Sync-DRAW: Automatic GIF Generation using Deep Recurrent Attentive Architectures

Gaurav Mittal*
gaurav.mittal.191013@gmail.com

Tanya Marwah*
IIT Hyderabad
ee13b1044@iith.ac.in

Vineeth N. Balasubramanian
IIT Hyderabad
vineethnb@iith.ac.in

Abstract

This paper introduces a novel approach for generating GIFs called Synchronized Deep Recurrent Attentive Writer (Sync-DRAW). Sync-DRAW employs a Recurrent Variational Autoencoder (R-VAE) and an attention mechanism in a hierarchical manner to create a temporally dependent sequence of frames that are gradually formed over time. The attention mechanism in Sync-DRAW attends to each individual frame of the GIF in sychronization, while the R-VAE learns a latent distribution for the entire GIF at the global level. We studied the performance of our Sync-DRAW network on the Bouncing MNIST GIFs Dataset and also, the newly available TGIF dataset. Experiments have suggested that Sync-DRAW is efficient in learning the spatial and temporal information of the GIFs and generates frames where objects have high structural integrity. Moreover, we also demonstrate that Sync-DRAW can be extended to even generate GIFs automatically given just text captions.

1. Introduction

Naturally occurring images and videos are very complex high-dimensional data, which makes them hard to model. Over the years, several generative and discriminative approaches have been proposed to perform tasks such as classification and recognition on such data. However, very recently, there has been a significant shift in how generative models are trained and used in computer vision. Contemporary deep generative models not only aim to learn the underlying distribution of the data but also in the process, attempt to endow the network with the capability to discriminate between two sets of input as well as create samples resembling the training data. Among the generative models that have been introduced in the last 2-3 years, Variational Autoencoders (VAEs) [16] and Generative Adversarial Networks (GANs) [7] have emerged to be the most promising deep learning-based approaches for image generation.

*Equal Contribution

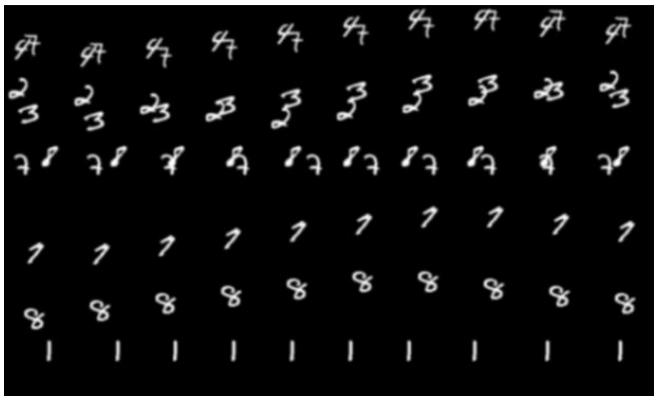

Figure 1. GIF frames automatically generated by Sync-DRAW using single-digit and two-digit bouncing MNIST GIFs datasets.

Existing efforts (described further in Section 2) in the last couple of years have primarily focused on generation of images using the aforementioned methods, by modeling the spatial structure of a given training dataset [9][19][26], as well as extending these methods to more novel applications such as adding texture/style to images [6]. Very limited work has been carried out in extending such generative models to videos, with most existing efforts focusing on video prediction [24]. Instead, we propose a novel network called Sync-DRAW which uses a Recurrent VAE to automatically generate animated GIF sequences that are similar to training GIF data. Sync-DRAW also takes a step further by learning to associate GIFs with captions, and thus learning to generate GIF sequences conditioned on a human-understandable stimulus such as text caption. To the best of our knowledge, this work is the first effort to automatically generate image sequences (GIFs) of this kind. Figure 1 shows examples of GIF frames generated by a trained Sync-DRAW model.

In recent prior work, image generation using a VAE has been performed using the Deep Recurrent Attentive Writer (DRAW) architecture [9]. DRAW uses a Recurrent Variational Autoencoder (R-VAE) to generate an image as a sequence of canvases over time, and an intriguing atten-

tion mechanism which allows the generation to follow the foveation of a human eye. The proposed Sync-DRAW combines these two components in a novel hierarchical fashion to allow the R-VAE to learn the entire video at a global level, while having a local attention mechanism that attends to each frame separately in *'synchronization'*; and hence, the name *Sync-DRAW*. While the proposed work is relevant to videos in general, we focus on the generation of animated GIFs in this work primarily owing to the reason that considering such an effort has not been done before, GIFs provide us with short well-defined image sequence settings to test and validate the proposed ideas of this work. Besides, the growth of social media has led to an increased use of animated GIFs, thus making generation of GIFs an interesting application area by itself. Further, the increasing popularity in the use of animated GIFs has also resulted in public availability of GIF-based datasets such as [18].

The remainder of this paper is organized as follows: Section 2 discusses the background and earlier efforts related to our work, Section 3 describes the proposed Sync-DRAW methodology, Section 4 describes the experimental results including the datasets used in this work, and Section 5 concludes the paper with pointers to future work.

2. Background and Related Work

Over the last decade, deep learning models have seen a rapid evolution in the way they are used for video processing and understanding. While early efforts focused on traditional feature learning and classification using Convolutional Neural Networks (CNNs) [13, 14, 23], subsequent efforts attempted to solve more interesting problems such as semantic segmentation in videos of natural scenes [4, 5, 20]. This was soon followed by the rise of Recurrent Neural Networks (RNNs), which allowed deep models to learn sequential information. The popularity of RNNs led to several models of CNNs combined with RNNs to model videos and their respective understanding, such as [2]. An exciting phase in this evolution of deep learning models over the last few years stands in our midst today, which is the recent focus towards automatic generation of data including applications such as handwriting generation [8], character prediction [25], and image generation [9, 21]. This work is a contribution to this new and developing subarea of deep learning.

