A Case for Object Compositionality in GANs

Sjoerd van Steenkiste*
Swiss AI Lab IDSIA, SUPSI, USI
sjoerd@idsia.ch

Karol Kurach
Google Brain
kkurach@google.com

Sylvain Gelly
Google Brain
sylvaingelly@google.com

Abstract

Deep generative models seek to recover the process with which the observed data was generated. Successful approaches in the domain of images rely on several core inductive biases. However, a bias to account for the compositional way in which humans structure a visual scene in terms of objects has frequently been overlooked. In this work we propose to structure the generator of a GAN to consider objects and their relations explicitly, and generate images by means of composition. We evaluate our approach on several multi-object image datasets, and find that the generator learns to identify and disentangle information corresponding to different objects at a representational level. A human study reveals that this leads to a better generative model of images.

1 Introduction

Deep generative models of images rely on the expressiveness of neural networks to learn the generative process directly from data [12, 21, 28]. Their structure is determined by the *inductive bias* of the neural network, which steers it to organize its computation to allow salient features to be recovered and ultimately captured in a representation [7, 8, 9, 21]. Recently, it has been shown that independent factors of variation, such as pose and lighting of human faces may be recovered in this way [6, 17].

A promising but under-explored inductive bias is *compositionality at the representational level of objects*, which accounts for the compositional nature of the visual world and our perception thereof [5, 27]. It allows a generative model to describe a scene as a composition of objects (entities), thereby disentangling visual information in the scene that can be processed largely independent of one another. It provides a means to efficiently learn a more accurate generative model of real-world images, and by explicitly considering objects at a representational level, it serves as an important first step in recovering corresponding object representations.

In this work we investigate object compositionality for Generative Adversarial Networks (GANs; [12]), and present a general mechanism that allows one to incorporate corresponding structure in the generator. Starting from strong independence assumptions about the objects in images, we propose two extensions that provide a means to incorporate dependencies among objects and background. Following prior work, we consider different representational slots for each object [14, 25], and a relational mechanism that preserves this separation accordingly [32]. We improve upon related work by modelling complex visual scenes that incorporate unstructured background as well as interactions among objects [10, 13, 14, 25] without relying on a priori information regarding these [2, 18, 31].

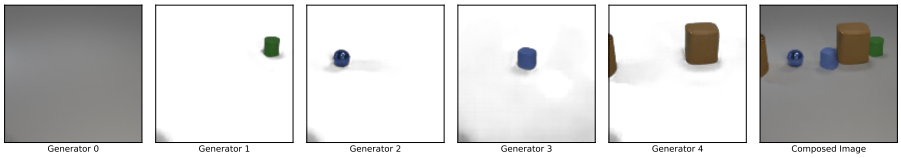


Figure 1: A scene (right) is generated as a composition of objects and background.

*Work done while an intern at Google Brain.

NIPS Workshop on Modeling the Physical World: Perception, Learning, and Control, 2018, Montréal, Canada.

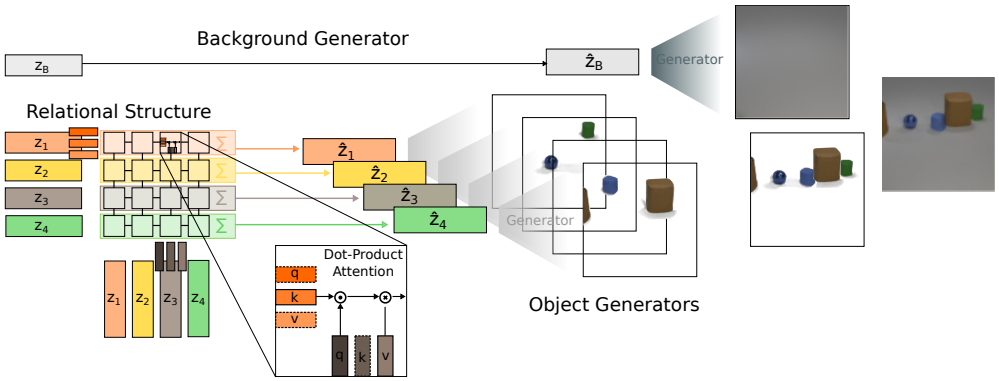


Figure 2: We propose to structure the generator of a GAN to generate images as compositions of individual objects and background. It is trained end-to-end as in Goodfellow et al. [12].

