GU J Sci, Part C, 10(4): 1127-1142 (2022)

Gazi University

Journal of Science

PART C: DESIGN AND TECHNOLOGY

http://dergipark.gov.tr/gujsc

Attentive Sequential Auto-Encoding Towards Unsupervised Object-centric Scene Modeling

Yarkın Deniz ÇETİN[1] Ramazan Gökberk CİNBİŞ[2,*]

[1]*İ.D. Bilkent University, Faculty of Engineering, Department of Computer Engineering, Ankara, Turkey*

[2]*Middle East Technical University, Faculty of Engineering, Departmant of Computer Engineering, Ankara, Turkey*

Article Info

Research article
Received: 02.07.2022
Revision: 18.10.2022
Accepted: 15.11.2022

Keywords

Virtual MCA
Real MCA
Pulse Processing

Abstract

This paper describes an unsupervised sequential auto-encoding model targeting multi-object scenes. The proposed model uses an attention-based formulation, with reconstruction-driven losses. The main model relies on iteratively writing regions onto a canvas, in a differentiable manner. To enforce attention to objects and/or parts, the model uses a convolutional localization network, a region level bottleneck auto-encoder and a loss term that encourages reconstruction within a limited number of iterations. An extended version of the model incorporates a background modeling component that aims at handling scenes with complex backgrounds. The model is evaluated on two separate datasets: a synthetic dataset that is constructed by composing MNIST digit instances together, and the MS-COCO dataset. The model achieves high reconstruction ability on MNIST based scenes. The extended model shows promising results on the complex and challenging MS-COCO scenes.

1. INTRODUCTION

Generative image modeling is a fundamental problem in machine learning, with numerous potential applications in various domains, such as computational art generation, image editing, representation learning and unsupervised recognition. For this reason, a variety of models have been proposed over the recent years, such as Generative Adversarial Networks (GANs) [1-3], variational auto-encoders (VAEs) [4-5], Moment Matching Networks [6], Normalizing Flows [7-11], Diffusion Models [12-13,34-35] and auto-regressive models [36].

Recent progress in generative model architectures and formulations has led to the construction of high-fidelity and high-resolution generative image models, e.g., Liu et al. [14], Brock et al. [15], Karras et al. [16], Karnewar and Wang [17], Karras et al. [18], Gu et al. [31], Rombach et al. [32], Zhang et al. [33]. These models are known to yield particularly successful results in single-object cases, such face generation [37]. However, even with the state-of-the-art techniques, the contemporary models suffer from two shortcomings: (i) the best performing models rely on supervised training where image labels and/or annotations are provided to the model during training [19], (ii) the modeling of complex scenes, involving multiple objects is largely an unresolved problem [20].

In this context, we propose a sequential image auto-encoder that can be trained over multi-object scene examples without annotations. The main goal of the proposed model is to learn a re-constructive operator that discovers object parts, objects and/or distinctive regions in an unsupervised manner, such that a scene can be reconstructed in a limited number of generation steps.

The proposed approach aims to make a step towards building models that can ultimately learn the structure of the world, in an object-centric way, from unlabeled natural images, as a way to tackle the difficulty of modeling complex scenes. In particular, the long-term research goal is to develop modeling principles that

*Corresponding author, e-mail: gcinbis@ceng.metu.edu.tr
DOI: 10.29109/gujsc.1139701

allow handling complex scenes in an object-by-object manner so that contemporary models performing well in single-object domains can be made to work well in the complex ones.

A main observation that suggests the possibility of unsupervised learning of object-based sequential scene models is the fact that modeling complex scenes in an object and/or part driven can be of lower complexity, compared to learning unstructured scene models, especially under certain constraints. The main reason here is that the spatial arrangement of objects within a scene typically has much more variability compared to the variability within the spatial structure of the object parts. For example, in a street scene, cars, people, trees and buildings may appear in pretty much arbitrary positions within the limits of geometric layout, with various nearby occurrences of the parts of different objects. In contrast, cars (and most other objects) have a relatively more rigid structure that depicts far smaller spatial variability, especially when the pose-driven variations are ignored. Therefore, from a complexity point of view, object-aware models are likely to provide much more scalable approaches, as opposed to tackling complete scenes as complex spatial arrangements of low-level patches.

Based on these reasonings, we therefore aim to make a step towards building unsupervised models that handle scenes in an object-by-object manner. More specifically, we aim to learn sequential auto-encoding processes where each write operation focuses on a single object, or a part of it, as opposed to writing patches with mixed object semantics. Ultimately, the goal is to construct approaches that are naturally capable of modeling natural scenes through object compositions, towards overcoming the limitations of contemporary models.

A major inspiration for our work is the DRAW model [21]. The DRAW model uses a variational auto-encoder where both the encoder and decoder components are recurrent neural networks (RNNs). At each time step, DRAW uses the output of the decoder RNN to decide the next image area to be read and written. The encoder RNN is updated according to the read area, its output is transformed into a latent variable sample, and the decoder RNN is updated according to that sample information. The patch to be written on the canvas, i.e., the image being constructed, is also decided using the output of the decoder RNN. The original DRAW work does not directly aim at learning object-based image models, e.g., the proposed model uses 64 read & write operations on simple MNIST images, which contain one object per image [21].