In order to effectively generate any meaningful information such as an image, a learning model should be capable of learning the latent hierarchy of feature representations for any given data, thus explaining the success of deep architectures [1] in this regard. Initial efforts in generating such data were based on Deep Belief Networks [10] and Deep Boltzmann Machines [22], which contained many layers of latent variables and millions of parameters. However, their overwhelming dependence on Markov Chain Monte Carlo-based sampling methods made them difficult to scale. Over the last few years, these models have been replaced by newer approaches such as Generative Adversarial Networks (GANs) [7] and Variational Auto-Encoders (VAEs) [16], which have proven to perform quite effectively on generative tasks, and have been shown to scale to large datasets. A GAN comprises of a generator and a discriminator network pitted against each other, where the generator network attempts to fool the other by generating samples resembling the dataset, and the discriminator attempts to differentiate between these sampled images and classify them as real or artificial. VAEs, on the other hand, have been developed based on principles from variational Bayesian inference [16], where a directed graphical model learns the latent distribution of the data, and data is generated by sampling from this latent distribution. The proposed work is built on the VAE approach to data generation.

The Deep Recurrent Attentive Writer (DRAW) [9] was the earliest work to utilize a Recurrent-VAE (R-VAE) to learn to generate images by progressively refining an image canvas over time using a sequence of samples generated from a learnt latent distribution. DRAW further combines this R-VAE with an attention mechanism [17], which allows the generated image to focus on one part of the image at each timestep. This notion of attention resembles human visual perception, and hence, has generated interest and been used recently for related applications such as generating captions for images [28] or even generating images from captions [19]. However, there has been no effort, to the best of our knowledge, on using such VAE-based approaches to model spatio-temporal latent representations, thereby generating image sequences/videos, which we address in this work.

In the unsupervised domain, there have been a few recent efforts that attempt to model videos using such generative models, which primarily focus on video prediction. For example, one of the earliest efforts in this regard, [24], seeks to learn unsupervised video representations using Long Short-Term Memory units (LSTMs) [11] to predict future frames in videos. However, these approaches are fundamentally different from the objectives of this work, since they do not attempt to generate complete videos or GIFs from scratch.

As mentioned earlier, the efforts closest to this work are DRAW [9] and AlignDRAW [19]. While we use a R-VAE similar to the above efforts, our work uses the attention mechanism and the R-VAE differently in a hierarchical manner, where a single R-VAE is used at a GIF level to capture spatiotemporal relationships, while we embed a separate attention mechanism in each frame of the GIF to capture local saliency. Further, we demonstrate that our approach holds the potential to automatically generate GIFs from captions alone. We now describe the methodology behind Sync-DRAW.

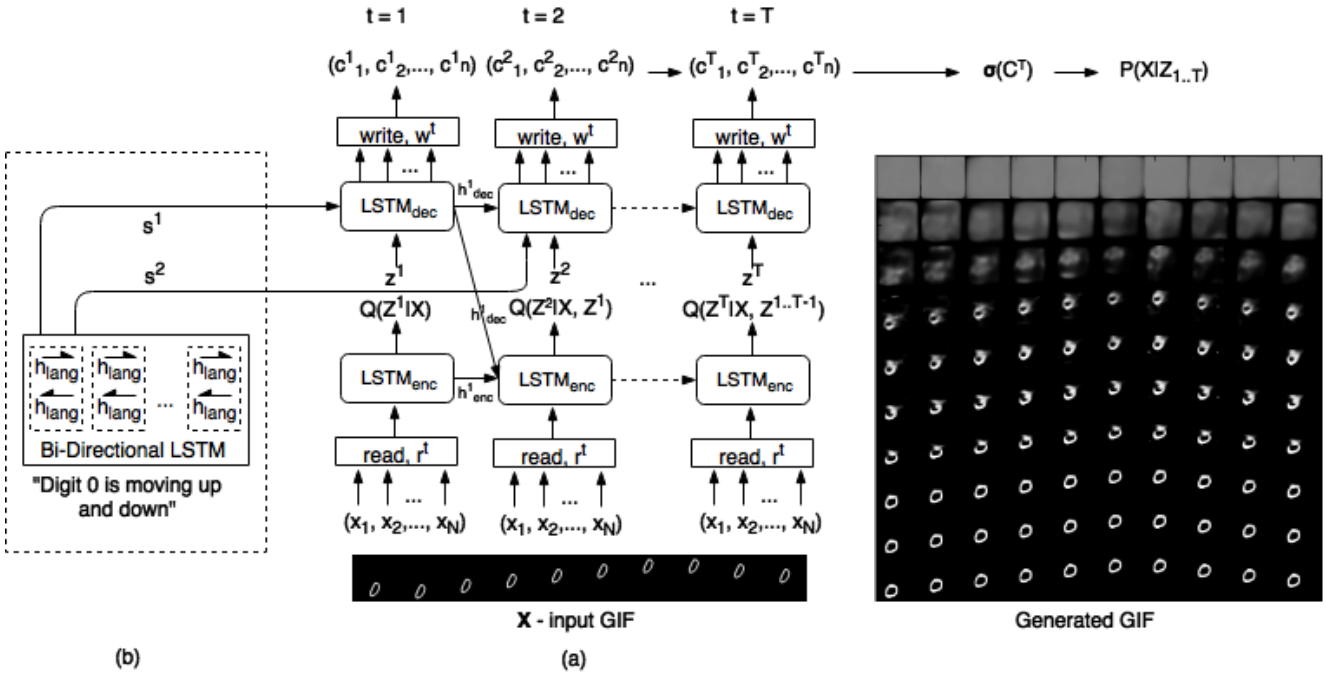

Figure 2. Sync-DRAW architecture: (a) depicts the architecture of Sync-DRAW highlighting the read and write mechanisms and the R-VAE; (b) depicts the extensibility of Sync-DRAW to incorporate the sentence alignment mechanism for generation of GIFs from captions.