2 Incorporating Structure

In this section we propose how to *structure* the generator of a GAN to achieve object compositionality. Our approach is motivated by considering the requirements to independently vary the different visual primitives (objects) that an image is composed of.

Multiple Components If we assume that images are composed of objects that are strictly independent of one another then (without loss of generality) we may structure our latent variables accordingly. For images having K objects, we consider K i.i.d. vector-valued random variables Z_i that each describe an object at a representational level. K copies of a deterministic generator $G(\boldsymbol{z})$ transform samples from each Z_i into images, such that their superposition results in the corresponding scene:

$$G_{multi}([\boldsymbol{z}_1, \cdots, \boldsymbol{z}_K]) = \sum_{1}^{K} G(\boldsymbol{z}_i) \quad , \quad \boldsymbol{z}_i \sim P(Z) \tag{1}$$

When each copy of G generates an image of a single object, the resulting generative model efficiently describes images in $P(X)$ in a compositional manner. Each object in (1) is described in terms of the same features (i.e. the Z_i's are i.i.d) and the weights among the generators are shared, such that any acquired knowledge in generating a specific object is transferred across all others. Hence, rather than having to learn about all combinations of objects (including their individual variations) that may appear in an image, it suffices to learn about the different variations of each individual object instead.

Relational Structure In order to model relationships between objects we introduce a *relational stage*, in which the representation of an object is *updated* as a function of all others, before each generator proceeds to generate its image (see Appendix B for details).

Following Zambalid et al. [32] we consider one or more "attention blocks" to compute interactions among the object representations. At its core is Multi-Head Dot-Product Attention (MHDPA; [29]) that performs non-local computation [30] or message-passing [11] when one associates each object representation with a node in a graph. When specific design choices are made, computation of this kind provides an efficient means to learn about relations between objects and update their representations accordingly [4]. A single head of an attention block updates $\boldsymbol{z}_i$ in the following way:

$$\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i = \text{MLP}^{(\cdot)}(\boldsymbol{z}_i) \qquad \boldsymbol{A} = \underbrace{softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})}_{\text{attention weights}} \boldsymbol{V} \qquad \hat{\boldsymbol{z}}_i = \text{MLP}^{up}(\boldsymbol{a}_i) + \boldsymbol{z}_i \tag{2}$$

where $d = dim(\boldsymbol{v}_i)$ and each MLP corresponds to a multi-layer perceptron. First, a query vector $\boldsymbol{q}_i$, a value vector $\boldsymbol{v}_i$, and a key vector $\boldsymbol{k}_i$ is computed for each $\boldsymbol{z}_i$. Next, the interaction of an object i with all other objects (including itself) is computed as a weighted sum of their value vectors. Finally, the resulting update vector $\boldsymbol{a}_i$ is projected back to $dim(\boldsymbol{z}_i)$ using MLP^{up} before being added.

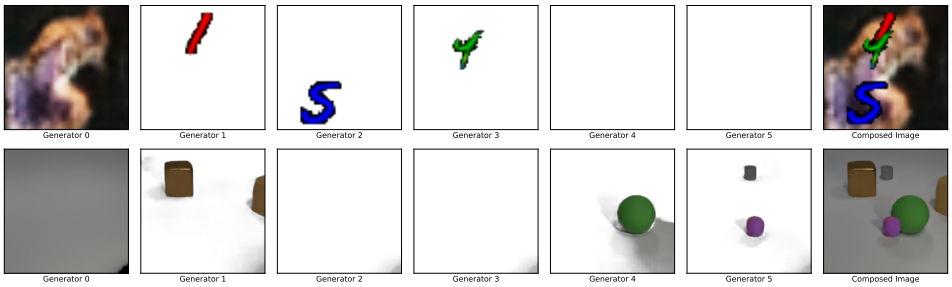


Figure 3: Generated samples by *5-GAN rel. bg.* on *CIFAR10 + MM* (top), and CLEVR (bottom). Left: the output of the background generator, middle: the outputs of each object generator, right: the composed image. Images are displayed as RGBA, with white denoting an alpha value of zero.