Our approach similarly proposes a sequential auto-encoder. In contrast to DRAW, however, we explicitly target exploring the potential of learning an objects-driven auto-encoder. Therefore, other than embracing the fully-differentiable read and write operators, and the sequential canvas-writing framework of DRAW, our approach drastically differs from the original DRAW model to be able to enforce object-by-object reconstruction of the scenes. The main distinctions of our model can be summarized as follows. First, instead of relying on RNN outputs to decide the soft attention areas as in DRAW, we define a dedicated convolutional localization network that decides on the read and write region coordinates at each time step. Second, instead of relying on RNNs, we define a bottleneck-limited region auto-encoder that operates locally on the area soft-attended according to the localization network output. The bottleneck in the encoding dimensionality restricts the model's per-patch generative complexity capacity, therefore, effectively enforces the whole model to focus on individual objects or their parts, as opposed to learning a degenerate generation process, e.g. holistically autoencoding the full scene like a conventional auto-encoder. Third, we propose the mean-squared error over time (MSEOT) loss, which increasingly penalizes discrepancies between the canvas and the original image. MSEOT term effectively enforces the network to reconstruct the scene in a minimal number of steps, and therefore, encourages learning to attend to object or major part patches, instead of smaller fragments. Fourth, we introduce an additional patch loss that directly measures the autoencoding quality of individual write operations. Finally, towards handling models with complex backgrounds, we introduce the background model, which first generates the background "stuff" so that the sequential model can focus on generating foreground "things".

The details of the method are given in Section 2. The experimental observations, which demonstrate the potential of the proposed approach in learning object-driven auto-encoding processes, are presented and discussed in Section 3. Finally, the conclusive remarks are given in Section 4.

2. METHOD

In this section, we present the details of the proposed model. We first give an overview of the overall architecture in Section 2.1. We then explain the localization network, region auto-encoder and differentiable reader/writer in Section 2.2, 2.3 and 2.4, respectively. We explain the training details in Section 2.5. Finally, we present model extensions towards handling scenes with complex backgrounds in Section 2.6.

2.1. Main Architecture

The architecture is built in a sequential way to reconstruct the given scene, within a limited number of steps. At each step, the model localizes the region to be read, extracts a summary of the location, re-constructs the localized region and accumulates the region reconstructions by writing onto a canvas.

More formally, the model can be expressed in terms of a series of sub-models, operations, and (intermediate) variables. The canvas image, which is updated over the read-write iterations, is expressed by c. The status of the canvas at a particular iteration t, is given by c_t, which is normally empty at $t = 0$ unless initialized by a background model (Section 2.6). At the beginning of each iteration, we first compute the error image ξ_t between the input image x, and the current canvas:

$$\xi_t = x - c_t \tag{1}$$

The next step of the model is to localize the region, where the model will soft-attend in the current iteration, using the localization network:

$$\omega_t = Localize(\xi_t) \tag{2}$$

The output of the localization network (Section 2.2) provides the parameters ω of the soft attention reader-writer (Section 2.4).

The next step is to read and encode the current region of interest in the error image domain:

$$\rho_t, f_t = Read(\xi_t, \omega_t) \tag{3}$$

$$z_t = Encoder(\rho_t) \tag{4}$$

The reader takes the error image ξ_t and the coordinates ω_t, and returns the read patch ρ_t and region reading filters f_t, which are re-used later in the writing stage. The encoder takes the read patch and returns the patch encoding z_t.

Once the localization, reading, and encoding steps are completed, the decoder sub-module takes the encoding z_t and synthesizes the region image to be written onto the canvas:

$$r_t = Decoder(z_t) \tag{5}$$

where r_t is the reconstruction output. The canvas is updated additively via the writing operation:

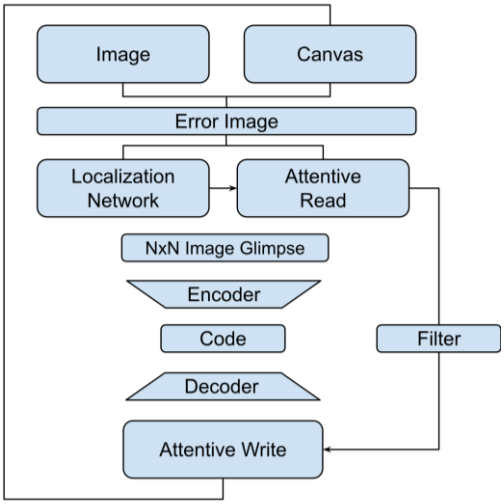

Figure 1. *The proposed sequential generation approach. The model iteratively attends to an image region, reconstructs the corresponding patch, and writes onto the canvas. All steps and modules are learned in an unsupervised manner.*

$$c_{t+1} = c_t + Write(r_t, f_t) \tag{6}$$

It is worth nothing that the same filters created at the reading stage are re-used here. A summary of all these steps is summarized in the diagram given in Figure 1. Overall, the model aims to reconstruct a given scene by attentively reading, reconstructing and writing patches onto a canvas.