3. Sync-DRAW Methodology

To the best of our knowledge, our proposed Sync-DRAW (Synchronized Deep Recurrent Attentive Writer) methodology is the first to use recent developments in Variational Auto-Encoders to go beyond the generation of still images and automatically learn to generate an animated GIF, which is a sequence of temporally related frames. Our methodology builds on the recently developed Deep Recurrent Attentive Writer (DRAW) architecture [9], which however is only restricted to generation of still images. DRAW comprises of a Recurrent Variational Auto-Encoder (R-VAE) coupled with an attention mechanism. These work together to generate an image by accumulating a sequence of canvases using the attention mechanism that works well in the spatial domain. However, in order to generate a temporally related sequence of frames, while it is trivially possible to have separate R-VAEs for drawing each frame, this approach is not practically feasible: (i) the number of frames in an image sequence may not be fixed, thus making it infeasible to decide the number of R-VAEs; and (ii) the number of parameters to be learnt across the VAEs may explode exponentially. Besides, this approach ignores the temporal relationship between the frames. In order to harness the spatio-temporal relationships, we propose a hierarchical approach where we have an R-VAE at the video level and an attention mechanism working at the frame level. We will later see that keeping a different attention mechanism for every frame is essential to ensure the objects in the GIFs have a well defined structure and boundaries. We now describe the architecture of Sync-DRAW.

3.1. Sync-DRAW Architecture

Let $X = \{x_1, x_2, \cdots, x_N\}$ be a GIF comprising of frames $x_i, i = 1, \cdots, N$ where N is the number of frames in the GIF. Let the dimensions for every frame be denoted by $A \times B$. We generate X over T timesteps, where at each timestep t, canvases for all frames are generated in synchronization. The Sync-DRAW architecture comprises of: (i) a *read mechanism* which takes a "glimpse" (portion of importance) from each frame of the GIF (Section 3.1.1); (ii) the *R-VAE*, which is responsible to learn a latent distribution from these glimpses (Section 3.1.2); and (iii) a *write mechanism* which finally generates a canvas for each frame (Section 3.1.3). Additionally, our approach is extensible to generate GIFs from just text captions, using the alignment approach introduced in [19] (Section 3.2). The complete network architecture in shown in Figure 2. Each of these components of Sync-DRAW is described individually below.

3.1.1 Read Mechanism

The read attention mechanism in Sync-DRAW is responsible for reading a particular patch ("glimpse") from each frame x_i at every timestep t. Similar to [9], the patch is read by dynamically computing $F^t_{i_p}$ and $F^t_{i_q}$, which correspond

to a set (of size K, a user-defined parameter) of $1 \times A$ and $1 \times B$-dimensional Gaussian filters respectively, for each frame x_i at timestep t, with i_p and i_q being the two spatial dimensions of frame x_i. (For convenience, we will refer to $F^t_{i_p}$ and $F^t_{i_q}$ as being of sizes $K \times A$ and $K \times B$ respectively.) In order to compute these filters, we obtain the following set of read attention parameters for each frame, $\tilde{g}^t_{i_p}, \tilde{g}^t_{i_q}, \sigma^t_i, \tilde{\delta}^t_i, \beta^t_i$ where $i \in \{1, \cdots, N\}$.

To ensure that $\tilde{g}^t_{i_p}, \tilde{g}^t_{i_q}$ and $\tilde{\delta}^t_i$ lie within the frame dimensions they are modified using the following equations:

$$g^t_{i_p} = \frac{A+1}{2}(\tilde{g}^t_{i_p} + 1); g^t_{i_q} = \frac{B+1}{2}(\tilde{g}^t_{i_q} + 1)$$

$$\delta^t_i = \frac{\max(A,B)-1}{N-1}\tilde{\delta}^t_i$$

$g^t_{i_p}$ and $g^t_{i_q}$ are used to calculate the centers for the horizontal and vertical filterbanks, $F^t_{i_p}$ and $F^t_{i_q}$, using the following equations:

$$\mu^t_{i_{p_u}} = g^t_{i_p} + (u - N/2 - 0.5)\delta^t_i \quad (1)$$

$$\mu^t_{i_{q_v}} = g^t_{i_q} + (v - N/2 - 0.5)\delta^t_i \quad (2)$$

for $u, v \in \{1, \cdots, K\}$. σ^t_i serves as the standard deviation for all filterbanks at timestep t and i^{th} frame. $F^t_{i_p}$ and $F^t_{i_q}$ are hence obtained as follows:

$$F^t_{i_p}[u, a] = \frac{1}{Z_{i_p}} \exp\Big(-\frac{(a - \mu^t_{i_{pu}})^2}{2(\sigma^t_i)^2}\Big) \quad (3)$$

$$F^t_{i_q}[v, b] = \frac{1}{Z_{i_q}} \exp\Big(-\frac{(b - \mu^t_{i_{q_v}})^2}{2(\sigma^t_i)^2}\Big) \quad (4)$$

where $a \in \{1, \cdots, A\}$, $b \in \{1, \cdots, B\}$, $u, v \in \{1, \cdots, K\}$ and Z_{i_p}, Z_{i_q} are the normalization constants.

