takes the features from the backbone network and passes it through a single 3×3 convolutional layer followed by a tanh activation function to generate an RGB image. As the spatial resolution of the features from each B_i increases (as explained in Section 4.2.1), the size of the image generated by each generator, also increases.

Multiple generators are trained together by minimizing the following loss function:

$$\mathcal{L}_G = \sum_{i=1}^{N} \mathcal{L}_{G_i}, \text{ where } \mathcal{L}_{G_i} = \mathbb{E}_{\mathbf{s}_i \sim p_{G_i}}[\log(1 - D_i(\mathbf{s}_i))] + \lambda D_{KL}(\mathcal{N}(\mu(\phi(\mathbf{t}_i), \Sigma(\mathbf{t}_i)) || \mathcal{N}(0,1))$$

The first term in $\mathcal{L}_{G_i}$ is the standard minimization term in the GAN framework which pushes the generator to generate better quality images. p_{G_i} is the distribution of the generator network. The D_{KL} term is used to learn the parameters of $\mu(\phi(\mathbf{t}_i))$ and $\Sigma(\mathbf{t}_i)$ of the Conditional Augmentation framework [25]. It is learned very similar to the reparameterization trick in VAEs [7]. λ is a regularization parameter, whose value we set to 1 for the experiments.

4.2.3 Discriminator

The discriminators (d_i) contains a set of down-sampling layers which converts the input tensor to $4 \times 4 \times 18N_d$ tensor. Following the spirit of conditional GAN [10], the encoded caption, $\tilde{t}_i$ is spatially replicated and joined by a 3×3 convolution to the incoming image. The final logit

is obtained by convolving with a $4 \times 4 \times 1$ kernel and a Sigmoid activation function.

The loss function for training the discriminator is as follows:

$$\mathcal{L}_{D_i} = \mathbb{E}_{\mathbf{I}_i \sim p_{data}}[\log D_i(\mathbf{I}_i)] + \mathbb{E}_{\mathbf{s}_i \sim p_{G_i}}[\log(1 - D_i(\mathbf{s}_i))]$$

p_{data} is the original data distribution and p_{G_i} is the distribution of the corresponding generator network. The multiple discriminators are trained in parallel.

4.2.4 Cross-Caption Cycle Consistency Network

CCCN is modeled as an LSTM which generates one word at each time-step conditioned on a context vector (derived by attending to specific regions of the image), the hidden state and the previously generated word. CCCN takes as input the same set of backbone features that the generator consumes. It is then pooled to reduce the spatial dimension. Regions of these feature maps are aggregated into a single context vector by learning to attend to these feature maps similar to the method proposed by [21]. Each word is encoded as its one-hot representation.

There is one CCCN block per generator. CCCN is trained by minimizing the cross-entropy loss between each of the generated words and words in the true caption. The true caption for Stage i is $(i+1)^{th}$ caption, and finally the first caption, as is explained in Section 4.1. The loss of each of the CCCN block is aggregated and back-propagated together.

4.3. Recurrent-C4Synth

The architecture of Cascaded-C4Synth limits the number of captions that can be consumed because the number of generator-discriminator pairs has to be decided and fixed during training. We overcome this problem by formulating a recurrent approach for text to image synthesis. At its core, Recurrent-C4Synth maintains a hidden state, which guides the image generation at each time step, along with a new caption. The hidden state by itself is modeled as a function of the previous hidden state and the image that was generated in the previous time step. This allows the hidden state to act like a shared memory, that captures the essential features from the different captions to generate good looking, semantically rich images. The exact way in which hidden state is updated is explained in Section 4.3.1.

Figure 4 presents the simplified architecture of Recurrent-C4Synth. We explain the architecture in detail here. The hidden state is realized as an $8 \times 8 \times 8$ tensor. The values for the initial hidden state is learned by the Initializer Module, which takes as input a noise vector (z) of length 100, sampled randomly from a Gaussian distribution. It is passed though a fully connected layer followed by non linearity and finally reshaped into a $8 \times 8 \times 8$ tensor. Our experimentations reveal that initializing the hidden state with Initializer Module helps the model to learn better than randomly initializing the same.

The hidden state along with the text embedding of the caption is passed to the generator to generate an Image. A discriminator guides the generator to generate realistic image while a Cross-Caption Cycle Consistency Network (CCCN) ensure that the captions that are generated from the image features are consistent with the second caption. As we unroll the network in time, different captions are fed to the generator at each time step. When the final caption is fed in, the CCCN makes sure that it is consistent with the first caption. Hence the network ensures that the cycle consistency between captions is maintained.