The model's temporal depth is specified by the number of iteration (T), a hyper-parameter that describes the number of steps it is allowed to make to reconstruct the image. While this is normally a hyper-parameter, in Section 2.5 we describe a technique that aims to enforce the network to learn reconstruction with a limited number of iterations so as to learn larger parts or objects, instead of non-semantic low-level patches.

The following subsections explain the details of the aforementioned submodules and operations.

2.2. Localization Network

The localization network takes a given error image ξ_t at each iteration t, and returns the parameters for the patch reading operator. For this purpose, we use a convolutional neural network (CNN) that aims to predict the image region that needs to be locally updated. The network, as shown in Figure 2, is composed of a series of convolution and max pooling operations, with ReLU activations. While the convolutional layers are aimed at coarsely preserving the spatial structure of the input, max pooling operations aim to progressively select a particular image region.

The final layer of the network takes the vectorized convolutional activations, transforms them via a fully connected layer and outputs the 5 parameters required by the attentive read/write operators, as explained later in Section 2.4.

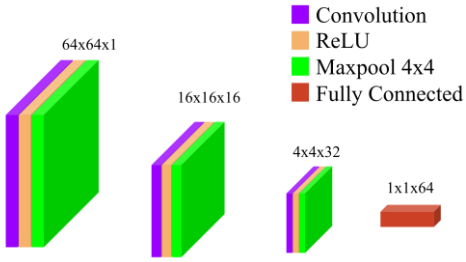

Figure 2. *The localization network used in the main experiments. This particular network shown is designed for 64x64 input images. The architecture can easily be adapted to other input resolutions.*

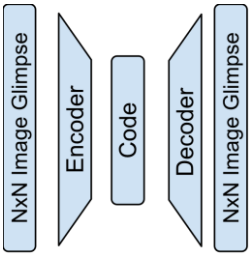

Figure 3. *Auto-encoder network with the bottleneck layer in the middle for reconstructing local image regions.*

2.3. Region Auto-Encoder

At each step, the localized region is encoded and reconstructed via a network with restricted encoding capacity, which we call region auto-encoder. The overall structure is visualized in Figure 3: the module takes a localized image region, encodes into a limited dimensionality code and tries to reconstruct the region of the same size as in input.

The bottleneck dimensionality can affect the learned region patterns. Larger dimensionalities can lead to localization of large regions, while the smaller ones can cause learning spatially too small patterns.

The region auto-encoder architecture used in our experiments is defined in Table 1. The encoder starts with three convolutional layers. The convolutional features are flattened and transformed via two consecutive fully connected layers. The bottleneck dimensionality is set to 1024 based on preliminary experiments. The encoded vector is approximately transformed back to the region via three consecutive transposed convolution layers.

2.4 Soft-Attention Reader-Writer

For attentive image reading and canvas writing operations, we utilize the kernel-based formulation used in Gregor et al. [21]. In this approach, $N \times N$ grids of Gaussian kernels are used for reading & writing image regions. The grid is defined by the stride parameter δ and the overall grid center coordinates (g_X, g_Y). The center μ of the (i, j)-th grid cell is defined as in Eq. 7, and Eq. 8:

Table 1. *Architecture of the region auto-encoder.*

Autoencoder Network	
Layer	Output Size
Convolution 1	16 x 16 x 16
Convolution 2	32 x 8 x 8
Convolution 3	64 x 4 x 4
Fully Connected	64, 128, 256, 512
Fully Connected	1024
Transpose Conv. 1	32 x 8 x 8
Transpose Conv. 2	16 x 16 x 16
Transpose Conv. 3	1 x 32 x 32

$$\mu_X^i = g_X + \left(i - \frac{N}{2} - 0.5\right)\delta \tag{7}$$

$$\mu_Y^j = g_Y + \left(j - \frac{N}{2} - 0.5\right)\delta \tag{8}$$

Given the grid configuration specified by all μ_x, μ_y values and the kernel variance parameter σ, a set of Gaussian kernels are formed. The gaussian kernels corresponding to the (i, j)-th grid cell using the corresponding grid center μ and σ values are defined as follows:

$$F_x(i, a) = \frac{1}{Z_X} exp\left(-\frac{\left(a - \mu_X^i\right)^2}{2\sigma^2}\right), \tag{9}$$

$$F_y(j, b) = \frac{1}{Z_Y} exp\left(-\frac{\left(b - \mu_Y^j\right)^2}{2\sigma^2}\right), \tag{10}$$

where F_x and F_y are the Gaussian kernel functions in the x and y directions, respectively. More specifically, $F_x[i, a]$ gives the reading (or writing) weight of pixels with $x = a$, for all cells in the i-th row of the grid. Similarly, $F_y[j, b]$ gives the reading (or writing) weight of pixels with $y = b$, for all cells in the j-th column of the grid. Z_X and Z_Y are normalization constants such that the kernel weights sum to one.

