# GENERATING MULTIPLE OBJECTS AT SPATIALLY DISTINCT LOCATIONS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Recent improvements to Generative Adversarial Networks (GANs) have made it possible to generate realistic images in high resolution based on natural language descriptions such as image captions. However, fine-grained control of the image layout, i.e. where in the image specific objects should be located, is still difficult to achieve. We introduce a new approach which allows us to control the location of arbitrarily many objects within an image by adding an object pathway to both the generator and the discriminator. Our approach does not need a detailed semantic layout but only bounding boxes and the respective labels of the desired objects are needed. The object pathway focuses solely on the individual objects and is iteratively applied at the locations specified by the bounding boxes. The global pathway focuses on the image background and the general image layout. We perform experiments on the Multi-MNIST, CLEVR, and the more complex MS-COCO data set. Our experiments show that through the use of the object pathway we can control object locations within images and can model complex scenes with multiple objects at various locations. We further show that the object pathway focuses on the individual objects and learns features relevant for these, while the global pathway focuses on global image characteristics and the image background.

## INTRODUCTION

Understanding how to learn powerful representations from complex distributions is the intriguing goal behind adversarial training on image data. While recent advances have enabled us to generate high-resolution images with Generative Adversarial Networks (GANs), currently most GAN models still focus on modeling images that either contain only one centralized object (e.g. faces (CelebA), objects (ImageNet), birds (CUB-200), flowers (Oxford-102), etc.) or on images from one specific domain (e.g. LSUN bedrooms, LSUN churches, etc.). This means that, overall, the variance between images used for training GANs tends to be low (Raj et al., 2017). However, many real-life images contain multiple distinct objects at different locations within the image and with different relations to each other. This is for example visible in the MS-COCO data set (Lin et al., 2014), which consists of images of different objects at different locations within one image. In order to model images with these complex relationships, we need models that can model images containing multiple objects at distinct locations. To achieve this, we need control over what kind of objects are generated (e.g. persons, animals, objects, etc.), the location, and the size of these objects. This is a much more challenging task than generating a single object in the center of an image.

Current work (Karacan et al., 2016; Johnson et al., 2018; Hong et al., 2018b; Wang et al., 2018) has often approached this challenge by using a semantic layout as additional conditional input. While this can be successful in controlling the image layout and object placement, it also places a high burden on the generating process since a complete scene layout must be obtained first. We propose a model that does not require a full semantic layout of the scene, but instead only requires the desired object locations and identities. One part of our model, called the global pathway, is responsible for generating the general layout of the complete image, while a second path, the object pathway, is used to explicitly generate the features of different objects based on the relevant object label and location.

The generator gets as input a natural language description of the scene (if existent), the locations and labels of the various objects within the scene, and a random noise vector. The global pathway uses this to create a scene layout encoding which describes high-level features of the scene and

generates a feature representation for the whole scene from this. The object pathway generates a feature representation of a given object at a location described by the respective bounding box. It is then applied iteratively over the scene at the locations specified by the individual bounding boxes. We then concatenate the feature representations of the global and the object pathway and use this to generate the final image.

The discriminator, which also consists of a global and object pathway, gets as input the image, the bounding boxes and their respective object labels, and the textual description. The global pathway is then applied to the whole image and obtains a feature representation of the global image features. In parallel, the object pathway focuses only on the areas described by the bounding boxes and the respective object labels and obtains feature representations of these specific locations. Again, the outputs of both the global and the object pathway are merged and the discriminator is trained to distinguish between real and generated images.

In contrast to previous work we do not generate a scene layout of the whole scene but only focus on relevant objects which are placed at the specified locations, while the global consistency of the image is the responsibility of the other part of our model. To summarize our model and contributions: 1) We propose a GAN model that enables us to control the layout of a scene without the use of a scene layout. 2) Through the use of an object pathway which is responsible for learning features of different object categories we gain control over the identity and location of arbitrarily many objects within a scene. 3) The discriminator judges not only if the image is realistic and aligned to the natural language description, but also whether the specified objects are at the given locations and of the correct object category. 4) We show that the object pathway does indeed learn relevant features for the different objects, while the global pathway focuses on general image features and the background.

## RELATED WORK

Having more control over the general image layout has gained more and more attention in recent work. It has been shown that this can lead to a higher quality of images (Reed et al., 2016a; Hong et al., 2018b) and is also an important requirement for semantic image manipulation (Hong et al., 2018a; Wang et al., 2018). Approaches that try to exert some control over the image layout utilize Generative Adversarial Nets (Goodfellow et al., 2014), Refinement Networks (e.g. Chen & Koltun (2017); Xu et al. (2018a)), recurrent attention-based models (e.g. Mansimov et al. (2016)), autoregressive models (e.g. Reed et al. (2016c)), and even memory networks supplying the image generation process with previously extracted image features (Zhang et al., 2018b).

One way to exert control over the image layout is by using natural language descriptions of the image, e.g. image captions. Reed et al. (2016b), Zhang et al. (2018a), Sharma et al. (2018), and Xu et al. (2018b) show that it is possible to use image captions to generate images adhering to the described scene. However, these approaches are trained only with images and their respective captions and it is not possible to specifically control the layout or placement of specific objects within the image. Several approaches suggested using a semantic layout of the image, generated from the image caption, to gain more fine-grained control over the final image. Karacan et al. (2016), Johnson et al. (2018), and Wang et al. (2018) use a scene layout to generate images in which given objects are drawn within their specified segments based on the generated scene layout. Hong et al. (2018b) use the image caption to generate bounding boxes of specific objects within the image and predict the object's shape within each bounding box. This is further extended by Hong et al. (2018a) by making it possible to manipulate images on a semantic level. While these approaches offer a more detailed control over the image layout they heavily rely on a semantic scene layout for the image generating process, often implying complex preprocessing steps in which the scene layout is constructed.

The two approaches most closely related to ours are by Reed et al. (2016a) and Raj et al. (2017). Raj et al. (2017) introduce a model that consists of individual "blocks" which are responsible for different object characteristics (e.g. color, shape, etc.). However, their approach was only tested on the synthetic SHAPES dataset (Andreas et al., 2016), which has only comparatively low variability and no image captions. Reed et al. (2016b), on the other hand, condition both the generator and the discriminator on either a bounding box containing the object or keypoints describing the object's shape. Through this, they are able to generate high-quality natural images based on image captions. However, the used images are still of relatively low variability (e.g. birds (Wah et al., 2011)) and only contain one object, usually located in the center of the image.