The network architecture of CCCN is same as that of Cascaded-C4Synth, while the architecture of the generator and discriminator is slightly different. We explain them in section 4.3.2. While Cascaded-C4Synth has separate generator, and the corresponding discriminator and CCCN at each stage, the Recurrent-C4Synth has only one generator, discriminator and CCCN. The weights of the generator is shared across time steps and is updated via Back Propagation Through Time (BPTT)[20].

4.3.1 Updating the Hidden State

In the first time step of the unrolled network, the hidden state is initialized by the Initializer Module. In the successive time steps, the hidden state and the image generated in the previous time step is used to generate the new hidden state, as shown in figure 4. The 64×64 images are down-sampled by a set of down-sampling convolutional layers to generate feature maps of spatial dimension 8×8. These feature maps are fused with the hidden state (also of spatial dimension 8×8) by eight 3×3 filters. This will result in a new hidden state of dimension $8 \times 8 \times 8$. If we denote the above operation by a function $UpdateHiddenState(.)$, then the recurrence relation at each time-step i, can be expressed as:

$$\hat{I}_i = Generator(h_i, \phi(t_1))$$
$$h_i = UpdateHiddenState(h_{i-1}, I_{i-1})$$

$\hat{I}_i$ is the image generated by the Generator, by consuming the hidden state (h_i) and the vector representation of the caption ($\phi(t_1)$) that was provided in time step i.

4.3.2 Generator and Discriminator

Recurrent-C4Synth uses a single generator to generate images of size 256×256. It consumes the hidden state h_i, and a vector representation $\phi(t_i)$ of the caption provided in the current time step. $\phi(t_i)$ is spatially replicated to each location of the hidden state and then fused by a 3×3 convolu-

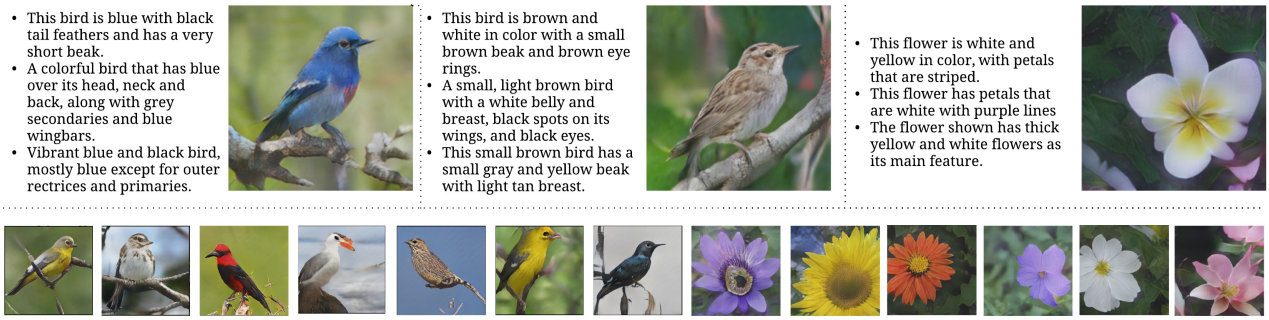


Figure 5: Generations from Cascaded-C4Synth. The first row shows the images generated and the corresponding captions consumed in the process. The first two images belong to Indigo Bunting, Tree Sparrow class of CUB dataset [19] and the last image belongs to Peruvian Lily class of Flowers dataset [11]. The bottom row showcases some random samples of generated images. (Kindly zoom in to see the detailing in the images.)