A final input parameter γ is used for scaling the read and write operation intensity values. The reading is achieved simply by a bilinear operation. For an input x, the value read for the grid coordinate (i, j) is given by:

$$\sum_a^H \sum_b^W \gamma F_x(i, a) F_y(j, b) x(a, b) \tag{11}$$

where $x(a, b)$ is the image pixel at the coordinates (a, b), and the summation runs over all image pixels. Similarly, the write operation produces the image to be added to the canvas according to the following formula, describing the write value for pixel at the grid coordinate (i, j):

$$\sum_{a}^{H}\sum_{b}^{W}\frac{1}{\gamma}F_x(i,a)F_y(j,b)x(a,b) \tag{12}$$

In color images, we apply the same operations to all channels individually.

Examples of kernels with different stride δ and the number of kernel settings are shown in Figure 4. The first column shows the original images. The second one shows outputs of patch read operations via Gaussian kernels. The third column shows the images rewritten onto the canvas, and the last one shows the filters and their respective densities over the canvas. Going from up to down, the number Gaussian kernels (N) and their σ parameters are as follows. First row: $N = 21, \sigma = 0.1$. The second row: $N = 42, \sigma = 0.1$. The third row: $N = 21, \sigma = 0.01$. The last row: $N = 42, \sigma = 0.01$. Notice that higher σ values yield more blurry readings, and the larger number of kernels provide higher resolution operations. Therefore, in practice, the stride δ parameter controls the size of the region that is read by the model, and by increasing δ, larger areas can be read at the expense of reduced resolution. The resolution, in general, can be adjusted by decreasing or increasing the number of kernels.

The Gaussian kernel based read/write scheme has two main advantages. First, the operation is fully differentiable, therefore, allows end-to-end gradient-based training. Second, using only 5 parameters, i.e., g_X, g_Y, δ, σ and γ, the read and write operation is determined, which simplifies the task of the localization model.

2.5. Training

Arguably the most commonly used loss function for training auto-encoders is the mean squared error (MSE) loss, which is defined as follows:

$$\frac{1}{HW}\sum_{i}^{H}\sum_{j}^{W}\left(x_{ij}-c_{ij}\right)^2 \tag{13}$$

While this is a sensible loss for measuring the overall reconstruction quality, a problem that we have observed is that the network may end up writing too many small patches especially when the number of read/write iterations is set to high. Similarly, when the iteration count is too small, the network then struggles learning the scene structures as it is not allowed to make sufficiently many changes on the canvas for reconstruction purposes.

A major difficulty in setting the optimal number of read/write iterations is the fact that complexity of the scenes can change significantly across images. A parameter that is too small for one scene can be too high for another one. Based on these observations, we opt to choose a relatively large number of iterations while penalizing more heavily the canvas errors towards the end. More specifically, we propose the *Mean Squared Error Over Time* (MSEOT) loss, which has the following form:

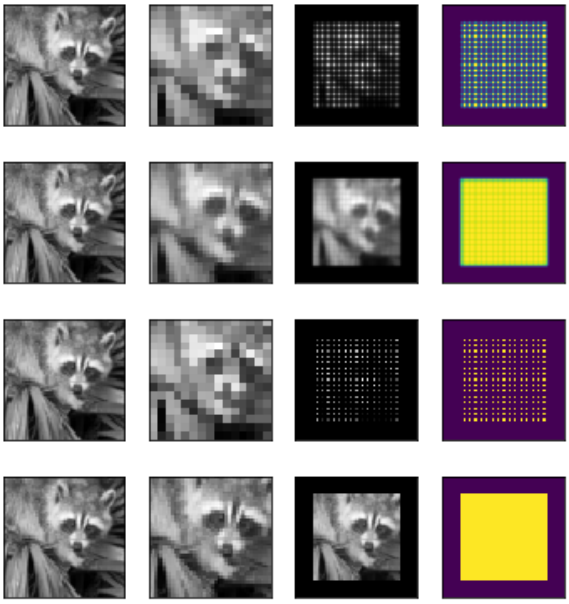

Figure 4. *Visualization of soft attention readers with varying number of kernel and stride parameters.*

$$L_{MSEOT} = \sum_{t=1}^{T} ||x - c_t||^2 (t-1)^2 \tag{14}$$

This loss term encourages the network to complete the scene in early iterations. In this manner, whenever reconstruction is possible with a few write operations, e.g., in scenes with few objects in them, the network will be rewarded for early reconstruction. However, where that is not possible, e.g., in relatively more complex scenes, the network may continue to write patches towards maintaining a smaller reconstruction loss at least towards the final iterations.

The proposed MSEOT loss measures the loss over the whole image. However, it is also preferable to write accurate regions so that each write corresponds to the whole object (or part) as opposed to an intensity-scaled version of it. To realize this loss term (L_{region}), at each iteration, we apply the read operation to the original image, and compute the MSE loss between the write output and the original patch.

Our final loss blends all these losses together, which are computed on a per-iteration basis:

$$L_{total} = \alpha_{region} L_{region} + \alpha_{MSEOT} L_{MSEOT} \tag{15}$$

where α_{region} and α_{MSEOT} are the weighting terms. The accumulation of these losses over the iterations are used to drive the network training.

2.6. Handling Scenes with Complex Backgrounds

This section explains the background module that we introduce to the model in order to adapt it from scenes with simple backgrounds (i.e., compositions of binary digits) to more complex, realistic scenes as in the MS-COCO [22] dataset. We also greatly increase the localization network's complexity to improve its modeling capacity and utilize progressive training, both of which are explained at the end of section.