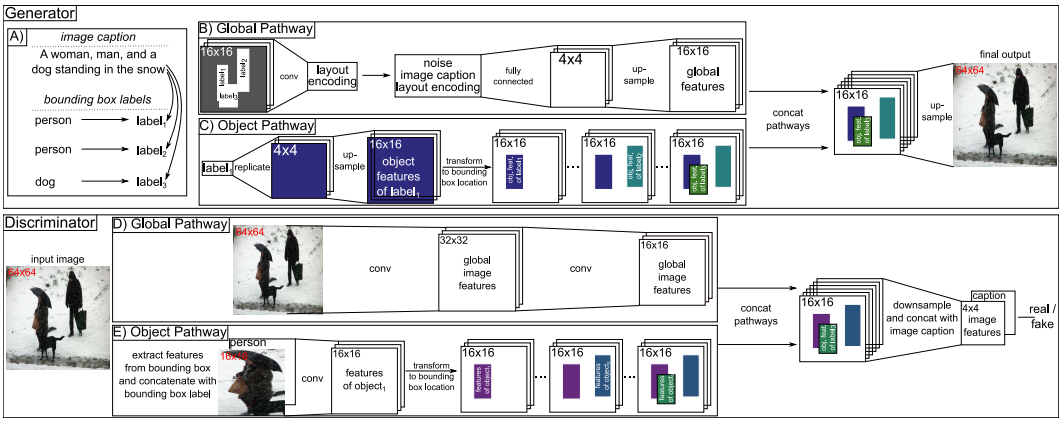


Figure 1: Both the generator and the discriminator of our model consist of a global and an object pathway. The global pathway focuses on global image characteristics, such as the background, while the object pathway is responsible for modeling individual objects at their specified location.

## APPROACH

For our approach[1] the central goal is to generate objects at arbitrary locations within a scene, while keeping the scene overall consistent. The generator (see Figure 1) gets as input a randomly sampled noise vector, the location and size of the individual bounding boxes, a label for each of the bounding boxes encoded as a one-hot vector, and, if existent, an image caption embedding obtained with a pretrained char-CNN-RNN network from Reed et al. (2016b). As a pre-processing step (A), the generator constructs labels for the individual bounding boxes from the image caption and the provided labels of each bounding box. For this we concatenate the image caption embedding and the one-hot vector of a given bounding box and create a new label embedding by applying a matrix-multiplication followed by a non-linearity (i.e. a fully connected layer). The resulting label contains the previous label as well as additional information from the image caption, such as color or shape, and is potentially more meaningful. In case of missing image captions we use the one-hot embedding only.

The generator consists of two different streams which get combined later in the process. First, the *global pathway* (B) is responsible for creating a general layout of the global scene. It processes the previously generated local labels for each of the bounding boxes and replicates them spatially at the location of each bounding box. In areas where the bounding boxes overlap the label embeddings are summed up, while the areas with no bounding boxes remain filled with zeros. Convolutional layers are applied to this layout to obtain a high-level layout encoding which is concatenated with the noise vector and the image caption embedding and the result is used to generate a general image layout.

Second, the *object pathway* (C) is responsible for generating features of the objects within the given bounding boxes. As input it receives the previously generated label and uses convolutional layers to create from this a features map of defined resolution. This feature map is further transformed with a Spatial Transformer Network (STN) Jaderberg et al. (2015) to fit into the bounding box at the given location on an empty canvas. The same convolutional layers are applied to each of the provided labels, i.e. we have one object pathway that is applied several times across different labels and whose output feeds onto the corresponding coordinates on the empty canvas. Again, features within overlapping bounding box areas are summed up, while areas outside of any bounding box remain zero.

As a final step, the outputs of the global and object pathways are concatenated along the channel axis and are used to generate the image in the final resolution, using common GAN procedures. The specific changes of the generator compared to standard architectures are the object pathway that generates additional features at specific locations based on provided labels, as well as the layout encoding which is used as additional input to the global pathway. These two extensions can be added to the generator in any existing architecture with limited extra effort.

[1]The complete approach and results will be made available via GitHub until presentation in the ICLR.

The discriminator receives as input an image (either original or generated), the location and size of the bounding boxes, the labels for the bounding boxes as one-hot vectors, and, if existent, the image caption embedding. Similarly to the generator, the discriminator also possesses both a global (D) and an object (E) pathway respectively. The global pathway takes the image and applies multiple convolutional layers to obtain a representation of the whole image. The object pathway first uses a STN to extract the objects from within the given bounding boxes and then concatenates these extracted features with the spatially replicated bounding box label. Next, convolutional layers are applied and the resulting features are again added onto an empty canvas within the coordinates specified by the bounding box. Note, similarly to the generator we only use *one* object pathway that is applied to multiple image locations, where the outputs are then added onto the empty canvas, summing up overlapping parts and keeping areas outside of the bounding boxes set to zero.

Finally, the outputs of both the object and global pathways are concatenated along the channel axis and we again apply convolutional layers to obtain a merged feature representation. At this point the features are concatenated with the spatially replicated image caption embedding (if existent) along the channel axis, one more convolutional layer is applied, and the output is classified as either generated or real. For the general training, we can utilise the same procedure that is used in the GAN architecture that is modified with our proposed approach.

## EVALUATION AND ANALYSIS

For the evaluation, we aim to study the quality of the generated images with a particular focus on the generalization capabilities and the contribution of specific parts of our model, in both controllable and large-scale cases[2]. Thus, in the following sections, we evaluate our approach on three different data sets: the Multi-MNIST data set, the CLEVR data set, and the MS-COCO data set.

### MULTI-MNIST

In our first experiment, we used the Multi-MNIST data set (Eslami et al., 2016) for testing the basic functionality of our proposed model. Using the implementation provided by Eslami et al. (2016), we created 50,000 images of resolution $64 \times 64$ px that contain exactly three normal-sized MNIST digits in non-overlapping locations on a black background.

