Keep Drawing It: Iterative language-based image generation and editing

Alaaeldin El-Nouby*[1,4], Shikhar Sharma[2], Hannes Schulz[2], Devon Hjelm[2,3,5],
Layla El Asri[2], Samira Ebrahimi Kahou[2], Yoshua Bengio[3,5,6], Graham W. Taylor[1,4,6]

[1]University of Guelph [2]Microsoft Research [3]Montreal Institute for Learning Algorithms
[4]Vector Institute for Artificial Intelligence [5]University of Montreal
[6]Canadian Institute for Advanced Research
aelnouby@uoguelph.ca,
{shikhar.sharma, hannes.schulz, devon.hjelm, layla.elasri, samira.ebrahimi}@microsoft.com
yoshua.bengio@mila.quebec, gwtaylor@uoguelph.ca

Abstract

Conditional text-to-image generation approaches commonly focus on generating a single image in a single step. One practical extension beyond one-step generation is an interactive system that generates an image iteratively, conditioned on ongoing linguistic input / feedback. This is significantly more challenging as such a system must understand and keep track of the ongoing context and history. In this work, we present a recurrent image generation model which takes into account both the generated output up to the current step as well as all past instructions for generation. We show that our model is able to generate the background, add new objects, apply simple transformations to existing objects, and correct previous mistakes. We believe our approach is an important step toward interactive generation.

1 Motivation and Related Work

Computer vision is a fundamental skill for building intelligent agents as vision is one of the primary modes from which humans build their experience and understanding. Applications of computer vision are far-reaching and in this work, we focus on visual generation. Intelligent systems that can generate visual outputs can be used for education, entertainment, graphic design, and the creative arts. A natural language interface to such systems would make computer vision technology accessible to a larger population. For instance, a model that can understand text-based instructions and edit an image based on these instructions would be usable by non-experts.

Recent advances in generative modeling of images [1–3] have fueled further advances in generation conditioned on labels [4], captions [5–8], and dialogues [9]. A next step is to make this process interactive and learn to iteratively generate images based on previously generated images and from continual linguistic input. Making use of the Collaborative Drawing [CoDraw, 10] dataset, we define the Generative Neural Visual Artist (GeNeVA) task. This task models an interaction between a teller and a drawer. With an image in mind, the teller guides the drawer through composing this image via linguistic instructions and feedback. This is a challenging task because the drawer needs to understand the mapping between the instructions and what they refer to on the drawing canvas. Additionally, the drawer needs to resolve ambiguous instructions and modify the existing drawing to accommodate new changes in a plausible manner. The ideal drawer should also be able to modify the drawing without violating any of the past instructions.

In recent work, Sharma et al. [9] proposed a model that generates images using dialogue. This model is not interactive and does not proceed iteratively: it reads the entire dialogue to generate a single

*work was performed during an internship with Microsoft Research

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

final image. There has also been recent work on generating images in a recurrent manner but without incorporating any external linguistic input nor feedback [11, 12]. Another model, proposed by Lin et al. [13], recursively generates parameters of transformations instead of generating the images themselves. To the best of our knowledge, our model is the first model that recursively generates and modifies intermediate images based on continual text instructions such that the images are consistent with past instructions. To summarize, the key contributions of this research are:

- We introduce the GeNeVA task and propose a novel recurrent Generative Adversarial Network (GAN) architecture that specializes in plausible image modification in the context of a dialogue.
- We introduce the Interactive CLEVR (i-CLEVR) dataset, a modified version of Compositional Language and Elementary Visual Reasoning (CLEVR) [14], for incremental building of CLEVR-style scenes based on linguistic instructions.

2 Model

In this section, we describe our conditional recurrent GAN model for the GeNeVA task. An overview of the model architecture is shown in Figure 1.

During an n-step interaction between a teller and a drawer, the teller provides a drawing canvas x_0 and instructions $Q = \{q_0, q_1, \ldots, q_n\}$. For each turn of the conversation, the conditioned generator G generates a new image: $\widetilde{x}_t = G(z \mid f_{G_{t-1}}, h_t)$, where z is a noise vector sampled from a normal distribution $\mathcal{N}(0, 1)$ of dimension N_z, $f_{G_{t-1}}$ is a context-free condition, and h_t is a context-aware condition. The context-free condition $f_{G_{t-1}} = E_G(\widetilde{x}_{t-1})$ is an encoding of the previously generated image $\widetilde{x}_{t-1}$ using an encoder E_G which is a shallow Convolutional Neural Network (CNN). The encoding produces low resolution feature maps of dimensions $(K_g \times K_g \times N_g)$.