Background modeling. One of the problems with the attentive model is the limited number of write operations the model has in its disposal to reconstruct a given image. On top of that, forcing the model to behave in a one-object-at-a-time behavior via MSEOT and all the explained design choices creates a great difficulty in scenes with complex backgrounds.

Since the non-trivial backgrounds are generally composed of *stuff*, which is typically a combination of countless objects and/or textures of trees, grasses, etc. Such objects are not compatible with the *one object (part) at a time* auto-encoding scheme.

In fact, when the original model is trained over images with non-trivial background, the model allocates a great number of write operations to writing stuff fragments, since the background is generally the largest "object" and typically constitutes the largest part of the reconstruction error.

This behavior leads to undesired one to one copying of unstructured patches, without showing any scene structure learning behavior. To prevent learning such degenerate models, we introduce a *background network*. Basically, the background network $B(x)$ is a simple, image-level bottleneck autoencoder that aims to capture the coarse scene structure.

This background network reduces the initial error of the canvas, allowing the model to start with a coarse scene, as opposed to an empty canvas. With the help of this network, the model starts to show promising results MS-COCO, which are later discussed in Section 3.

Table 2 presents the architectural details of the background network. Its encoder consists of a simple convolutional layer, followed by a fully connected layer. The decoding starts with another fully connected layer, followed by a transposed convolutional layer.

Improved localization network. We observe that the capacity of the localization network needs to be increased significantly to be able to produce usable region estimates in complex scenes. The architecture of the improved localization is presented in Table 3. As it can be seen from the table, the network mainly consists of a series of convolution, batch normalization and max pooling triplets. All activations in the network use Leaky ReLU activations. The resulting embeddings are then converted to 5 read/write parameters using a final fully connected layer.

Progressive training. Finally, we observe that setting the number of read/write iterations is more problematic in the case of real-world images with complex backgrounds, even with the MSEOT loss. In our preliminary experiments, we have observed that when the model size is kept fixed, the model tends to struggle in handling the complexity of the domain due to insufficient complexity, or quickly overfits to a degenerate autoencoder that does write operations poorly aligned with the scene structure. As a partial solution, we observe that starting with fewer read/write iterations and increasing it at later iterations yields a more stable training process.

Table 2. *Architecture of the background network.*

Background Network	
Layer	**Output Size**
Convolution 1	4 x 32 x 32
Fully Connected	16
Fully Connected	4096
Transpose Conv. 1	1 x 64 x 64

Table 3. *Architecture of the localization network used in MS-COCO experiments.*

Autoencoder Network	
Layer	**Output Size**
Convolution 1_1	32 x 64 x 64
BatchNorm 1_1	32 x 64 x 64
MaxPool	32 x 32 x 32
Convolution 1_2	32 x 64 x 64
BatchNorm 1_2	32 x 64 x 64
Convolution 2_1	64 x 32 x 32
BatchNorm 2_1	64 x 32 x 32
MaxPool	64 x 16 x 16
Convolution 2_2	64 x 32 x 32
BatchNorm 2_2	64 x 32 x 32
MaxPool	64 x 16 x 16
Convolution 3_1	128 x 16 x 16
BatchNorm 3_1	128 x 16 x 16
MaxPool	128 x 8 x 8
Convolution 3_2	128 x 8 x 8
BatchNorm 3_2	128 x 8 x 8
MaxPool	128 x 4 x 4
Fully Connected	256

3. RESULTS AND DISCUSSION

This section first explains the two datasets that we use to empirically observe the potential of the presented model. We then present our main experimental results on the MNIST-Scenes dataset, and the exploratory results on the MS-COCO dataset.

3.1. Datasets

For our main experiments to empirically observe the potential of the proposed model, we generate a new MNIST-based dataset, which we call the *MNIST Scenes* dataset. This dataset consists of scenes with multiple MNIST digits, therefore, it provides an experimental setup with simple shapes where the autoencoder can be trained in a reasonable amount of time. The dataset contains 10000 and 2000 randomly generated images for training and testing, respectively. The dataset is artificially generated using a Python script.

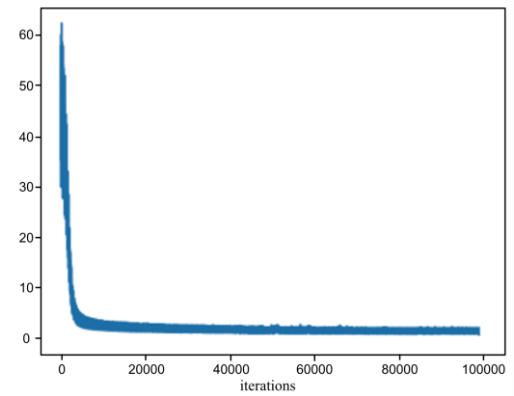

Figure 5. *Test reconstruction loss over training iterations on MNIST-Scenes.*

The second dataset that we use is the MS-COCO dataset [22], which contains greatly complex scenes that are typically composed of a large number of small and large objects, only some of which are annotated in the data set. To avoid computational difficulties, we resize the images to 64x64. In this resize operation, to preserve the aspect ratio of the objects, we first crop the largest possible square in each image and then, resize it to 64x64. We use 40000 training images and 4000 test images.