As a first step, we tested whether our model can learn to generate digits at the specified locations and whether we can control the digit identity, the generated digit's size, and the number of generated digits per image. According to the results, we can control the location of individual digits, their identity, and their size, even though all training images contain exactly three digits in normal size. Figure 2 shows that we can control how many digits are generated within an image (rows A–B, for two to five digits) and various sizes of the bounding box (row C). As a second step, we created an additional Multi-MNIST data set in which all training images contain only digits 0–4 in the top half and only digits 5–9 in the bottom half of the image. For testing digits in the opposite half, we can see that the model is indeed capable of generalizing the position (row D, left), i.e. it can generate digits 0–4 in the bottom half of the image and digits 5–9 in the top half of the image. Nevertheless, we also observed that this does not always work perfectly, as the network sometimes alters digits towards the ones it has seen during training at the respective locations, e.g. producing a "4" more similar to a "9" if in bottom half of the image, or generating a "7" more similar to a "1" if in top half of the image.

As a next step, we created a Multi-MNIST data set with images that only contain digits in the top half of the image, while the bottom half is always empty. We can see (Figure 2, row D, right) that the resulting model is not able to generate digits in the bottom half of the image. Controlling for the location still works, i.e. bounding boxes are filled with "something", but the digit identity is not clearly recognizable. Thus, the model is able to control both the object identity and the object location within an image and can generalize to novel object locations to some extent.

To test the impact of our model extensions, i.e. the object pathway in both the generator and the discriminator as well as the layout encoding, we performed ablation studies on the previously created Multi-MNIST data set with three digits at random locations. We first disabled the use of the layout encoding in the generator and left the rest of the model unchanged. In the results (Figure 2, row

[2]More detailed information about the implementation can be found in the Appendix.

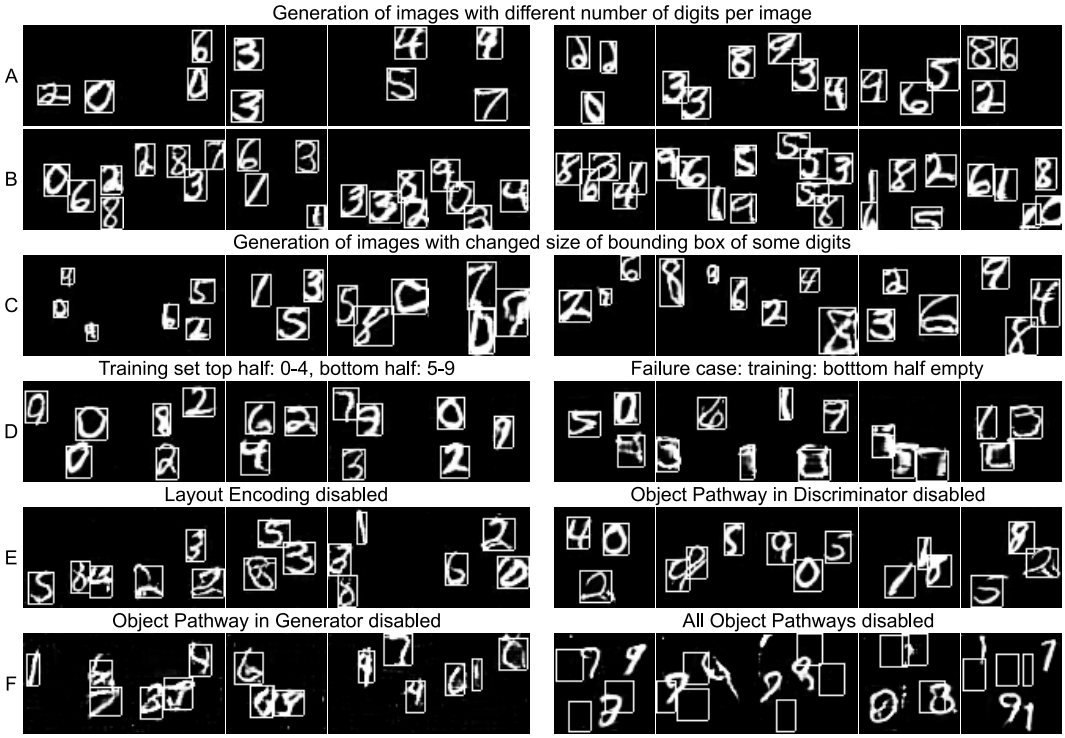


Figure 2: Multi-MNIST images generated by the model. The training set consists only of images with three individual digits of normal size. The bounding boxes are highlighted for visualization.

E, left), we can see that, overall, both the digit identity and the digit locations are still correct, but minor imperfections can be observed within various images. This is most likely due to the fact that the global pathway of the generator has no information about the digit identity and location until its features get merged with the object pathway. As a next test, we disabled the object pathway of the discriminator and left the rest of the model unmodified. Again, we see (row E, right) that we can still control the digit location, although, again, minor imperfections are visible. More strikingly, we have a noticeably higher error rate in the digit identity, i.e. the wrong digit is generated at a given location, most likely due to the fact that there is not object pathway in the discriminator controlling the object identity at the various locations. In comparison, the imperfections are different when only the object pathway of the generator is disabled (row F, left). The layout encoding and the feedback of the discriminator seem to be enough to still produce the digits in the correct image location, but the digit identity is often incorrect or not recognizable at all. Finally, we tested disabling the object pathway in both the discriminator and the generator (see row F, right). This leads to a loss of control of both image location as well as identity and sometimes even results in images with more or fewer than three digits per image. This shows that only the layout encoding, without any of the object pathways, is not enough to control the digit identity and location. Overall, these results indicate that we do indeed need both the layout encoding, for a better integration of the global and object pathways, and the object pathways in both the discriminator and the generator, for optimal results.

## CLEVR

In our second experiment we used more complex images containing multiple objects of different colors and shapes. The goal of this experiment was to evaluate the generalization ability of our object pathway across different object characteristics. For this, we performed tests similar to Raj et al. (2017), albeit on the more complex CLEVR data set (Johnson et al., 2017). In the CLEVR data set objects are characterized by multiple properties, in our case the shape, the color, and the size. Based on the implementation provided by Johnson et al. (2017), we rendered 25,000 images with a resolution of $64 \times 64$ pixels containing $2 - 4$ objects per image. The label for a given bounding box of an object is the object shape and color (both encoded as one-hot encoding and then concatenated), while the object size is specified through the height and width of the bounding box.