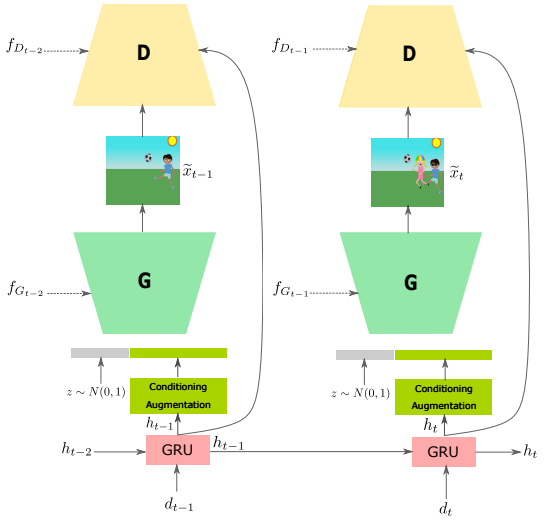


Figure 1: At each time step, G generates a new image conditioned on the previous image encoding $f_{G_{t-1}}$ and the conversation encoding h_t.

Each instruction q_t is encoded using a bi-directional Gated Recurrent Unit (GRU) on top of GloVe word embeddings [15]. The instruction encoding is denoted by d_t.

To provide context, both in terms of instructions and previously generated images, the context-aware condition $h_t = R(d_t, h_{t-1})$ is the output of a recursive function R which takes the instruction encoding d_t as well as the previous condition h_{t-1} as inputs. We implement R with a GRU. The dimension of h_t is N_c.

The context-free condition $f_{G_{t-1}}$ represents the prior that the model is given from the most recently generated image whereas the context-aware condition h_t represents the additions or modifications that the teller is describing for the new image. We would like to let the model decide how much it should rely on $f_{G_{t-1}}$ given the context h_t. Let $L_{f_{G_t}}$ be the feature maps of an intermediate layer in the generator. $L_{f_{G_t}}$ has the exact same dimensions as $f_{G_{t-1}}$. The feature maps passed to the next layer $\hat{L}_{f_{G_t}}$ are a convex combination of $f_{G_{t-1}}$ and $L_{f_{G_t}}$:

$$\hat{L}_{f_{G_t}} = \sigma_t L_{f_{G_t}} + (1 - \sigma_t) f_{G_{t-1}} \ ,$$

where σ_t is a vector of length N_g whose elements lie in the range [0, 1], computed from the context-aware condition h_t using two linear layers followed by a sigmoid function.

In our implementation, the context-aware condition h_t is concatenated to the noise vector z after applying conditioning augmentation [6] as shown in Figure 1. Similar to Miyato and Koyama [16], h_t is used to condition batch normalization for all the convolutional layers of the generator.

Since we are modeling iterative composition of images, having a discriminator D that at each step only distinguishes between realistic and unrealistic images will not be sufficient. The discriminator

should be able to tell when the image is modified incorrectly or not modified at all. To ensure this, we make three modifications to the discriminator. First, D is conditioned on the context-aware vector h_t and context-free feature maps $f_{D_{t-1}} = E_D(\widetilde{x}_{t-1})$ of dimension $(K_d \times K_d \times N_d)$, where the image encoder E_D is a shallow CNN. To encourage the discriminator to focus on the modifications, $f_{D_{t-1}}$ is subtracted from the discriminator's intermediate layer L_{f_D} with the same spatial dimensions (same number of channels N_d as well). The context-aware condition h_t is applied through projections [16]. Second, for the discriminator loss, in addition to labeling real images as positive examples and generated images as negative examples, we add a term for the combination of (real image, wrong instruction), similar to Reed et al. [5]. Finally, we add to the discriminator the auxiliary objective [17] of detecting all objects added at the current time step:

$$L_D = L_{D_{\text{real}}} + \lambda L_{D_{\text{fake}}} + (1-\lambda) L_{D_{\text{wrong}}} + \beta L_{\text{aux}}.$$

The generator and discriminator are trained alternately to minimize the adversarial hinge loss [18–20].