3.2. MNIST-Scenes Experiments

In the MNIST-Scenes experiments, our main goal is to explore the potential of the network to learn a part/object based scene model. The network is trained for 250 Epochs with a batch size of 25. The Gaussian kernel size selected for experiments is 25×25, roughly one Gaussian per pixel in the MNIST digits. While we find that the selection of Gaussian kernel size is not a critical choice, selecting too small values tends to cause learning of very small image fragments, as opposed to parts or objects. Similarly, selecting two high values tends to cause reading and writing regions that cover multiple objects, instead of individual ones.

Figure 5 shows the reconstruction loss measured on the test set, over the training iterations. As the loss is measured on images unused for model fitting, the decrease in the loss over time quantitatively shows the progress made by the model in terms of sequential autoencoding of novel scenes.

The experiments show high segmentation performance with objects with no textured background. However, when textured background is present, the network fails to localize correctly without using higher code sizes. This in turn creates overfitting with many object patches and breaks the object-by-object reconstruction behavior.

In Figure 6, a qualitative result is presented on a scene containing seven different digits. The first image on top-left shows the original image. The following ones, listed in row-major order, visualize the canvas after each write operation. The result shows that the model is able to locate and write individual objects, achieving the desired behavior.

Figure 6. *An example MNIST reconstruction result. The top-left image is the original image. The following ones show the canvas after each read/write iteration.*

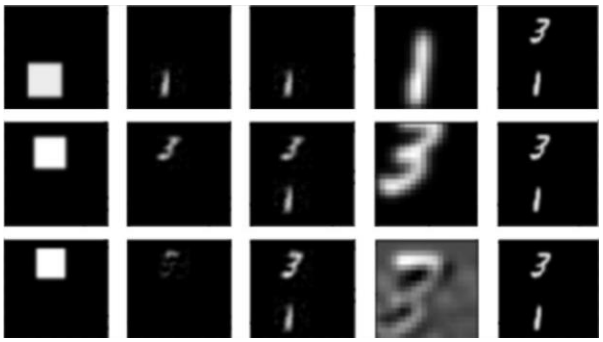

Figure 7. *An image reconstruction example on MNIST-Scenes. Time points to down. The leftmost part shows the filter weights on canvas. The second column shows the image patches written at that time step. The third row shows the cumulative state of the canvas. The fourth column shows the patches read by the filter. The last column shows the original image.*

To further illustrate the inner workings of the trained model, we illustrate the details of the attentive read and write operations in Figure 7. In the figure, each row shows a time step, starting from the initial one. The first column shows the attended regions, denoted by the filter weights. The second and third columns show the region to be written onto the canvas and the updated canvas status, respectively. The fourth column shows the patch that is read as the source of these write operations. Finally, the last column shows the original image. From the steps, it can be observed that the model first reads and reconstructs the digit "1" alone, and then the digit "3". The following iteration acts mainly as an improvement step that adds details for the digit "3" onto the canvas.

An interesting question is whether the network is able to implicitly localize the objects. While the network lacks an explicit encoding of object locations due to the infinite-support differentiable read/write formulation, we can still look into the localization network's behavior by analyzing the embeddings provided by the localization network, which is aims to define the areas of attention for the sequential reconstruction purposes. For this purpose, we generate a sequence of images where the number “36” is smoothly moved to form a lower triangle, spanning the regions close to the lower border, the right border and the diagonal areas. We coarsely categorize these images in terms of the location of the number: on the diagonal (*diag*), on the right (*right*) and in the down parts of the image (*lower*). Finally, we calculate the position encodings given by the localization network and compute their 2-dimensional t-SNE [23] embeddings.

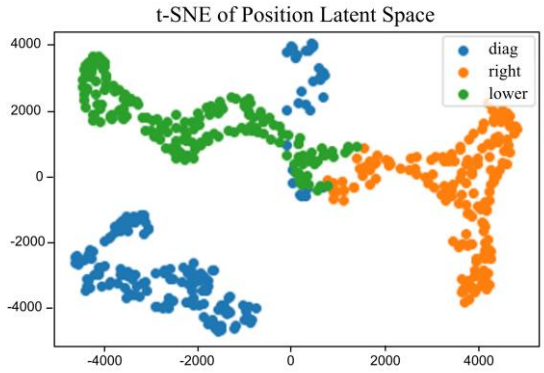

Figure 8. *Analysis of the latent space generated by the localization network. The t-SNE embeddings are grouped into colors based on the location of the digit in the provided input image. All embeddings are obtained based on the same image model.*