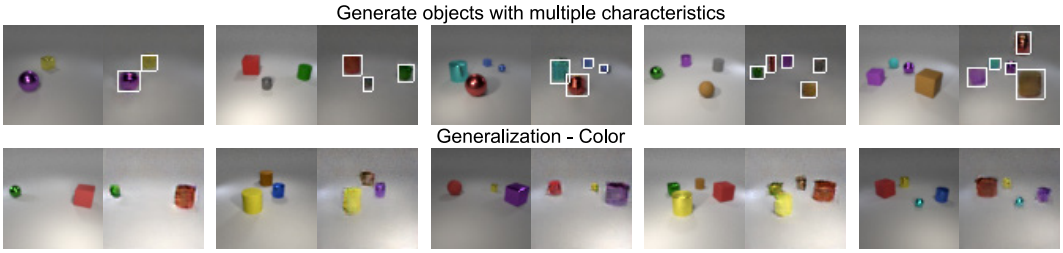


Figure 3: Images from the CLEVR data set. The left image of each pair shows the rendered image according to specific attributes. The right image of each pair is the image generated by our model.

Similar to the first experiment, we tested our model for controlling the object characteristics, size, and location. In the first row of Figure 3 we present the results of the trained model, where the left image of each pair shows the originally rendered one, while the right image was generated by our model. We can confirm that the model can control both the location and the objects' shape and color characteristics. The model can also generate images containing an arbitrary number of objects (forth and fifths pair), even though a maximum of four objects per image was seen during training.

The CLEVR data set offers a split specifically intended to test the generalization capability of a model, in which cylinders can be either red, green, purple, or cyan and cubes can be either gray, blue, brown, or yellow during training, while spheres can have any of these colors. During testing, the colors between cylinders and cubes are reversed. Based on these restrictions, we created a second data set of 25,000 training images for testing our model. Results of the test are shown in the second row of Figure 3 (again, left image of each pair shows the originally rendered one, while the right image was generated by our model). We can see that the color transfer to novel shape-color combinations takes place, but, similarly to the Multi-MNIST results, we can see some artifacts, where e.g. some cubes look a bit more like cylinders and vice versa. Overall, the CLEVR experiment confirms the indication that our model can control object characteristics (provided through labels) and object locations (provided through bounding boxes) and can generalize to novel object locations, novel amounts of objects per image, and novel object characteristic combinations within reasonable boundaries.

### MS-COCO

For our final experiment, we used the MS-COCO dataset Lin et al. (2014) to evaluate our model on natural images of complex scenes. In order to keep our evaluation comparable to previous work, we used the 2014 train/test split consisting of roughly 80,000 training and 40,000 test images and rescaled the images to a resolution of $256 \times 256$ px. At train-time, we used the bounding boxes and object labels of the three largest objects within an image, i.e. we used zero to three bounding boxes per image. Similarly to work by Johnson et al. (2018) we only considered objects that cover at least 2% of the image for the bounding boxes. To evaluate our results quantitatively, we computed both the Inception Score (IS, larger is better), which tries to evaluate how recognizable and diverse objects within images are (Salimans et al., 2016), as well as the Fréchet Inception Distance (FID, smaller is better), which compares the statistics of generated images with real images (Heusel et al., 2017). As a qualitative evaluation, we generated images that contain more than one object, and checked, whether the bounding boxes can control the object placement. We tested our approach with two commonly used architectures for text-to-image synthesis, namely the StackGAN (Zhang et al., 2017) and the AttnGAN (Xu et al., 2018b), and compared the images generated by these and our models.

In the StackGAN, the training process is divided into two steps: first, it learns a generator for images with a resolution of $64 \times 64$ pixels based on the image captions, and second, it trains a second generator, which uses the smaller images ($64 \times 64$ px) from the first generator and the image caption as input to generate images with a resolution of $256 \times 256$ px. Here, we added the object pathways and the layout encoding at the beginning of both the first generator and the second generator and used the object pathway in both discriminators. The other parts of StackGAN architecture and all hyperparameters remain the same as in the original training procedure for the MS-COCO data set. We trained the model three times from scratch and randomly sampled 3 times 30,000 image captions from the test set for each model. We then calculated the IS and FID values on each of the nine samples of 30,000 generated images and report the averaged values. As presented in Table 1, our StackGAN with

| Model | Resolution | IS ↑ | FID ↓ |
|---|---|---|---|
| GAN-INT-CLS Reed et al. (2016b) | $64 \times 64$ | $7.88 \pm 0.07$ | $60.62$ |
| StackGAN-V2 Zhang et al. (2018a) | $256 \times 256$ | $8.30 \pm 0.10$ | $81.59$ |
| StackGAN Zhang et al. (2018a) | $256 \times 256$ | $8.45 \pm 0.03$[1] | $74.05$ |
| PPGN Nguyen et al. (2017) | $227 \times 227$ | $9.58 \pm 0.21$ | |
| ChatPainter (StackGAN) Sharma et al. (2018) | $256 \times 256$ | $9.74 \pm 0.02$ | |
| Semantic Layout Hong et al. (2018b) | $128 \times 128$ | $11.46 \pm 0.09$[2] | |
| HDGan Zhang et al. (2018c) | $256 \times 256$ | $11.86 \pm 0.18$ | $71.27 \pm 0.12$[3] |
| AttnGAN Xu et al. (2018b) | $256 \times 256$ | $23.61 \pm 0.21$[4] | $33.10 \pm 0.11$[3] |
| *StackGAN + Object Pathways (Ours)*[5] | $256 \times 256$ | $12.12 \pm 0.31$ | $55.30 \pm 1.78$ |
| *AttnGAN + Object Pathways (Ours)* | $256 \times 256$ | $24.76 \pm 0.43$ | $33.35 \pm 1.15$ |

[1] Recently updated to $10.62 \pm 0.19$ in its source code.
[2] When using the ground truth bounding boxes at test time (as we do) the IS increases to $11.94 \pm 0.09$.
[3] FID score was calculated with samples generated with the pretrained model provided by the authors.
[4] The authors report a "best" value of $25.89 \pm 0.47$, but when calculating the IS with the pretrained model provided by the authors we only obtain an IS of $23.61$. Other researchers on the authors' Github website report a similar value for the pretrained model.
[5] We use the updated source code (IS of $10.62$) as our baseline model.