The loss function for the auxiliary task is simply a binary cross entropy over N objects. The generator loss is unchanged. Additionally, to help with training stability, we apply zero-centered gradient penalty regularization on the real data alone as suggested by Mescheder, Geiger, and Nowozin [21].

The network architecture for the generator and discriminator follows the ResBlocks architecture introduced by Miyato and Koyama [16]. Following SAGAN [20], we use spectral normalization for all layers in the discriminator. We add a self-attention layer to the intermediate layers with spatial dimensions 16×16 for the discriminator and the generator. The image encoders E_G and E_D have separate parameters, as they have to be optimized for adversarial objectives, the generator's and the discriminator's respectively. For the training dynamics, the generator and discriminator parameters are updated every time step, while the encoders' parameters are updated every sequence. The text encoder and the GRU are trained with respect to the discriminator objective only. For training, we use teacher forcing with the ground truth images x_{t-1} instead of the generated image $\widetilde{x}_{t-1}$, but we use $\widetilde{x}_{t-1}$ during test time. Additional implementation details are provided in Appendix D.

3 Datasets

For the GeNeVA task, we require a dataset that contains textual instructions describing drawing actions and the corresponding ground-truth images for each instruction. To the best of our knowledge, the only publicly available dataset that fits this task is Collaborative Drawing (CoDraw). We provide a brief summary of CoDraw in Appendix B.2. We also create a new dataset called i-CLEVR specifically designed for this task.

i-CLEVR: CLEVR [14] is a popular dataset for Visual Question Answering (VQA). It is programmatically generated, and its generation code is open-source[2]. CLEVR contains images of collections of objects with different shapes, colors, materials and sizes. Each image is assigned complex questions about object counts, attributes or existence.

We modify CLEVR to create an interactive version with sequences of images. Every image comes with an instruction to add an object with some specific attributes in a position relative to objects that already exist in the image. The task is to add the object with the correct attributes in a plausible position based on the textual instruction. To make the task more complex and force the model to have context, we refer to the most recently added object by *it* instead of stating its attributes. The initial drawing canvas x_0 for i-CLEVR consists of an empty background. The dataset contains 10,000 sequences. Each sequence has 5 images and instructions. More details about the dataset generation along with examples can be found in Appendix B.1. We plan to release the dataset to the public soon.

Both datasets only have at most one instance of the same object in a sequence. For CoDraw, we treat the concatenated utterances of the drawer and the teller at time step t as the instruction.

4 Results

Evaluation Metrics: Standard metrics for evaluating GANs such as the inception score or FID only capture how realistic the generations look relative to the real images. They cannot capture if

[2] `https://github.com/facebookresearch/clevr-dataset-gen`

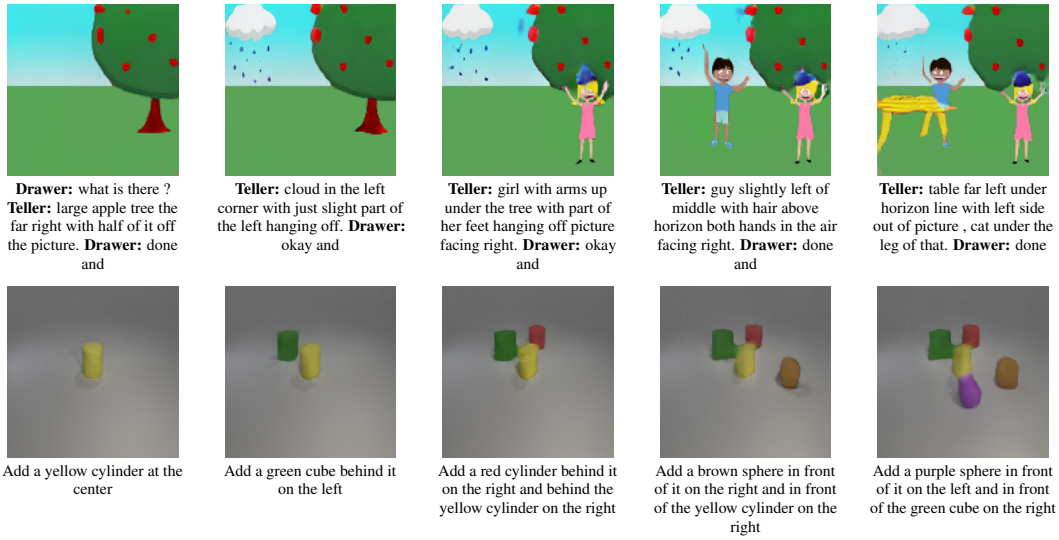


Figure 2: Generation examples from our model on CoDraw (top row) and i-CLEVR (bottom-row).