Incorporating Background In order to describe complex visual scenes we consider a "background" group that captures the remaining visual content in an image. The information in this group typically does not have a sufficiently regular visual appearance, or lacks the structure with which objects may be easily described. Hence, we incorporate an additional generator (see Figure 2) having its own set of weights to generate the background from a separate vector of latent variables $\boldsymbol{z}_b \sim P(Z_b)$.

Alpha Compositing A remaining challenge is in *combining* objects with background and modelling occlusion. An adaptation of the sum in (1) to incorporate pixel-level weights requires the background generator to assign a weight of zero to all pixel locations where objects appear, thereby increasing the complexity of generating the background exponentially. Instead, we require the object generators to generate an additional alpha channel for each pixel in these cases, and use alpha compositing to combine the outputs of the different generators and background through repeated application of:

$$\boldsymbol{x}_{new} = (\boldsymbol{x}_i\boldsymbol{\alpha}_i + \boldsymbol{x}_j\boldsymbol{\alpha}_j(1-\boldsymbol{\alpha}_i))/\boldsymbol{\alpha}_{new} \qquad \boldsymbol{\alpha}_{new} = \boldsymbol{\alpha}_i + \boldsymbol{\alpha}_j(1-\boldsymbol{\alpha}_i) \tag{3}$$

3 Experiments

We evaluate the proposed structure on five multi-object datasets. We denote *k-GAN ind.* to describe a generator consisting of $K = k$ components and *k-GAN rel.* if it incorporates relational structure. We append "*bg.*" if the model includes a background generator. We compare GANs that incorporate our proposed structure (*k-GAN*) to a strong GAN baseline (spanning on the order of 40-50 different configurations). Additional results, hyperparameters, and generated samples can be found in Appendices A, B & C respectively.

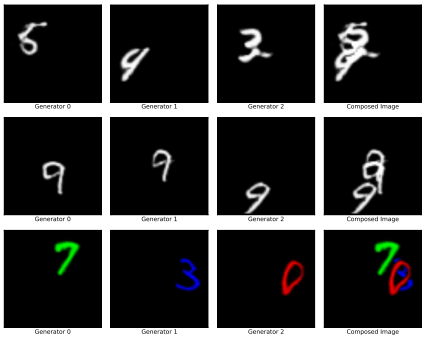
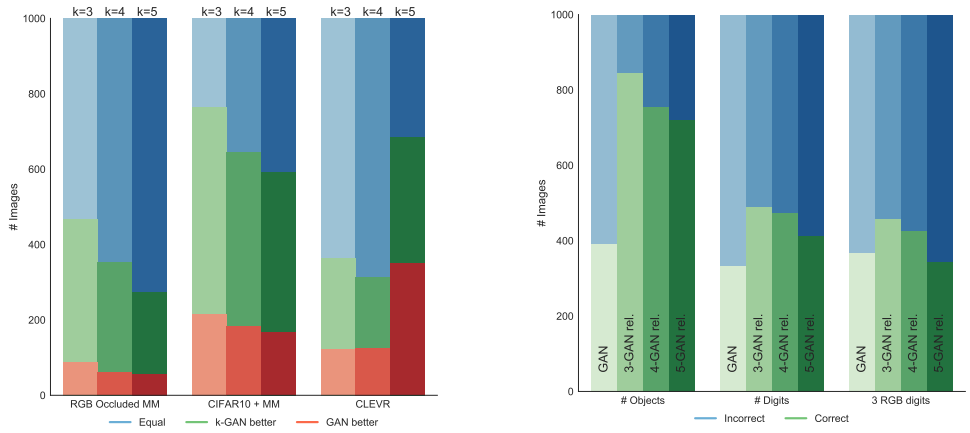


Figure 4: On the right are generated samples by *3-GAN* on Multi-MNIST: *Independent* (top), *Triplet* (middle), and *RGB Occluded* (bottom). The other columns show the output of each object generator.