Table 1: Comparison of the Inception Score (IS) and Fréchet Inception Distance (FID) on the MS-COCO data set for different models. Note: the IS and FID values of our models are not necessarily directly comparable to the other models, since our model gets, in addition to the image caption, up to three bounding boxes and their respective object labels as input at test time.

added object pathways outperforms the original StackGAN both on the IS and the FID, increasing the IS from $10.62$ to $12.12$ and decreasing the FID from $74.05$ to $55.30$. Note, however, that this might also be due to the additional information our model is provided with as it receives up to three bounding boxes and respective bounding box labels per image in addition to the image caption.

We also extended the AttnGAN by (Xu et al., 2018b), the current state-of-the-art model on the MS-COCO data set (based on the Inception Score), with our object pathway to evaluate its impact on a different model. As opposed to the StackGAN, the AttnGAN consists of only one model which is trained end-to-end on the image captions by making use of multiple, intermediate, discriminators. Three discriminators judge the output of the generator at an image resolution of $64 \times 64$, $128 \times 128$, and $256 \times 256$ px. Through this, the image generation process is guided at multiple levels, which helps during the training process. Additionally, the AttnGAN implements an attention technique through which the networks focus on specific areas of the image for specific words in the image caption and adds an additional loss that checks if the image depicts the content as described by the image caption. There, in the same way as for the StackGAN, we added our object pathway at the beginning of the generator as well as to the discriminator that judges the generator outputs at a resolution of $64 \times 64$ px. All other discriminators, the higher layers of the generator, and all other hyperparameters and training details stay unchanged. Table 1 shows that adding the object pathway to the AttnGAN increases the IS of our baseline model (the pretrained model provided by the authors) from $23.61$ to $24.76$, while the FID is roughly the same as for the baseline model.

To evaluate whether the StackGAN model equipped with an object pathway (StackGAN+OP) actually generates objects at the given positions we generated images that contain multiple objects and inspected them visually. Figure 4 shows some example images, more results can be seen in the Appendix in Figure 6. We can observe that the StackGAN+OP indeed generates images in which the objects are at appropriate locations. In order to more closely inspect our global and object pathways, we can also disable them during the image generation process. Figure 5 shows additional examples, in which we generate the same image with either the global or the object pathway disabled during the generation process. Row C of Figure 5 shows images in which the object pathway was disabled and, indeed, we observe that the images contain mostly background information and objects at the location of the bounding boxes are either not present or of much less detail than when the object pathway is enabled. Conversely, row D of Figure 5 shows images which were generated when the global pathway was disabled. As expected, areas outside of the bounding boxes are empty, but we also observe that the bounding boxes indeed contain images that resemble the appropriate objects.

Figure 4: Examples of images generated from the given caption from the MS-COCO data set. *A)* shows the original images and the respective image captions, *B)* shows images generated by our StackGAN+OP (with the corresponding bounding boxes for visualization), and *C)* shows images generated by the original StackGAN Zhang et al. (2017)[3]

These results indicate, as in the previous experiments, that the global pathway does indeed model holistic image features, while the object pathway focuses on specific, individual objects.

When we add the object pathway to the AttnGAN (AttnGAN + OP) we can observe similar results[4]. Again, we are able to control the location and identity of objects through the object pathway, however, we observe that the AttnGAN+OP, as well as the AttnGAN in general, tends to place objects corresponding to specific features at many locations throughout the image. For example, if the caption contains the word "traffic light" the AttnGAN tends to place objects similar to traffic lights throughout the whole image. Since our model only focuses on generating objects at given locations, while not enforcing that these objects *only* occur at these locations, this behavior leads to the result that the AttnGAN+OP generates desired objects at the desired locations, but might also place the same object at other locations within the image. Overall, our experiments on the MS-COCO data set indicate that it is possible to add our object pathway to pre-existing GAN models without having to change the overall model architecture or training process. Through the addition of the object pathway it is possible to obtain more control over the image generation process and in some cases it even increases the quality of the generated images when evaluated through the IS or FID.

### DISCUSSION

Our experiments indicate that we do indeed get additional control over the image generation process through the introduction of object pathways in GANs. This enables us to control the identity and location of multiple objects within a given image based on bounding boxes and thereby facilitates the generation of more complex scenes. We further find that the division of work on a global and object pathway seems to improve the image quality both subjectively and based on quantitative metrics such as the Inception Score and the Fréchet Inception Distance.

The results further indicate that the focus on global image statistics by the global pathway and the more fine-grained attention to detail of specific objects by the object pathway works well. This is visualized for example in rows C and D of Figure 5. The global pathway (row C) generates features for the general image layout and background but does not provide sufficient details for individual objects. The object pathway (row D), on the other hand, focuses entirely on the individual objects and generates features specifically for a given object at a given location. While this is the desired behavior of our model it can also lead to sub-optimal images if there are not bounding boxes for objects that should be present within the image. This can often be the case if the foreground object is too small (in our case less than 2% of the total image) and is therefore not specifically labeled. In this case, the objects are sometimes not modeled in the image at all, despite being prominent in the respective image caption, since the object pathway does not generate any features. We can observe this, for example, in images described as "*many sheep are standing on the grass*", where the individual sheep

[3]Generated with the model from: `https://github.com/hanzhanggit/StackGAN-Pytorch`
[4]Examples of images generated by the AttnGAN+OP can be seen in the Appendix in Figure 7.