the model is correctly modifying the images as described in the GeNeVA task instructions. A good evaluation metric for this task needs to identify if all the objects that were described by the teller are present in the generated image. It should also check that the objects' positions and relationships match the instructions. To capture all of these constraints, we train an object localizer on the training data. For every example, we compare the detections of this localizer on the real images and the generated ones. We present the precision, recall, and F1-score for this object detection task. We also construct a graph where the nodes are objects present in the images and edges are positional relationships: left, right, behind, front. We compare the graphs constructed from the real and the generated images to test the correct placement of objects, without requiring the model to draw the objects in the same exact locations (which would have defied its generative nature). More details about this evaluation metric can be found in Appendix C.

Table 1: Evaluation results for experiments on CoDraw and i-CLEVR datasets

Metric	CoDraw	i-CLEVR
Precision	67.61	75.71
Recall	56.36	64.68
F1-Score	61.47	69.76
Relational Similarity	42.01	56.91

We present some examples of images generated by our model in Figure 2. Due to space constraints, more generated images are presented in Appendix A.

Qualitative and Quantitative Results: On CoDraw, we observe that the model is able to generate scenes that are consistent with the conversation and generation history, and gets most of the coarse details correct (such as large objects and their relative positions) but has difficulty in capturing fine-grained details (such as tiny objects, facial expressions, and object poses). The model also struggles when a single instruction asks to add several objects at once. For i-CLEVR, the model captures spatial relationships and colours very accurately as demonstrated in Figure 2. However, there is some ambiguity in the object shapes, especially between cylinders and cubes. In some instances, the model fails to add the fifth object when the image is already crowded and there is no space left to add it without moving the others. We present all the quantitative metrics in Table 1.

Conclusion and Future Work: We presented a recurrent GAN model for the GeNeVA task and show that the model is able to draw reasonable images for the provided instructions. An interesting future research direction would be to have a system that can also ask questions to the user when it needs clarifications. More datasets are also needed to scale this task to photo-realistic images.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. 2014.

[2] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2014.

[3] A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu. "Pixel Recurrent Neural Networks". In: *International Conference on Machine Learning (ICML)*. 2016.

[4] M. Mirza and S. Osindero. "Conditional Generative Adversarial Nets" (2014). arXiv: 1411.1784 [cs.AI].

[5] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. "Generative Adversarial Text to Image Synthesis". In: *International Conference on Machine Learning (ICML)*. 2016.

[6] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks". In: *International Conference on Computer Vision (ICCV)*. 2017.

[7] S. Hong, D. Yang, J. Choi, and H. Lee. "Inferring Semantic Layout for Hierarchical Text-to-Image Synthesis". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[8] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. "AttnGAN: Fine-Grained Text to Image Generation With Attentional Generative Adversarial Networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[9] S. Sharma, D. Suhubdy, V. Michalski, S. E. Kahou, and Y. Bengio. "ChatPainter: Improving Text to Image Generation using Dialogue". In: *International Conference on Learning Representations (ICLR) Workshop*. 2018.

[10] J.-H. Kim, D. Parikh, D. Batra, B.-T. Zhang, and Y. Tian. "CoDraw: Visual Dialog for Collaborative Drawing" (2017). arXiv: 1712.05558 [cs.CV].

[11] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic. "Generating images with recurrent adversarial networks" (2016). arXiv: 1602.05110 [cs.LG].

[12] J. Yang, A. Kannan, D. Batra, and D. Parikh. "LR-GAN: Layered recursive generative adversarial networks for image generation". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017.

[13] C.-H. Lin, E. Yumer, O. Wang, E. Shechtman, and S. Lucey. "ST-GAN: Spatial Transformer Generative Adversarial Networks for Image Compositing". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[14] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. Girshick. "Inferring and Executing Programs for Visual Reasoning". In: *International Conference on Computer Vision (ICCV)*. 2017.

[15] J. Pennington, R. Socher, and C. Manning. "Glove: Global vectors for word representation". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[16] T. Miyato and M. Koyama. "cGANs with Projection Discriminator". In: *International Conference on Learning Representations (ICLR)*. 2018.