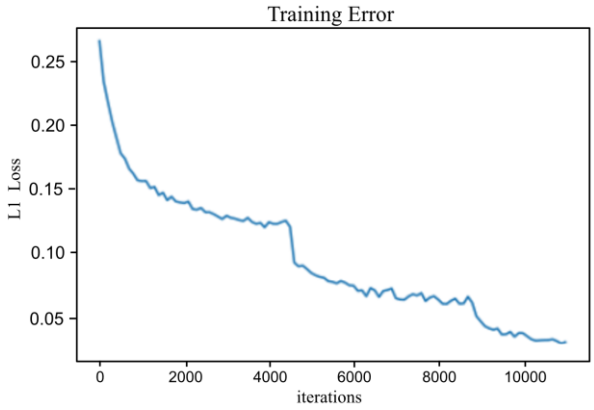

Figure 9. *The reconstruction error on the MS-COCO train set over the training iterations.*

The tSNE embeddings are presented in Figure 8. From the results, we observe that similarly located objects tend to yield similar tSNE embeddings, which suggests that the localization network is able to learn localizations of objects, at least to some degree. We also observe that diagonal positions have a more spreaded tSNE distribution, which is understandable as the diagonal spans a wide range of coordinates.

3.3. MS-COCO Experiments

The main goal of the MS-COCO experiments is to observe the possibility of extending the model to real-world scenes with complex backgrounds, and observe the difficulties therein. For this purpose, we monitor the ability of the model to *fit* to the train data, which is a difficulty on its own. For this purpose, we monitor the reconstruction loss over the training iterations. As it can be seen in Figure 9, the model converges slowly towards lower errors. We also observe the major error drops around iterations 5000 and 9000, following the additions of extra iterations. This observation suggests that the progressive training of the model helps the convergence.

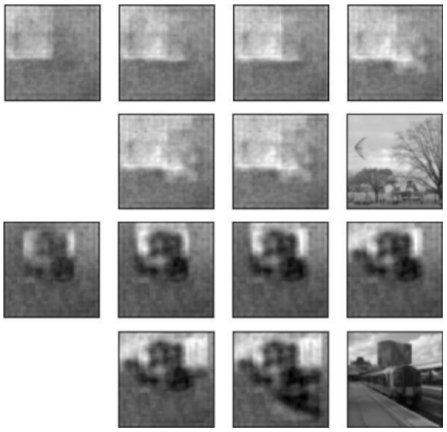

Figure 10. *Qualitative reconstruction results on MS-COCO.*

Two example qualitative results are given in Figure 10. In each example, we observe the change of the canvas starting from the one generated by the background model, over the read/write iterations. The very last image in each series shows the original one. In both cases, we observe that the model sequentially adds details to the scene. However, the clearly major difference between the final canvas states and the original images point out the difficulty of the task.

5. CONCLUSION

In this work, we have presented a fully-differentiable, end-to-end trained sequential autoencoding model for multi-object scenes. The model is trained in an unsupervised manner, including its localization network, as well as the read & write heads, purely driven by reconstruction quality based losses. At each iteration, the model soft-attends to a region in the image area, and updates a canvas based on its regional auto-encoding output. To enforce learning to reconstruct over major parts and objects, instead of a series of small patches, enforce the model to reconstruct the scene within a few steps via the Mean Squared Error Over Time loss. Similarly, to avoid attention onto too large or complex regions, we use a bottleneck auto-encoder at each attended region.

The experiments show that, in simple scenes, the model yields positive results with a tendency to reconstruct scenes to a large degree in an object-driven manner. While the real-world MS-COCO experiments, with the background model, also shows promising results, there is a large gap between the reconstructions and the actual scenes.

As a future work, the progress towards learning object-centric scene models can also naturally yield novel approaches to the *unsupervised image segmentation* problem, where the goal is to group pixels into semantically coherent clusters. As shown in our experimental results, the sequential generative scene model can discover objects as writing blocks. This information, therefore, has the potential of providing valuable image segmentation information, in a way fundamentally differs from the traditional techniques such as pixel-to-pixel similarity-driven graph cuts [24-25], mean-shift clustering [26], boundary-detection [27], bottom-up multi-scale hierarchical image segmentation [28] or deep network based on similar foundations [29]. An important research direction is to replace the bottleneck auto-encoder with the recent variants of deep generative models, such as GANs [1,30], VAEs [4-5,31] or the Diffusion Models [12-13].

REFERENCES

[1] Goodfellow I. J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., & Bengio Y. (2014). Generative Adversarial Networks. Advances in Neural Information Processing Systems.

[2] Arjovsky M., Chintala S., & Bottou L. (2017). Wasserstein GAN. ArXiv:1701.07875 [Cs, Stat].

[3] Karras T., Laine S., & Aila T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. Proc. CVPR.

[4] Kingma Diederik P., & Welling, M. (2014). Auto-Encoding Variational Bayes. International Conference on Learning Representations.

[5] Rezende D. J., Mohamed S., & Wierstra D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. ArXiv:1401.4082.

[6] Li Y., Swersky K., & Zemel R. (2015). Generative Moment Matching Networks. PMLR.

[7] Dinh L., Sohl-Dickstein J., & Bengio S. (2016). Density estimation using Real NVP.

[8] Kobyzev I., Prince S. J., & Brubaker M. A. (2020). Normalizing flows: An introduction and review of current methods. IEEE transactions on pattern analysis and machine intelligence, 43(11), 3964-3979.