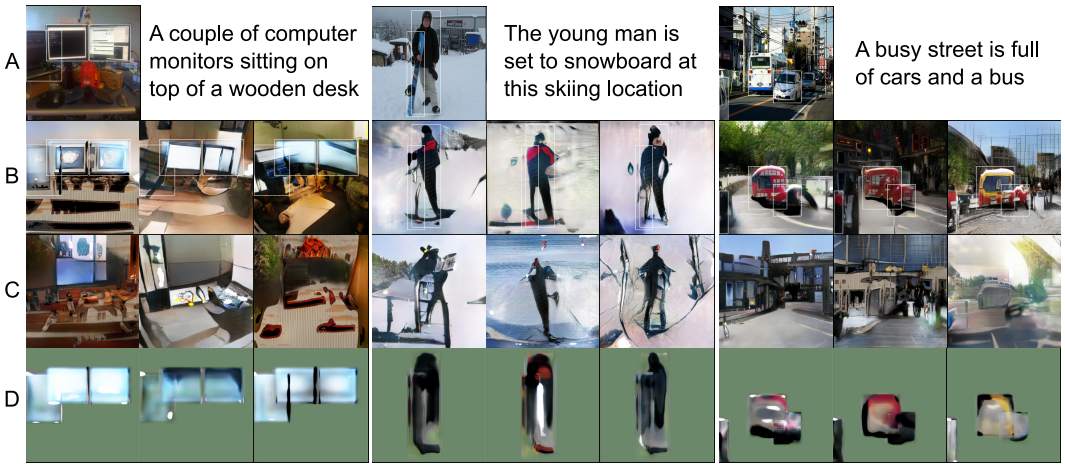


Figure 5: Examples of images generated from the given caption from the MS-COCO data set. *A)* shows the original images and the respective image captions, *B)* shows images generated by our StackGAN+OP (with the corresponding bounding boxes for visualization) with the object pathway enabled, *C)* shows images generated by the our StackGAN+OP when the object pathway is disabled, and *D)* shows images generated by the our StackGAN+OP when the global pathway is disabled.

are too small to warrant a bounding box. In this case, our model will often only generate an image depicting grass and other background details, while not containing any sheep at all.

Another weakness is that bounding boxes that overlap too much (empirically an overlap of more than roughly 30%) also often lead to sub-optimal objects at that location. Especially in the overlapping section of bounding boxes we often observe local inconsistencies or failures. This might be the result of our merging of the different features within the object pathway since they are simply added to each other at overlapping areas. A more sophisticated merging procedure could potentially alleviate this problem Another approach would be to additionally enhance the bounding box layout by predicting the specific object shape within each bounding box, as done for example by Hong et al. (2018b).

Finally, currently our model does not generate the bounding boxes and labels automatically. Instead, they have to be provided at test time which somewhat limits the usability for unsupervised image generation. However, even when using ground truth bounding boxes, our models still outperform other current approaches that are tested with ground truth bounding boxes (e.g. Hong et al. (2018b)) based on the IS and FID. This is even without the additional need of learning to specify the shape within each bounding box as done by Hong et al. (2018b). In the future, this limitation can be avoided by extracting the relevant bounding boxes and labels directly from the image caption, as it is done for example by Zhang et al. (2018c), Xu et al. (2018a), and Tan et al. (2018).

## CONCLUSION

With the goal of understanding how to gain more control over the image generation process in GANs, we introduced the concept of an additional object pathway. Such a mechanism for differentiating between a scene representation and object representations allows us to control the identity, location, and size of arbitrarily many objects within an image, as long as the objects do not overlap too strongly. In parallel, a global pathway, similar to a standard GAN, focuses on the general scene layout and generates holistic image features. The object pathway, on the other hand, gets as input an object label and uses this to generate features specifically for this object which are then placed at the location given by a bounding box The object pathway is applied iteratively for each object at each given location and as such, we obtain a representation of individual objects at individual locations and of the general image layout (background, etc.) as a whole. The features generated by the object and global pathway are then concatenated and are used to generate the final image output. Our tests on synthetic and real-world data sets suggest that the object pathway is an extension that can be added to common GAN architectures without much change to the original architecture and can, along with more fine-grained control over the image layout, also lead to better image quality.

ACKNOWLEDGMENTS

The acknowledgements are excluded in this version due to the double-blind review.

## REFERENCES

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016.

Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision*, pp. 1520–1529, 2017.

S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, koray kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Seunghoon Hong, Xinchen Yan, Thomas Huang, and Honglak Lee. Learning hierarchical semantic image manipulation through structured representations. *arXiv preprint arXiv:1808.07535*, 2018a.

Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7986–7994, 2018b.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pp. 2017–2025, 2015.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1988–1997, 2017.

Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1219–1228, 2018.

Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *arXiv preprint arXiv:1612.00215*, 2016.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pp. 740–755, 2014.

Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. In *International Conference on Learning Representations*, 2016.

Anh Nguyen, Yoshua Bengio, and Alexey Dosovitskiy. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4467–4477, 2017.

Amit Raj, Cusuh Ham, Huda Alamri, Vincent Cartillier, Stefan Lee, and James Hays. Compositional generation of images. In *Advances in Neural Information Processing Systems ViGIL*, 2017.

Scott Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In *Advances in Neural Information Processing Systems*, pp. 217–225, 2016a.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pp. 1060–1069, 2016b.

Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Victor Bapst, Matt Botvinick, and Nando de Freitas. Generating interpretable images with controllable structure. 2016c.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Shikhar Sharma, Dendi Suhubdy, Vincent Michalski, Samira Ebrahimi Kahou, and Yoshua Bengio. Chatpainter: Improving text to image generation using dialogue. In *International Conference on Learning Representations Workshop*, 2018.

Fuwen Tan, Song Feng, and Vicente Ordonez. Text2scene: Generating abstract scenes from textual descriptions. *arXiv preprint arXiv:1809.01110*, 2018.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8798–8807, 2018.

Kun Xu, Haoyu Liang, Jun Zhu, Hang Su, and Bo Zhang. Deep structured generative models. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018a.

Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018b.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE International Conference on Computer Vision*, pp. 5907–5915, 2017.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–16, 2018a.

Shengyu Zhang, Hao Dong, Wei Hu, Yike Guo, Chao Wu, Di Xie, and Fei Wu. Text-to-image synthesis via visual-memory creative adversarial network. In *Pacific Rim Conference on Multimedia*, pp. 417–427, 2018b.

Zizhao Zhang, Yuanpu Xie, and Lin Yang. Photographic text-to-image synthesis with a hierarchically-nested adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6199–6208, 2018c.

## IMPLEMENTATION DETAILS

Here we provide some more details about the exact implementation of our experiments.

### MULTI-MNIST

To train our GAN on the Multi-MNIST data set we use the Stage-I Generator and Discriminator from the StackGAN MS-COCO architecture[5]. However, we reduce the number of convolutional features in both the generator and the discriminator, i.e. $DF_DIM = 64$ and $GF_DIM = 128$. We also do not use any image captions and as such all steps involving image captions are skipped. The bounding box labels are one-hot vectors of size $[1, 9]$ encoding the digit identity. We train the networks for a total of 20 epochs on our training data set consisting of 50,000 images.