[17] A. Odena, C. Olah, and J. Shlens. "Conditional Image Synthesis with Auxiliary Classifier GANs". In: *International Conference on Machine Learning (ICML)*. 2017.

[18] J. H. Lim and J. C. Ye. "Geometric gan" (2017). arXiv: 1705.02894 [stat.ML].

[19] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. "Spectral Normalization for Generative Adversarial Networks". In: *International Conference on Learning Representations (ICLR)*. 2018.

[20] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. "Self-Attention Generative Adversarial Networks" (2018). arXiv: 1805.08318 [stat.ML].

[21] L. Mescheder, A. Geiger, and S. Nowozin. "Which Training Methods for GANs do actually Converge?" In: *International Conference on Machine learning (ICML)*. 2018.

[22] Blender Online Community. "Blender - a 3D modelling and rendering package" (2016).

[23] J. Ba, R. Kiros, and G. E. Hinton. "Layer Normalization" (2016). arXiv: 1607.06450 [stat.ML].

[24] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". *Journal of Machine Learning Research (JMLR)* (2015).

[25] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." (2014). arXiv: 1412.6980 [cs.LG].

[26] T. Tieleman and G. Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". *COURSERA: Neural networks for machine learning* 4.2 (2012).

A Generation Examples

In this section we present some examples generated using our model on two datasets, CoDraw and i-CLEVR.