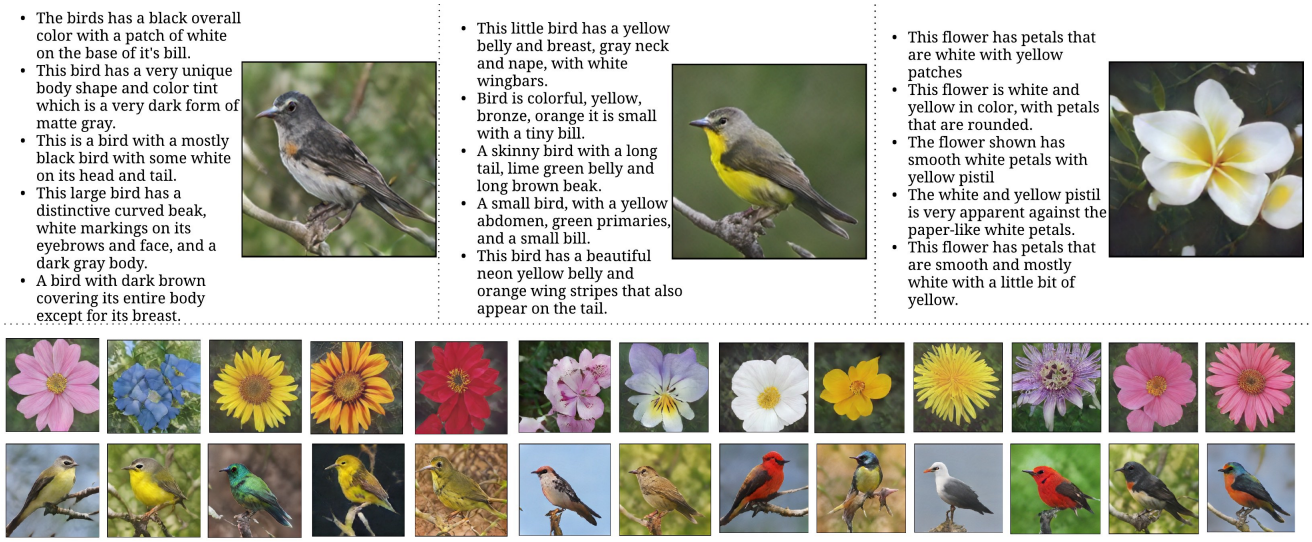


Figure 6: Generations from Recurrent-C4Synth. The first two images are generated from the caption belonging to Black Footed Albatross class and Great Crested Flycatcher class of CUB dataset [19], while the last one is from the Moon Orchid class of Flowers dataset [11]. The last two rows contains random generations from both the datasets. (Kindly zoom in to see the detailing in the images.)

tion layer. This results in a feature map of spatial resolution 8×8.

One easy way to generate 256×256 images from these feature maps would be to stack five up-convolution layers (each doubling the spatial resolution) back to back. Our experiments showed that such a method will not work in practice. Hence, we choose to generate intermediate images of spatial resolution 64×64 and 128×128 also. This is achieved by attaching $3 \times 3 \times 3$ kernels after the third and fourth up-sampling layer. The extra gradients (obtained by discriminating the intermediate images) that flow through the network will help the network to learn better.

In-order to discriminate the two intermediate images and the final image, we make use of three separate discriminators. The architecture of each of the discriminator is similar to Cascaded-C4Synth.

5. Experiments and Results

5.1. Datasets and Evaluation Criteria

We evaluate Cascaded-C4Synth and Recurrent-C4Synth on Oxford-102 flowers dataset [11] and Caltech-UCSD Birds (CUB) [19] datasets. Oxford-102 contains 102 categories of flowers counting to 8189 images in total, while CUB contains 200 bird species with 11,788 images. Following the previous methods [14, 25, 24], we pre-process the dataset to improve the object to image ratio.

We gauge the performance of the generated images by its

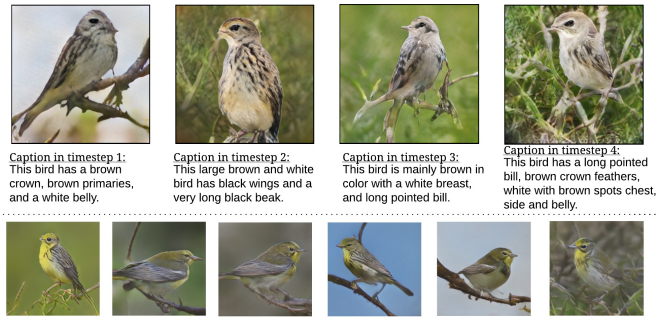


Figure 7: The top row shows the images generated by Recurrent-C4Synth at each time-step. The corresponding captions that was consumed is also added. The bottom row shows generated birds of the same class, but with varying pose and background. These are generated by keeping the captions the same and varying the noise vector used to condition the GAN.

'Inception Score'[15], which has emerged as the dominant way of measuring the quality of generative models. The inception model has been fine-tuned on both the the datasets so that we can have a fair comparison with previous methods like [14, 25, 24, 26].

5.2. Results

We validate the efficacy of Cascaded-C4Synth and Recurrent-C4Synth by comparing it with GAN-INT-CLS [14], GAWWN [13], StackGAN [25], StackGAN++ [24] and HD-GAN [26] (Our future work will include integrating attention in our framework, and comparing against attention-based frameworks such as [17]).

5.2.1 Quantitative Results

Method	Oxford-102 [11]	CUB [19]
GAN-INT-CLS [14]	$2.66 \pm .03$	$2.88 \pm .04$
GAWWN [13]	-	$3.62 \pm .07$
StackGAN [25]	$3.20 \pm .01$	$3.70 \pm .04$
StackGAN++ [24]	-	$3.82 \pm .06$
HDGAN [26]	$3.45 \pm .07$	$\mathbf{4.15 \pm .05}$
Cascaded C4Synth	$3.41 \pm .17$	$3.92 \pm .04$
Recurrent C4Synth	$\mathbf{3.52 \pm .15}$	$4.07 \pm .13$

Table 1: Comparison of C4Synth methods with other text to image synthesis methods. The number reported are Inception Scores (higher is better).