### CLEVR

To train our GAN on the CLEVR data set we also use the Stage-I Generator and Discriminator from the StackGAN MS-COCO architecture[5]. However, we reduce the number of convolutional features in both the generator and the discriminator, i.e. $DF_DIM = 48$ and $GF_DIM = 96$. We also do not use any image captions and as such all steps involving image captions are skipped. The bounding box labels are two one-hot vectors of size $[1, 8]$ and $[1, 3]$ encoding the object shape and color. The two one-hot vectors are concatenated to one final vectors of shape $[1, 11]$ and are used in this form during training. We train the networks for a total of 40 epochs on our training data set consisting of 25,000 images.

### MS-COCO

**StackGAN + Object Pathway** StackGAN-Stage-I: for training the Stage-I generator and discriminator (images of size $64 \times 64$ pixels) we take the provided StackGAN architecture for MS-COCO[5] which in this case represents our model without the object pathway (i.e. only the global pathway). Aside from the following modifications all hyperparameters and architecture parameters are the same as in the original StackGAN.

On the *Stage-I generator* we perform the following modifications:

- Bounding Box Labels: to obtain the bounding box labels we concatenate the image caption embedding[6] and the one-hot encoded bounding box label and apply a fully connected layer with $128$ units, batch normalization, and a ReLU activation to it, to obtain a label of shape $[1, 128]$ for each bounding box.
- Layout Encoding: to generate the layout encoding we first replicate each bounding box label spatially at the locations of the bounding boxes on an empty canvas of shape $[128, 16, 16]$. We then apply three convolutional layers with stride 2 and $64, 32, 16$ filters and batch normalization to this, and reshape the output to obtain a 64-dimensional vector which is our layout encoding.
- Object Pathway: the object pathway gets as input the previously generated bounding box label and replicates it spatially to a shape of $[128, 4, 4]$. We then use two convolutional layers with $768$ and $364$ filters and nearest-neighbor upsampling after each convolutional layer to obtain a feature representation of shape $[384, 16, 16]$. A Spatial Transformer Network is then applied to this feature representations to reshape it to the height and width of the bounding box and it is then added to an empty canvas of shape $[384, 16, 16]$ at the location of the bounding box. This procedure is repeated for each bounding box (maximum of three during training) and the output of the object pathway is a feature representation of shape $[384, 16, 16]$ containing features of the specified objects at the locations of the respective bounding boxes.
- Global Pathway: the only difference between our global pathway and the original StackGAN is that our global pathway gets as input not only the image caption embedding and random

[5] `https://github.com/hanzhanggit/StackGAN-Pytorch`

[6] Downloaded from `https://github.com/reedscot/icml2016`

noise but additionally the layout encoding, concatenated with the image caption embedding and the noise.

- Combination of global and object pathway: the output of the object pathway is merged with the output of the global pathway (standard StackGAN layers) at the resolution of $[-1, 16, 16]$ by concatenating the two outputs along the channel axis. After this we continue using the standard StackGAN architecture to generate images of resolution $[3, 64, 64]$ pixels.

On the *Stage-I discriminator* we perform the following modifications:

- Object Pathway: on the input image (resolution of $64 \times 64$ pixels) we use a Spatial Transformer Network to extract the features from the bounding box and reshape those features to a shape of $[3, 16, 16]$. We spatially replicate the bounding box label (one-hot encoding) to a shape of $[-1, 16, 16]$ and concatenate them with the extracted features along the channel axis. This is then given to the object pathway which consists of one convolutional layer with $192$ filters, batch normalization, and a LeakyReLU activation ($alpha = 0.2$). The output of the object pathway is again transformed to the width and height of the bounding box with a Spatial Transformer Network and then added to an empty canvas of shape $[192, 16, 16]$. This procedure is done to each of the bounding boxes within the image (maximum of three during training).
- Global Pathway: the global pathway consists of the standard StackGAN layers, i.e. it gets as input the image ($64 \times 64$ pixels) and applies convolutional layers with stride 2 to it.
- Combination of global and object pathway: the output features of the global and object pathways get concatenated along the channel axis at the resolution of $[-1, 16, 16]$. After this we continue in the same way as the original StackGAN.

StackGAN-Stage-II: in the second part of the training we train a second generator and discriminator to generate images with a resolution of $256 \times 256$ pixels. The generator gets as input images with a resolution of $64 \times 64$ pixels (generated by the Stage-I generator) and the image caption, and uses them to generate images with a $256 \times 256$ pixels resolution. A new discriminator is trained to distinguish between real and generated images.

On the *Stage-II generator* we perform the following modifications:

- Bounding Box Labels: the procedure to obtain the bounding box labels is the same as in the Stage-I generator.
- Image Encoding: as in the standard StackGAN we use convolutional layers to encode the input image of shape $[3, 64, 64]$ into a feature representation of shape $[-1, 16, 16]$.
- Layout Encoding: to generate the layout encoding, we first replicate each bounding box label spatially at the locations of the bounding boxes on an empty canvas of shape $[128, 16, 16]$ and then concatenate it along the channel axis with the image encoding and the spatially replicated image caption embedding. As in the standard StackGAN we then apply more convolutional layers with residual connection to obtain the final image embedding of shape $[-1, 16, 16]$, which provides the input for both the object and the global pathway.
- Object Pathway: the object pathway gets as input the image embedding described in the previous step. It then uses a Spatial Transformer Network to extract the features within the bounding box and reshapes those features to $[-1, 16, 16]$. It then applies two convolutional layers with $192$ and $96$ filters and increases the resolution by a factor of two after each layer through nearest neighbor upsampling. After that it uses a Spatial Transformer Network to embed the features within the bounding box region on an empty $[-1, 64, 64]$ canvas. This is done for each of the bounding boxes within the image.
- Global Pathway: the global pathway also gets as input the image encoding and uses the same convolutional layers and upsampling procedures as the original StackGAN Stage-II generator.
- Combination of global and object pathway: the output of the object pathway is merged with the output of the global pathway (standard StackGAN layers) at the resolution of $[-1, 64, 64]$ by concatenating the two outputs along the channel axis. After this we continue using the standard StackGAN architecture to generate images of shape $[3, 256, 256]$.