The last parameter, β^t_i, plays a pivotal role in allowing the network to learn the temporal relationships between the patches read from the frames. It lets the model decide the level of importance to be given to each frame for generating the GIF at any time step. Before feeding to the R-VAE, the patch read from each frame of the input GIF is scaled by its respective $\beta^t_i, i \in \{1, \cdots, N\}$ as follows:

$$read(x_i) = \beta^t_i (F^t_{i_p}) x_i (F^t_{i_q})^T \quad (5)$$

where $read(x_i)$ is of size $K \times K$.

3.1.2 R-VAE

The R-VAE forms the core of the Sync-DRAW architecture, and is responsible to generate a single sample from the latent distribution of the entire GIF at each time step. The core components of a standard R-VAE are the encoder LSTM, $LSTM_{enc}$ (which outputs h_{enc}), the latent representation, z, and the decoder LSTM, $LSTM_{dec}$ (which outputs h_{dec}) [3]. We now describe how we derive each of these components in Sync-DRAW.

As mentioned earlier, the DRAW architecture [9] uses an R-VAE along with an attention mechanism to draw a sequence of canvases (over T time steps) to finally generate a still image. Instead, in the proposed Sync-DRAW architecture, the R-VAE runs for T time steps and generates a set of canvases $C^t = \{c^t_1, c^t_2, \cdots, c^t_N\}$ at every timestep t, where $\boldsymbol{\sigma}(C^t)$ is the GIF generated after t timesteps. The fact that the R-VAE works at the GIF level ensures that the canvases for the frames across the GIF are generated in synchronization and by doing so, the temporal dependencies between the frames are preserved. At every t, we define a new quantity, $\hat{X}^t$, which represents the error, as follows:

$$\hat{X}^t = X - \boldsymbol{\sigma}(C^{t-1}) \quad (6)$$

where $\boldsymbol{\sigma}$ is the sigmoid function, and X is the input GIF. The LSTM-encoder at time t is then processed as follows:

$$h^t_{enc} = LSTM_{enc}\left(h^{t-1}_{enc}, \left[R^t, h^{t-1}_{dec}\right]\right) \quad (7)$$

where $R^t = [r^t_1, r^t_2, \cdots, r^t_N]$ with $r^t_i = [read(x_i), read(\hat{x}^t_i)]$. Next, h^t_{enc} is used to compute the variable z^t which is sampled from the latent distribution $Q\left(Z^t | h^t_{enc}\right)$ (as in [9]). This sampling is carried out using the reparameterization technique described in [9] and [19]:

$$z^t \sim Q\left(Z^t | h^t_{enc}\right) \quad (8)$$

where z^t is the global latent representation for the entire GIF at time t. z^t is used to generate the entire GIF, while the local attention parameters (per frame) capture the temporal intricacies and also the structure of the digits is maintained between the frames. z_t is fed to the LSTM-decoder, $LSTM_{dec}$, producing h^t_{dec}:

$$h^t_{dec} = LSTM_{dec}\left(h^{t-1}_{dec}, z^t\right) \quad (9)$$

h^t_{dec} is used to compute parameters for the write attention mechanism as described in the next section. We note that the read attention parameters (in Section 3.1.1) at time t are computed using a linear function of h^{t-1}_{dec}, whose coefficients are learnt during training, and σ, $\tilde{\delta}$, and β are defined in the logarithm scale to ensure that values are always positive.

3.1.3 Write Mechanism

As part of the write mechanism, we need two things per frame: what to write, and where to write. The former is obtained using an attention window, w^t, which is computed as a linear function of h^t_{dec}, whose co-efficients are learnt during training. The latter is obtained by defining a new set of parameters (mirroring the parameters in the read mechanism), which are now computed using h^t_{dec} (as against using

h_{dec}^{t-1} for the read parameters). Next, a set of filterbanks $\hat{F}_{i_p}^t$, $\hat{F}_{i_q}^t$ are computed using these write attention parameters in a similar fashion as in Section 3.1.1. Finally, the canvas c_i^t corresponding to every frame is created as follows:

$$write(h_{dec}^t) = \frac{1}{\hat{\beta}_i^t}(\hat{F}_{i_p}^t)^T w_i^t (\hat{F}_{i_q}^t) \tag{10}$$

$$c_i^t = c_i^{t-1} + write(h_{dec}^t) \tag{11}$$

For more information regarding the read and write mechanisms, we refer the interested reader to [9].

3.2. Sync-DRAW with Text Alignment: Generating GIFs from Captions

We can further extend Sync-DRAW to generate GIFs given a caption by following the methodology in [19]. We first train a separate bidirectional LSTM, $LSTM_{lang}$ (Figure 2b) on the captions. The encoding from $LSTM_{lang}$, h_{lang}, is then used to generate an alignment of the captions with h_{dec}^{t-1} to give a vector of alignment probabilities, α_k^t (which capture the weight assigned to each word in the caption). Using these α's, a sentence representation, s^t, for each time step t, pertaining to the entire GIF, is computed as:

$$s_t = \alpha_1^t h_1^{lang} + \alpha_2^t h_2^{lang} + \ldots + \alpha_N^t h_N^{lang} \tag{12}$$

$$\alpha_i^t = \frac{\exp(v^\top \tanh(U h_i^{lang} + W h_{t-1}^{gen} + b))}{\sum_{k=1}^N \exp(v^\top \tanh(U h_k^{lang} + W h_{t-1}^{gen} + b))} \tag{13}$$

Equation 9 is then modified by concatenating z^t with s^t to give the new LSTM-decoder:

$$h_{dec}^t = LSTM_{dec}^t\left(h_{dec}^{t-1}, [z^t, s^t]\right) \tag{14}$$

The rest of the components, read and write mechanisms, remain the same.