A.1 CoDraw

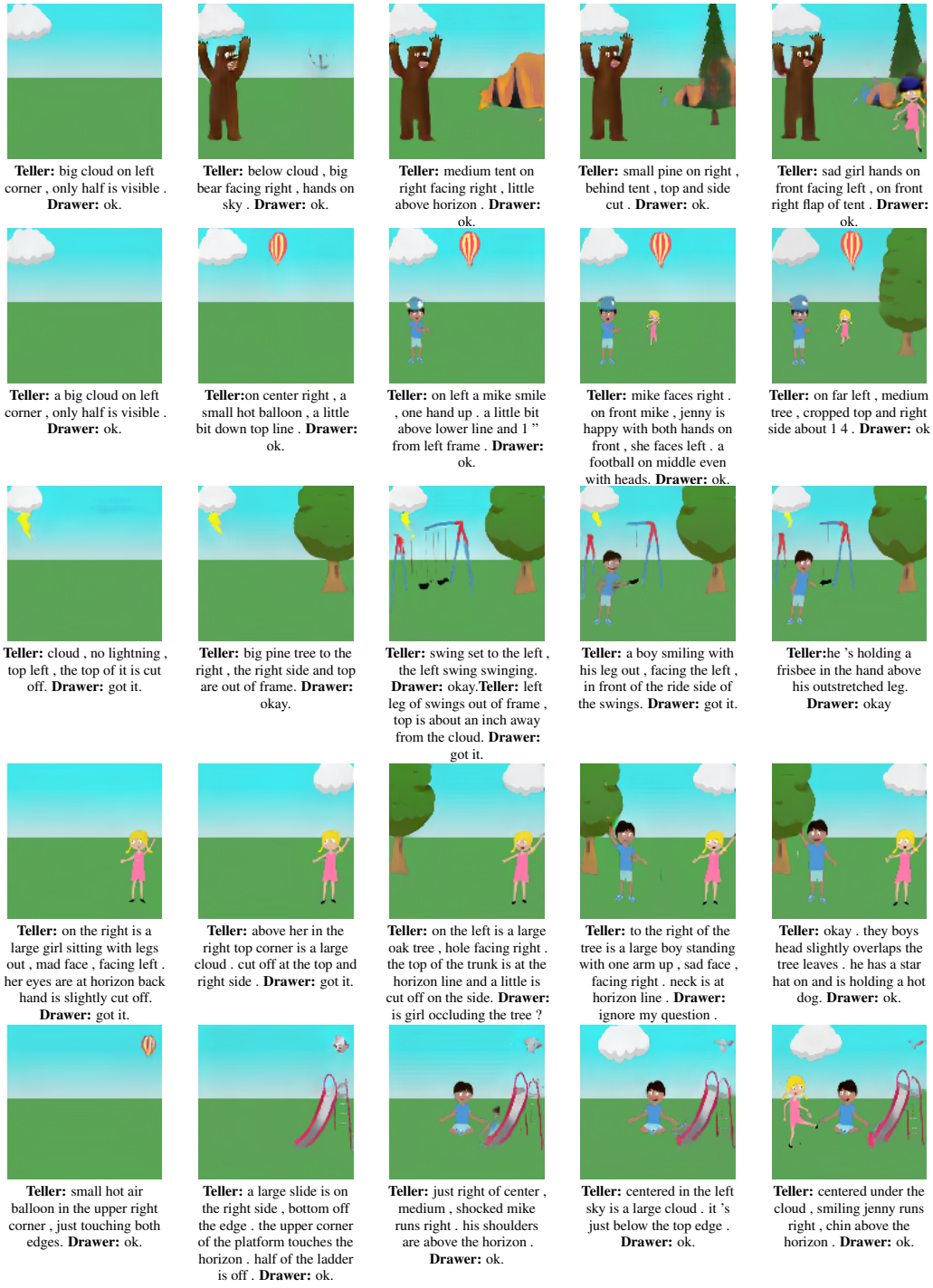


Figure 3: Generation examples from our model for the CoDraw dataset.

A.2 i-CLEVR

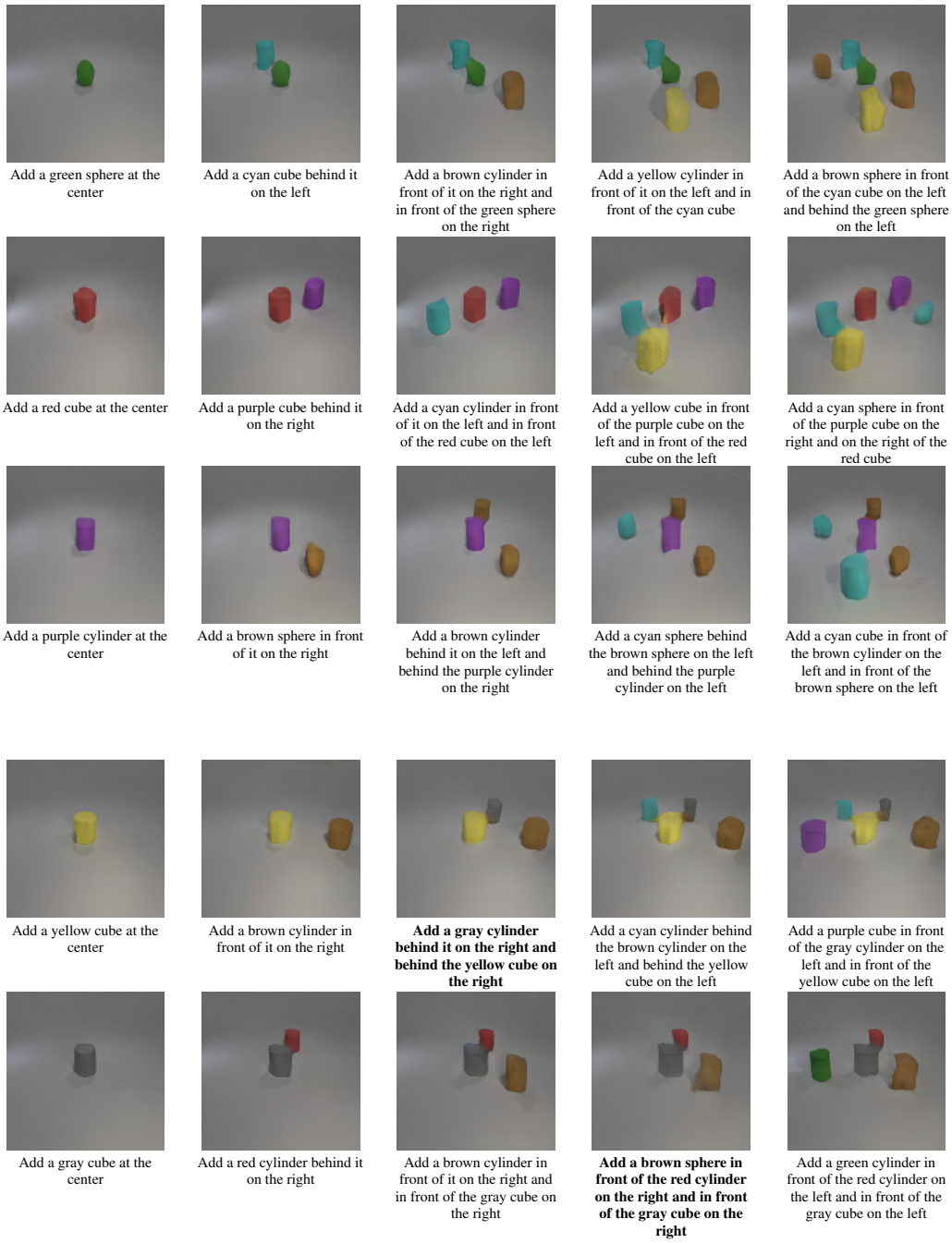
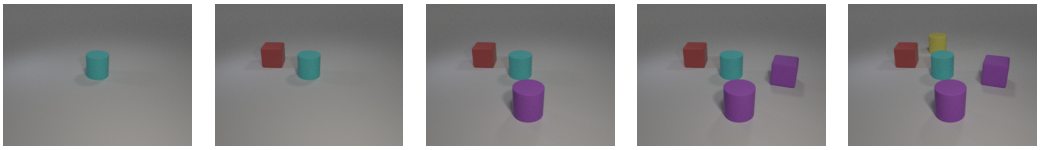