On the *Stage-II discriminator* we perform the following modifications:

- Object Pathway: on the input image (resolution of $256 \times 256$ pixels) we use a Spatial Transformer Network to extract the features from the bounding box and reshape those features to a shape of $[3, 32, 32]$. We spatially replicate the bounding box label (one-hot encoding) to a shape of $[-1, 32, 32]$ and concatenate them with the extracted features along the channel axis. This is then given to the object pathway which consists of two convolutional layers with 192 and 96 filters, batch normalization, and a LeakyReLU activation ($alpha = 0.2$). The output of the object pathway is again transformed to the width and height of the bounding box with a Spatial Transformer Network and then added to an empty canvas of shape $[96, 32, 32]$. This procedure is done to each of the bounding boxes within the image (maximum of three during training).
- Global Pathway: the global pathway consists of the standard StackGAN layers, i.e. it gets as input the image ($256 \times 256$ pixels) and applies convolutional layers with stride 2 to it.
- Combination of global and object pathway: the output features of the global and local features get concatenated along the channel axis at the resolution of $[-1, 32, 32]$. After this we continue in the same way as the original StackGAN.

**AttnGAN + Object Pathway** On the AttnGAN[7] we only modify the training at the lower layers of the generator and the first discriminator (working on images of $64 \times 64$ pixels resolution). For this we perform the same modifications as described in the StackGAN-Stage-I generator and discriminator.

## ADDITIONAL EXAMPLES OF MS-COCO RESULTS: STACKGAN

Figure 6 shows results of text-to-image synthesis on the MS-COCO data set with the StackGAN architecture. Rows A show the original image and image caption, rows B show the images generated by our StackGAN + Object Pathway and the given bounding boxes for visualization, and rows C show images generated by the original StackGAN (pretrained model obtained from `https://github.com/hanzhanggit/StackGAN-Pytorch`). The last block of examples (last row) show typical failure cases of our model, where there is no bounding box for the foreground object present. As a result our model only generates the background, without the appropriate foreground object, even though the foreground object is very clearly described in the image caption.

## ADDITIONAL EXAMPLES OF MS-COCO RESULTS: ATTNGAN

Figure 7 shows results of text-to-image synthesis on the MS-COCO data set with the AttnGAN architecture. Rows A show the original image and image caption, rows B show the images generated by our AttnGAN + Object Pathway and the given bounding boxes for visualization, and rows C show images generated by the original AttnGAN (pretrained model obtained from `https://github.com/taoxugit/AttnGAN`). The last block of examples (last row) show typical failure cases, in which the model does generate the appropriate object within the bounding box, but also places the same object at multiple other locations within the image.

[7] `https://github.com/taoxugit/AttnGAN`

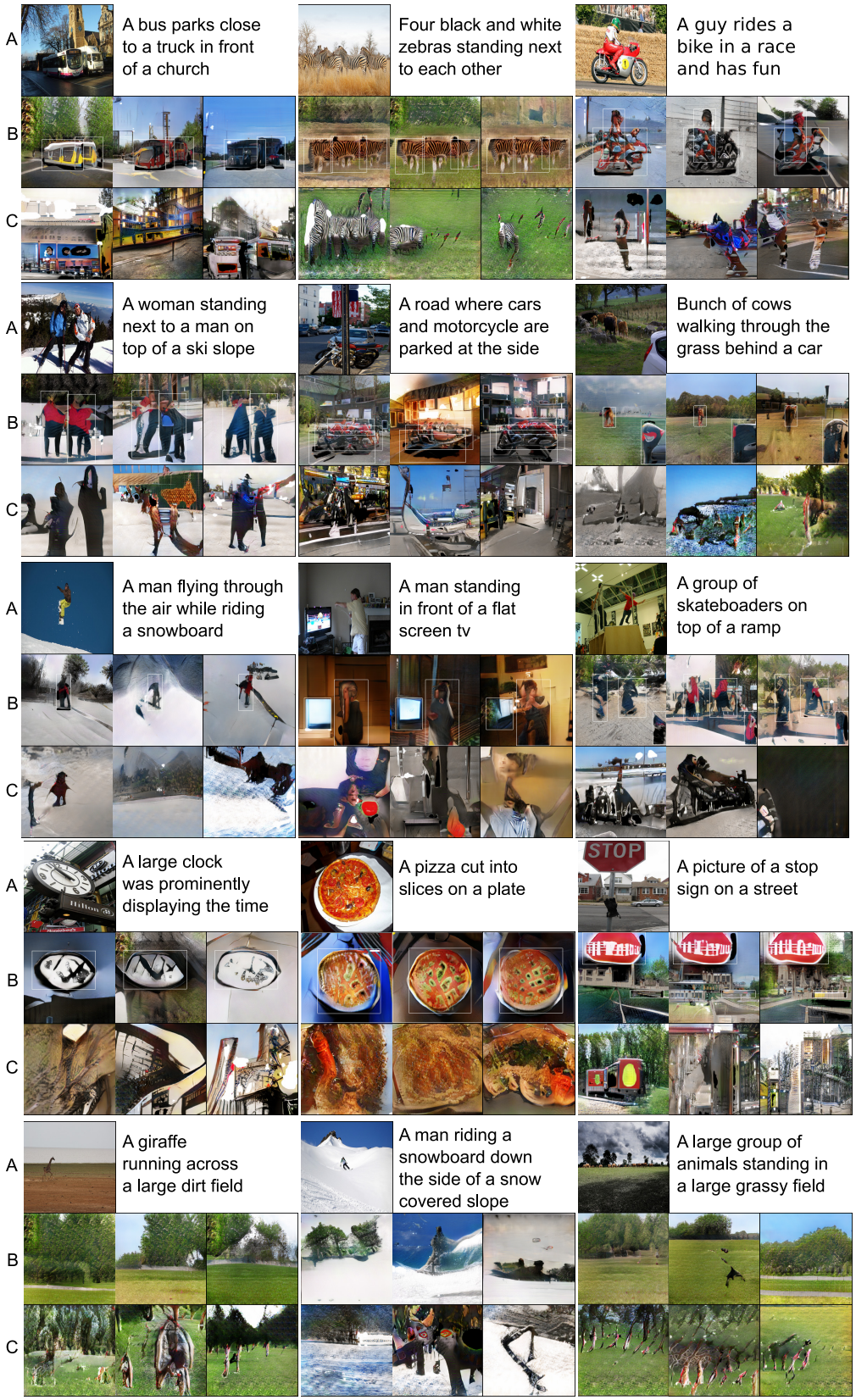


Figure 6: Please refer to previous page for information about the figure.

Figure 7: Please refer to page 14 for information about the figure.