3.3. Loss Function

The loss function for Sync-DRAW is composed of two types of losses, both of which are computed at the GIF level. The first is the reconstruction loss, L_X, that is defined as the binary cross-entropy loss between the pixels comprising of the original GIF X and the corresponding ones in the generated GIF, $\boldsymbol{\sigma}(C^T)$. Since the GIFs in TGIF have been normalized to be between 0 and 1, they are assumed to be color emission probabilities and hence the same loss function can be used. The second is the KL-divergence loss, L_Z, defined between some latent prior $P(Z^t)$ and $Q(Z^t|h_{enc}^t) \sim \mathcal{N}(\mu^t, (\sigma^t)^2)$ and summed over all T timesteps. We assume prior $P(Z^t)$ as a standard normal distribution and thus, L_Z is given by (similar to [9]):

$$L_Z = \frac{1}{2}\left(\sum_{t=1}^{T} \mu_t^2 + \sigma_t^2 - \log \sigma_t^2\right) - T/2 \tag{15}$$

The final loss is the sum of the two losses, L_X and L_Z.

When caption-based alignment is included, the loss function for KL-divergence changes as the prior P can no longer be assumed to a standard normal distribution. The prior has to be conditioned on the sentence representation, s_t, and the resulting loss will hence be:

$$L_Z = \frac{\sum_{t=1}^{T} (\mu - \mu_{prior})^2 + \sigma^2}{\sigma_{prior}^2} - 2\log\frac{\sigma}{\sigma_{prior}} \tag{16}$$

where $\mu_{prior} = \tanh\left(W_\mu h_{dec}^{t-1}\right)$ and $\sigma_{prior} = \exp\left(\tanh\left(W_\sigma h_{dec}^{t-1}\right)\right)$, W_μ and W_σ being weight matrices that are learnt during training.

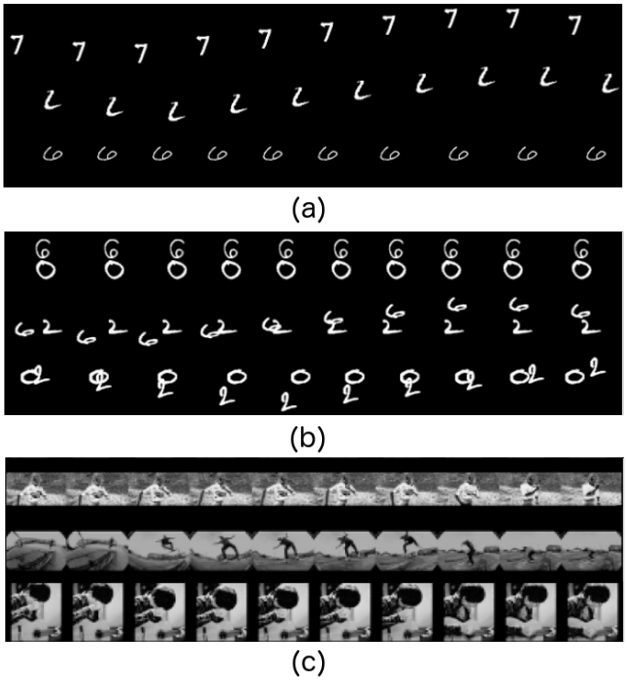

Figure 3. Examples from GIF datasets used in this work:. (a) Single-Digit Bouncing MNIST GIFs; (b) Two-Digit Bouncing MNIST GIFs; (c) TGIF Dataset [18]

3.4. Testing Phase

To generate a GIF using Sync-DRAW during testing, a sequence of z^ts are sampled from the learned latent distribution, $Q(Z^t|h_{enc}^t)$, for T timesteps and fed to the $LSTM_{dec}$ to generate the canvases which are accumulated to form the final GIF (as in Equation 11).

4. Experimental Results

We now describe the experimental results of Sync-DRAW. Considering the limited availability of GIF-based datasets, we studied the performance on the following datasets with increasing complexity: (i) Single-Digit

Bouncing MNIST GIFs (which has been used in similar earlier efforts [12][24][27]), (ii) Two-digit Bouncing MNIST GIFs; and (iii) TGIF, a recently developed GIF dataset [18]. Figure 3 shows sample GIFs from each of the datasets. The Bouncing-MNIST dataset was created with text captions (as described later in this section) to study the usefulness of Sync-DRAW in generating GIFs from captions.

For all our experiments, the number of timesteps T is taken to be 10. The choice of T is independent of the number of frames forming the GIFs. Further, the K for read and write attention parameters was chosen to be 8. Stochastic Gradient Descent with Adam [15] is used for training with initial learning rate as 10^{-3}, β_1 as 0.5 and β_2 as 0.999. Additionally, to avoid gradient from exploding, a threshold of 10.0 was used.[1]

4.1. Baseline Methodology

Considering this is the first work, to the best of our knowledge, on automatic GIF generation, there was no existing baseline methodology to compare against the performance of Sync-DRAW. Hence, as our baseline method, we designed another methodology to extend [9] to generate GIF images, by modeling the GIF as a spatio-temporal cuboid and adding a set of filterbanks F_Z^t of size $K \times N$ to operate over the temporal dimension (in addition to the two filterbanks in the spatial dimensions). We present these results for comparison later in this section.