[9] Kingma Durk P. & Dhariwal P. (2018). Glow: Generative Flow with Invertible 1x1 Convolutions. In Advances in Neural Information Processing Systems 31 (pp. 10215–10224).

[10] Behrmann J., Grathwohl W., Chen R. T. Q., Duvenaud, D., & Jacobsen, J.-H. (2019). Invertible Residual Networks. ArXiv:1811.00995 [Cs, Stat].

[11] Köhler J., Klein L. & Noé F. (2020). Equivariant Flows: exact likelihood generative learning for symmetric densities. ArXiv:2006.02425 [Physics, Stat].

[12] San-Roman R., Nachmani E., & Wolf L. (2021). Noise estimation for generative diffusion models. ArXiv Preprint ArXiv:2104.02600.

[13] Huang C.-W., Lim J. H., & Courville A. C. (2021). A variational perspective on diffusion-based generative models and score matching. Advances in Neural Information Processing Systems, 34.

[14] Liu K., Tang W., Zhou F., & Qiu G. (2019, October). Spectral Regularization for Combating Mode Collapse in GANs. ICCV.

[15] Brock A., Donahue J., & Simonyan K. (2018). Large Scale GAN Training for High Fidelity Natural Image Synthesis. ArXiv:1809.11096 [Cs, Stat].

[16] Karras T., Laine S., Aittala M., Hellsten J., Lehtinen J., & Aila T. (2020). Analyzing and Improving the Image Quality of StyleGAN. Proc. CVPR.

[17] Karnewar A., & Wang O. (2020). MSG-GAN: Multi-Scale Gradients for Generative Adversarial Networks. CVPR.

[18] Karras T., Aittala M., Laine S., Härkönen E., Hellsten J., Lehtinen J., & Aila T. (2021). Alias-free generative adversarial networks. Advances in Neural Information Processing Systems, 34.

[19] Casanova A., Careil M., Verbeek J., Drozdzal M., & Romero-Soriano, A. (2021, November). Instance-Conditioned GAN. NeurIPS.

[20] Zhang Y., Ling H., Gao J., Yin K., Lafleche J.-F., Barriuso A., Torralba A., & Fidler S. (2021). DatasetGAN: Efficient Labeled Data Factory with Minimal Human Effort. ArXiv:2104.06490 [Cs].

[21] Gregor K., Danihelka I., Graves A., Rezende D. J., & Wierstra D. (2015). DRAW: A Recurrent Neural Network For Image Generation. ArXiv:1502.04623 [Cs].

[22] Lin T.-Y., Maire M., Belongie S., Hays J., Perona P., Ramanan D., Dollár P., & Zitnick C. L. (2014). Microsoft COCO: Common Objects in Context. 740–755.

[23] van der Maaten L., & Hinton G. (2008). Visualizing Data using t-SNE . Journal of Machine Learning Research, 9, 2579–2605.

[24] Felzenszwalb P. F., & Huttenlocher D. P. (2004). Efficient graph-based image segmentation. International Journal of Computer Vision, 59(2), 167–181.

[25] Cour T., Benezit F., & Shi J. (2005). Spectral segmentation with multiscale graph decomposition. IEEE Conference on Computer Vision and Pattern Recognition, 2, 1124–1131.

[26] Comaniciu D., & Meer P. (2002). Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5), 603–619.

[27] Arbelaez P., Maire M., Fowlkes C., & Malik J. (2009). From contours to regions: An empirical evaluation. IEEE Conference on Computer Vision and Pattern Recognition, 2294–2301.

[28] Pont-Tuset J., Arbelaez P., Barron J. T., Marques F., & Malik J. (2016). Multiscale combinatorial grouping for image segmentation and object proposal generation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(1), 128–140.

[29] Xia, X. & Kulis B. (2017). W-net: A deep model for fully unsupervised image segmentation. ArXiv Preprint ArXiv:1711.08506.

[30] Karras T., Aila T., Laine S., & Lehtinen J. (2017). Progressive growing of GANs for improved quality, stability, and variation. Proc. Int. Conf. Learn. Represent.

[31] Gu, S., Chen, D., Bao, J., Wen, F., Zhang, B., Chen, D., ... & Guo, B. (2022). Vector quantized diffusion model for text-to-image synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10696-10706).

[32] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10684-10695).

[33] Zhang, B., Gu, S., Zhang, B., Bao, J., Chen, D., Wen, F., ... & Guo, B. (2022). Styleswin: Transformer-based gan for high-resolution image generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11304-11314).

[34] Preechakul, K., Chatthee, N., Wizadwongsa, S., & Suwajanakorn, S. (2022). Diffusion autoencoders: Toward a meaningful and decodable representation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10619-10629).

[35] Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., & Aberman, K. (2022). Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. arXiv preprint arXiv:2208.12242.

[36] Lee, D., Kim, C., Kim, S., Cho, M., & Han, W. S. (2022). Autoregressive Image Generation using Residual Quantization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11523-11532).

[37] Kammoun, Amina, Rim Slama, Hedi Tabia, Tarek Ouni, and Mohmed Abid. "Generative Adversarial Networks for face generation: A survey." ACM Computing Surveys (CSUR) (2022).