Figure 4: Generation examples from our model for the i-CLEVR dataset. Instructions where the model made a mistake are in bold.

B Datasets

As the GeNeVA is a newly-proposed task, we needed to either repurpose existing, or create new datasets for evaluation. In this section, we describe the details of building the two datasets used in our experiments.



Add a cyan cylinder at the center | Add a red cube behind it on the left | Add a purple cylinder in front of it on the right and in front of the cyan cylinder | Add a purple cube behind it on the right and in front of the red cube on the right | Add a yellow cylinder behind the purple cylinder on the left and behind the red cube on the right

Figure 5: An example i-CLEVR images-instruction sequence.

B.1 i-CLEVR

CLEVR [14] is one of the popular datasets for the VQA task which is programmatically generated. We build on top of the open-source generation code[3] for CLEVR to create interactive CLEVR (i-CLEVR). Each example in the dataset consists of a sequence of 5 (image, instructions) pairs. Starting from an empty canvas (background), each instruction describes an object to add the canvas in terms of shape and color. The instruction also describes where the object should be placed relative to existing objects in the scene. In order to make the task more challenging and to require awareness of the context, the most recently added object is referred to as "*it*". An example from the i-CLEVR dataset is presented in Figure 5.

To generate the image for each step in the sequence, an object with random attributes is rendered to the scene using Blender [22]. However, we make sure that all objects have a unique combination of attributes. Each object can have one of 3 shapes (cube, sphere, cylinder) and one of 8 colors. In contrast to CLEVR, we have a fixed material and size for objects. For the first image in the sequence, the object placement is fixed to the image center. For all the following images, the objects are placed in a random position while maintaining visibility (not completely occluded) and at a minimum distance from other objects.

As for the instruction generation, we use a simple text template. For example, the second instruction in the sequence will have the following template:
Add a [object color] [object shape] [relative position: depth] it on the [relative position: horizontal]

Starting from the third instruction, the object position is described relative to two objects. These two objects are chosen randomly.

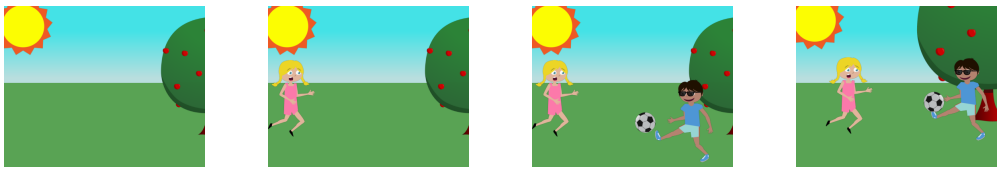
The i-CLEVR dataset consists of 10,000 sequences, consisting of 50,000 images and instructions. The training split consists of 6,000 sequences while the validation and testing splits consist of 2,000 sequences each. The dataset and its generation code will be made public.

B.2 CoDraw

CoDraw [10] is a recently released clip art-like dataset. It consists of scenes of images of children playing in a park. The children have different poses and expressions, the scenes include other objects as trees, tables, animals, etc. There are 58 object types in total. For every scene, there is a conversation between a teller and a drawer (both Amazon Mechanical Turk (AMT) workers) in natural language. The drawer updates the canvas based on teller instructions, the drawer can ask questions as well for clarification. The dataset consists of 9,993 scenes of varying lengths. An example of such scenes is shown in Figure 6. The initial drawing canvas x_0 for CoDraw provided to the drawer consists of the background having just the sky and grass.