4.2. Results on Single-Digit Bouncing MNIST

As in earlier work [12][24][27], we generated the Single-Digit Bouncing MNIST GIF dataset by having the MNIST handwritten digits move over time across the frames of the sequence, as shown in Figure 3a. Each GIF contains 10 frames each of size 64×64 with a single 28×28 digit either moving left-right or up-down. The initial position for the digit in each GIF is chosen randomly in order to increase variation among the samples. The training set contains $12,000$ such GIFs.

The results of Sync-DRAW on Single Digit MNIST GIFs are shown in Figures 4 and 6. The figures illustrate the usefulness of the proposed Sync-DRAW methodology (an R-VAE working at the video level and an attention mechanism working separately at the frame level) in gracefully generating the final GIF as a sequence of canvases. When compared to the baseline methodology results in Figure 5, the quality of the generated digits is clearly superior with the digits having well-defined structure and boundaries. An interesting observation to point out is that while each frame has its own attention mechanism in the proposed framework, it can be seen that the same area of the digit is being attended to at every timestep t in a synchronized manner,

[1]The codes and the relevant material can be found at https://github.com/syncdraw/Sync-DRAW

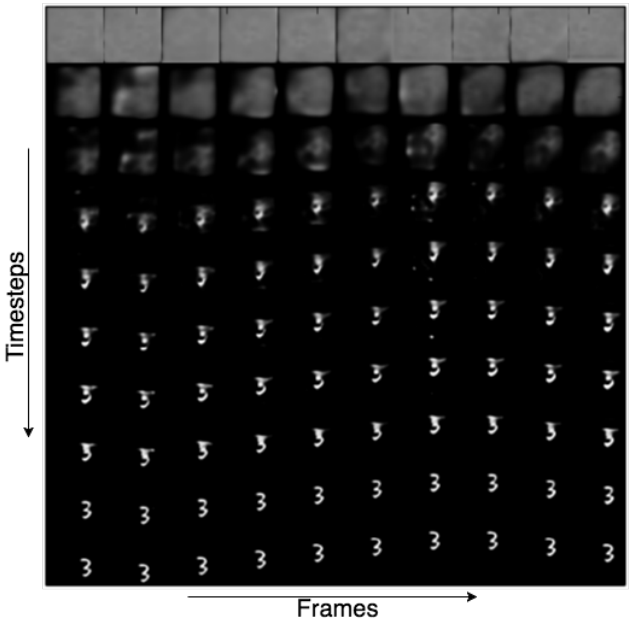

Figure 4. Sync-DRAW results on Single-Digit Bouncing MNIST GIFs (showing evolution of GIF over the timesteps). x-axis denotes the different frames of the GIF, while y-axis denotes the evolution of the GIF over the T time steps.

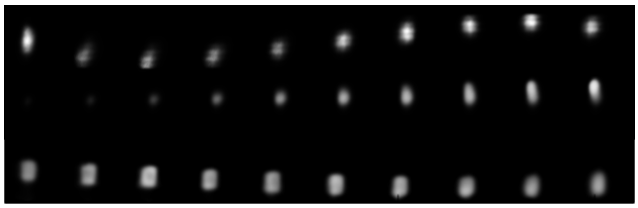

Figure 5. Results of the baseline method (Section 4.1) on Single-Digit bouncing MNIST GIFs

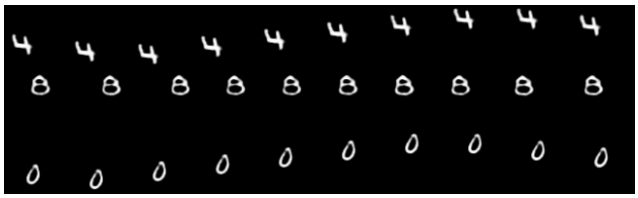

Figure 6. More results using Sync-DRAW results on Single-Digit Bouncing MNIST GIFs

Figure 8. Results of the baseline method (Section 4.1) on Two-Digit Bouncing MNIST GIFs