Table 1 summarizes the Inception Score of competing methods on Oxford-102 flowers dataset [11] and Caltech-UCSD Birds (CUB) [19] dataset along with the results of C4Synth models. On Oxford-102 dataset, Recurrent-C4Synth method gives state-of-the-art result, improving the previous baseline. On CUB dataset, the results are comparable with HDGAN [26].

The results indicate that Recurrent-C4Synth has an edge over Cascaded-C4Synth. It is worth noting that both the methods perform better than four out of five other baseline methods.

5.2.2 Qualitative results

Figure 5 and 6 shows the generations from Cascaded-C4Synth and Recurrent-C4Synth methods respectively. The generations from Cascaded-C4Synth method consumes three captions, as is restricted by the architecture, while the Recurrent-C4Synth method consumes five captions. The quality of the images generated by both the methods are comparable as is evident from the Inception Scores. All the generated images are of 256×256 pixels in resolution. The supplementary section contains more image generations.

The images that are generated at each time step by the Recurrent-C4Synth method is captured in the top row of Figure 7. The captions that are consumed in each step is also shown. This figure validates our assertion that the recurrent formulation progressively generates better images by consuming one caption at a time.

The bottom row of Figure 7 shows the interpolation of the noise vector, used to generate the hidden state of Recurrent-C4Synth, while fixing the captions used. This results in generating the same bird in different orientations and backgrounds.

5.2.3 Zero Shot generations

We note that while training both the C4Synth architectures with Oxford-102 flowers dataset [11] and Caltech-UCSD Birds (CUB) [19] datasets, the classes used for training and testing are disjoint. We use the official train-test split for both the datasets. CUB has 150 train+val classes and 50 test classes, while Oxford-102 has 82 train+val classes and 20 test classes. Hence all the results shown in the paper are zero-shot generations, where none of the classes of captions that are used to generate the image in test phase, has ever been seen in training phase.

6. Conclusion

We formulate two generative models for text to image synthesis, Cascaded-C4Synth and Recurrent-C4Synth, which makes use of multiple captions to generate an image. The method is able to generate plausible images on Oxford-102 flowers dataset [11] and Caltech-UCSD Birds (CUB) [19] dataset. We believe that attending to specific parts of the captions at each stage, would improve the results of our method. We will explore this in a future work. The code and trained models will be made available.

References

[1] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 2927–2936. IEEE, 2015.

[2] R. Y. Benmalek, C. Cardie, S. Belongie, X. He, and J. Gao. The neural painter: Multi-turn image generation. *arXiv preprint arXiv:1806.06183*, 2018.

[3] A. Ghosh, V. Kulharia, A. Mukerjee, V. Namboodiri, and M. Bansal. Contextual rnn-gans for abstract reasoning diagram generation. *arXiv preprint arXiv:1609.09444*, 2016.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[5] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic. Generating images with recurrent adversarial networks. *CoRR*, abs/1602.05110, 2016.

[6] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

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

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[9] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

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

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

[12] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147. Association for Computational Linguistics, 2010.

[13] S. Reed, Z. Akata, H. Lee, and B. Schiele. Learning deep representations of fine-grained visual descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–58, 2016.

[14] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[15] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[16] S. Sharma, D. Suhubdy, V. Michalski, S. E. Kahou, and Y. Bengio. Chatpainter: Improving text to image generation using dialogue. *ICLR Workshops*, 2018.

[17] Q. H. H. Z. Z. G. X. H. X. H. Tao Xu, Pengchuan Zhang. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. 2018.

[18] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. *CoRR*, abs/1609.02612, 2016.

[19] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[20] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. *Backpropagation: Theory, architectures, and applications*, 1:433–486, 1995.

[21] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[22] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. *arXiv preprint arXiv:1711.10485*, 2017.

[23] Z. Yi, H. Zhang, P. Tan, and M. Gong. Dualgan: Unsupervised dual learning for image-to-image translation. *arXiv preprint*, 2017.

[24] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *arXiv: 1710.10916*, 2017.

[25] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[26] Z. Zhang, Y. Xie, and L. Yang. Photographic text-to-image synthesis with a hierarchically-nested adversarial network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[27] Z. Zhang, L. Yang, and Y. Zheng. Translating and segmenting multimodal medical volumes with cycle-and shape-consistency generative adversarial network. *arXiv preprint arXiv:1802.09655*, 2018.

[28] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.