Pre-processing: In some instances of the original dataset, the drawer waited for multiple teller turns before modifying the image. In these cases, we concatenate consecutive turns into a single turn until the drawer modifies the image. We also concatenate turns until a new object has been added or removed. Thus, every turn has an image corresponding to it with different number of objects. We inject a special delimiting token between the teller and drawer utterances in the same turn. The teller

[3] `https://github.com/facebookresearch/clevr-dataset-gen`



Turn 1	Turn 2	Turn 3	Turn 4
Teller: top left corner boig sun , orange part cut . right side far right medium apple tree . i see 4 apples	**Teller:** left side girl big size , running , facing right . head above horizon .	**Teller:** covering the tree , on the right side of the scene is a boy , kicking , facing left . head on green part . big size , black glasses . kicking ball .	**Teller:** make tree a size bigger , move it up and left a bit . boys hand covers trunk
Drawer: ok ready	**Drawer:** ok	**Drawer:** ok	**Drawer:** ok

Figure 6: An example CoDraw [10] images-conversation pair

and drawer text contains several spelling mistakes and we run the Bing Spell Check API[4] over the entire dataset to make corrections. For words that are not present in the train vocabulary, we use the "unk" word embedding from GloVe. We use the same train-valid-test split proposed in the original CoDraw dataset.

C Evaluation Metrics

The **object detector and localizer** is based on the Inception-v3 architecture. We modify the last layer for object detection and replace it with two heads. The first head is a linear layer with a sigmoid activation function to serve as the object detector. It is trained with binary cross-entropy loss. The second head is a linear layer where we regress all the objects' coordinates. This head is trained with L2-loss with a mask applied to only compute loss over objects that occur in the ground truth image provided in the dataset. We initialize the model using pre-trained weights trained over the ILSVRC12 (ImageNet) dataset and fine-tune on the CoDraw or i-CLEVR datasets.

Relational Similarity: To compare the arrangement of objects qualitatively, we use the above object detector/localizer to determine the type and position of objects in the ground truth and the generated image. We estimate a scene graph for each image, in which the detected objects and the image center are the vertices. The directed edges are given by the left-right and front-back relations between the vertices. To compute a relational similarity metric on scene graphs, we determine how many of the ground truth relations are present in the generated image:

$$\text{relational similarity} = \text{recall} \times \frac{|E_{G_{gen}} \cap E_{G_{gt}}|}{|E_{G_{gt}}|}$$

where 'recall' is the recall over objects detected in the generated image w.r.t objects detected in the ground truth image. $E_{G_{gt}}$ is the set of relational edges for the ground truth image corresponding to vertices common to both ground truth images and generated images, and $E_{G_{gen}}$ is the set of relational edges for the generated image corresponding to vertices common to both ground truth images and generated images. The graph similarity for the complete dataset is reported by taking the mean across time-steps of individual examples and then a mean over the entire dataset. This metric is somewhat strict as it penalizes relationships based on how the objects are positioned in the ground truth image. The instructions might allow for more plausible positions for relationships that are not explicitly mentioned in the instructions.

D Implementation Details

We add layer normalization [23] for the text encoding GRU as well as the the GRU modeling R. We add batch normalization [24] to the output of the image encoder E_c. We found that adding these normalization methods was important for gradient flow to all modalities.

[4] `https://azure.microsoft.com/en-us/services/cognitive-services/spell-check/`

For training, we used teacher forcing by using the ground truth images x_{t-1} instead of the generated image $\widetilde{x}_{t-1}$, but we use $\widetilde{x}_{t-1}$ at test time. Generated images on both i-CLEVR and CoDraw are of size 128×128 pixels. We use the Adam optimizer [25] for the GAN, with learning rates of 0.0004 for the discriminator and the 0.0001 for the generator, trained with an equal number of updates. We use RMSProp [26] for the text encoder with learning rate of 0.001, and also RMSProp for the GRU with learning rate of 0.0001.

The default hyper-parameters are $N_z = 100$, $N_c = 1024$, $K_g = 16$, $N_g = 128$, $K_d = 8$, $N_d = 256$ and $\gamma = 10$. We found $\beta = 10$ and $\lambda = 0.8$ to work the best in our experiments.