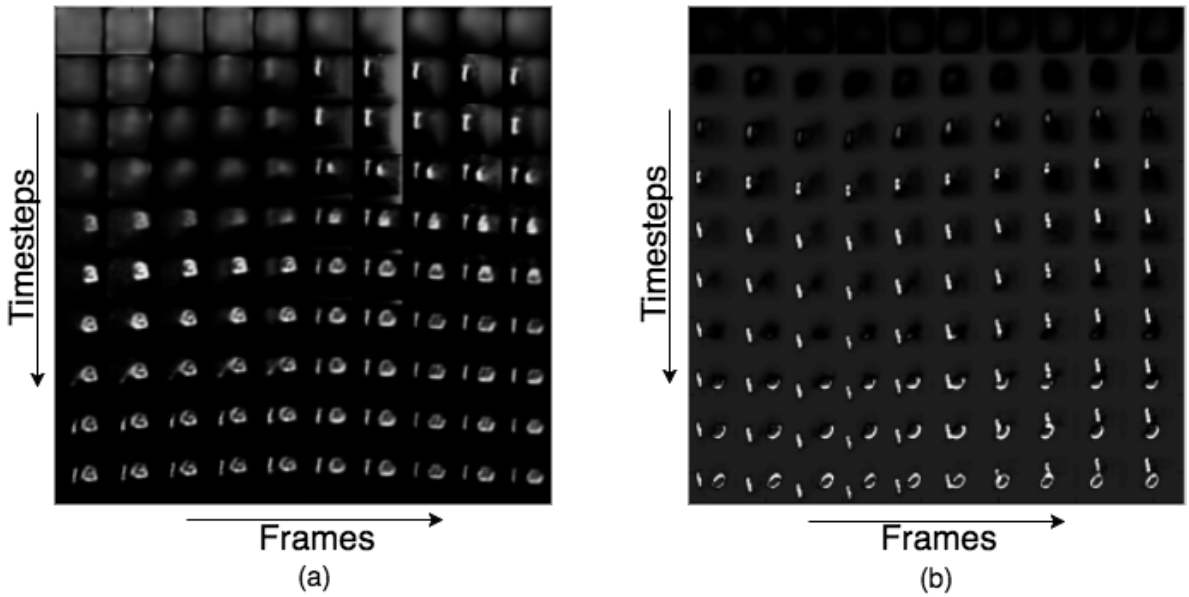

Figure 7. (a) Sync-DRAW results on Two-Digit Bouncing MNIST GIFs; (b) Sync-DRAW results on Two-Digit Bouncing MNIST GIFs when captions are included. Clearly, the additional information helps in better capturing the 'objectness' of the digits.

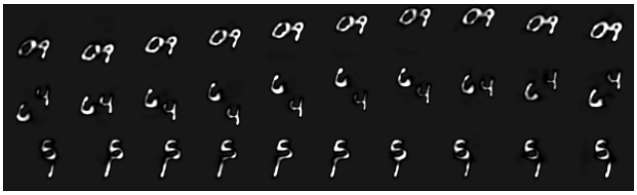

Figure 9. More results using Sync-DRAW results on Two-Digit Bouncing MNIST GIFs

even though they are present at different pixel locations in each frame, thus validating the proposed methodology. This observation is pivotal in emphasizing the need for the proposed hierarchical approach (local attention-global generation) to GIF generation. More results pertaining to single-digit Bouncing MNIST can be seen in Figure 6.

4.3. Results on Two-Digit Bouncing MNIST

Extending the Bouncing MNIST GIF dataset, we generated the two-digit Bouncing MNIST GIF dataset where two digits move independent of one another, either *up and down* or *left and right*, as shown in Figure 3b. The dataset consists of a training set of $12,000$ GIFs, with each GIF containing 10 frames of size 64×64 with two 28×28 digits. In order to ensure variability, the initial positions for both the digits were chosen randomly. In case of an overlap, the intensities are added, clipping the sum if it goes beyond 1[2].

The results of applying Sync-DRAW to this two-digit Bouncing MNIST dataset are shown in Figures 7a and 9. The figures show that Sync-DRAW attends to both the digits at every time step simultaneously. In the initial time steps, a single structure is formed which then breaks down to give the two digits in the subsequent time steps. We believe that this is the reason that though the digits that are generated have a well defined structure and a boundary, they are still not as clear as when compared to results on the Single-Digit Bouncing MNIST GIFs. We infer that there is a need for a "stimulus" for the attention mechanism to know that there are two digits that need to be attended to. This claim is substantiated in the subsequent sections where we include alignment with captions in the Sync-DRAW approach, giving the attention mechanism this required stimulus. Once again, the baseline method performs rather poorly as shown in Figure 8.

4.4. Results on TGIF Dataset

The recently released TGIF dataset [18] is perhaps the only publicly available GIF dataset, to the best of our knowledge, containing around 100,000 GIFs. As shown in Figure 3c , these GIFs vary significantly in complexity. We studied the performance of Sync-DRAW on a subset from the TGIF dataset, where each of the GIFs contained one person. The frames were converted to grayscale, and the intensity values were normalized to lie between 0 and 1. The frames are resized to 128×128 after appropriately padding with zeros. We then chose 10 random frames from each GIF (maintaining their sequence), which were then used for our experiments. Figure 10 shows some of the GIFs generated by Sync-DRAW. These results suggest that even for a highly complex input data, the proposed approach is able to under-

[2]All the Bouncing MNIST datasets are normalized to values lying between 0 and 1.

stand the scene and its intricacies and therefore, preserves the structural integrity of the objects across the frames to a good extent.

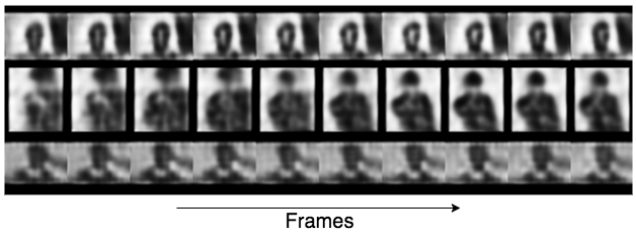

Figure 10. Sync-DRAW results on TGIF dataset: Three different GIFs generated from the trained model at test time (we don't show the evolution over the timesteps due to space constraints).

4.5. Generating GIFs from Captions

As discussed in Section 3, the proposed Sync-DRAW methodology can be easily extended to generate GIFs from captions, building on earlier work which was proposed to generate images from captions [19]. For every GIF in the MNIST GIFs dataset, a sentence caption describing the GIF was included in the dataset. For the Single-Digit Bouncing MNIST GIFs, the concomitant caption was of the form *'The digit 0 is moving left and right'* or *'The digit 9 is moving up and down'*. We restricted to these two motions in our dataset. Hence, for the single-digit dataset, we have 20 different combinations of captions. In order to challenge Sync-DRAW, we split our dataset in such a way that for all the digits, the training and the test sets contained different motions for the same digit, i.e. if a digit occurs with the motion involving *up and down* in the training set, the caption for the same digit with the motion *left and right* (which is not used for training) is used in the testing phase. Figure 11 shows some of the GIFs generated from captions present in the test set. It can be observed that even though the caption was not included in the training phase, Sync-DRAW is able to capture the implicit alignment between the caption, the digit and the movement fairly impressively.

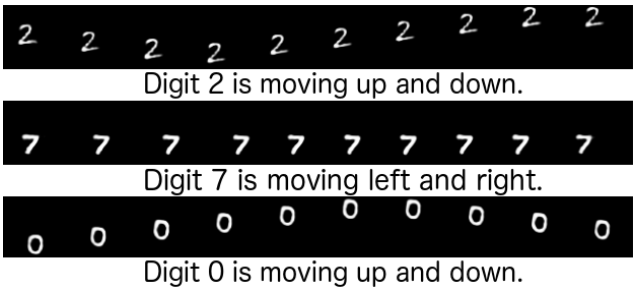

Figure 11. Sync-DRAW generates GIFs from just captions on the Single-Digit Bouncing MNIST: Results above were automatically generated by the trained model at test time.

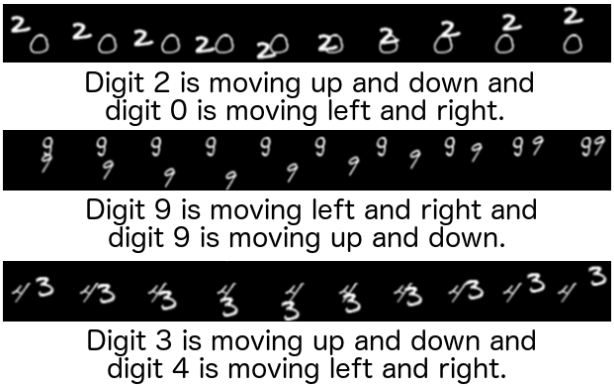

Figure 12. Sync-DRAW generates GIFs from just captions on the Two-Digit Bouncing MNIST: Results above were automatically generated by the trained model at test time.

We conducted similar experiments for the Two-Digit Bouncing MNIST GIFs, where the captions included the respective motion information of both the digits, for example *'The digit 0 is moving up and down, and digit 1 is moving left and right'*. Figure 12 shows the results on this dataset. Interestingly, in Figure 7b, we notice that when Sync-DRAW is conditioned on captions, the quality of the digits is automatically enhanced, as compared to the results in Section 4.3 (Figure 7a). These results give the indication that in the absence of captions (or additional stimuli), the attention mechanism in Sync-DRAW focuses on a small patch of a frame at a time, but possibly ignores the presence of different objects in the scene and visualizes the whole frame as one entity. However, by introducing the alignment w.r.t. captions, the attention mechanism receives the much needed "stimulus" to differentiate between the different objects and thereby cluster their generation, resulting in GIFs with better resolution. This is in concurrence with the very idea of an attention mechanism, which when guided by a stimulus, learns the spatio-temporal relationships in the GIF in a significantly better manner[3].

5. Conclusions and Future Work

This paper presents Sync-DRAW, a new approach to automatically generate animated GIFs using a Recurrent Variational Auto-Encoder and an attention mechanism, combined together in a hierarchical manner. We demonstrate Sync-DRAW's capability to generate increasingly complex GIFs starting from Single-Digit Bouncing MNIST to more complex GIFs from the TGIF dataset. We also demonstrate the capability of Sync-DRAW to be easily extended to generate GIFs for a given caption. The results show great promise in this approach. Our ongoing/future efforts include extending this work longer video sequences as well

[3]More results and analysis are included in the Supplementary Materials due to space constraints

as creating natural videos from captions.

References

[1] Y. Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. 2

[2] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015. 2

[3] O. Fabius and J. R. van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014. 4

[4] K. Fragkiadaki, P. Arbeláez, P. Felsen, and J. Malik. Learning to segment moving objects in videos. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4083–4090. IEEE, 2015. 2

[5] F. Galasso, N. Shankar Nagaraja, T. Jimenez Cardenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3527–3534, 2013. 2

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. 1

[7] 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. 1, 2

[8] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 2

[9] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015. 1, 2, 3, 4, 5, 6

[10] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 2

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[12] S. E. Kahou, V. Michalski, and R. Memisevic. Ratm: Recurrent attentive tracking model. *arXiv preprint arXiv:1510.08660*, 2015. 6

[13] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 2

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 2

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 6

[16] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2013. 1, 2

[17] H. Larochelle and G. E. Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Advances in neural information processing systems*, pages 1243–1251, 2010. 2

[18] Y. Li, Y. Song, L. Cao, J. Tetreault, L. Goldberg, A. Jaimes, and J. Luo. TGIF: A New Dataset and Benchmark on Animated GIF Description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2, 5, 6, 7

[19] E. Mansimov, E. Parisotto, J. L. Ba, and R. Salakhutdinov. Generating images from captions with attention. *International Conference on Learning Representations*, 2016. 1, 2, 3, 4, 5, 8

[20] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1777–1784, 2013. 2

[21] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *International Conference on Learning Representations*, 2015. 2

[22] R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *AISTATS*, volume 1, page 3, 2009. 2

[23] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014. 2

[24] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *International Conference on Machine Learning (ICML)*, 2015. 1, 2, 6

[25] I. Sutskever, J. Martens, and G. E. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011. 2

[26] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016. 1

[27] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810, 2015. 6

[28] 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 *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057, 2015. 2