Datasets We consider three variations of *Multi-MNIST (MM)*, in which each image (64×64) consists of three rescaled MNIST digits: in *Independent MM* digits are chosen randomly, *Triplet MM* requires that all digits in an image are of the same type, and *RGB Occluded MM* requires that each image consist of exactly one red, green, and blue digit. Additionally we consider *CIFAR10 + MM* in which the digits from *RGB Occluded MM* are drawn onto a randomly chosen (resized) CIFAR10 [22] image, and a processed (128×128) version of CLEVR [19].

Utilizing Structure In analyzing the output of each generator for *k-GAN*, we consistently find that the final image is generated as a composition of images consisting of individual objects and background (Figures 3 & 4). Hence, in the process of learning to generate images, *k-GAN* learns about what are individual objects, and what is background, without relying on prior knowledge or conditioning. By *disentangling* this information at the representational level, it opens the possibility to recover corresponding object representations.



(a) Comparing image quality

(b) Properties on RGB Occluded MM

Figure 5: Results of human evaluation a) comparing the quality of the generated images by *k-GAN* (k=3,4,5) to *GAN* b) Properties of generated images by *k-GAN* (k=3,4,5) and *GAN* on *RGB Occluded MM*. *k-GAN* generates better images (a) that are more faithful to the reference distribution (b).

On CLEVR, in which images may have a greater number of objects than the number of components K that was used during training, we find that the generator continues to learn a factored solution. Visual primitives are now made up of multiple objects as can be seen in Figure 3. Similarly, in analyzing generated images by *k-GAN ind.* when $k > 3$ on Multi-MNIST, we find that the generator decodes part of its latent space as "no digit" as an attempt at generating the correct number of digits.

From the generated samples in Appendix C we observe that relations among the objects are correctly captured in most cases. In analyzing the background generator we find that it sometimes generates a single object together with the background. It rarely generates more than one object, confirming that although it is capable, it is indeed more efficient to generate images as compositions of objects.

Human evaluation We asked humans to compare the images generated by *k-GAN rel.* (k=3,4,5) to our baseline on *RGB Occluded MM*, *CIFAR10 + MM* and *CLEVR*, using the configuration with a background generator for the last two datasets. For each model we select the 10 best hyper-parameter configurations (lowest FID [16]), from which we each generate 100 images. We asked up to three raters for each image and report the majority vote or "Equal" if no decision can be reached.

Figure 5a reports the results when asking human raters to compare the visual quality of the generated images by *k-GAN* to those by *GAN*. It can be seen that *k-GAN* compares favorably across all datasets, and in particular on *RGB Occluded MM* and *CIFAR10 + MM* we observe large differences. We find that *k-GAN* performs better even when $k > 3$, which can be attributed to the relational mechanism, allowing all components to agree on the correct number of digits.

In a second study we asked humans to report about properties of the generated images in order to asses if they are more faithful to the reference distribution. In Figure 5b it can be seen that *k-GAN* more frequently generates images that have the correct number of objects, number of digits, and that satisfy all properties simultaneously (color, digit count, shapes). The difference between the correct number of digits and correct number of objects suggests that the generated objects are often not recognizable as digits. This does not appear to be the case from the generated samples in Appendix C, suggesting that the raters may not have been familiar enough with the variety of MNIST digits.

4 Conclusion

We have argued for the importance of compositionality at the representational level of objects in GANs. On a benchmark of multi-object datasets we have shown that by incorporating (relational) structure *k-GAN* learns about individual objects and background in the process of synthesizing samples. A human study revealed that this leads to a better generative model of images. We are hopeful that in disentangling information corresponding to different objects at a representational level these may ultimately be recovered. Hence, we believe that this work is an important contribution towards learning object representations of complex real-world images without any supervision.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[2] Samaneh Azadi, Deepak Pathak, Sayna Ebrahimi, and Trevor Darrell. Compositional gan: Learning conditional image composition. *arXiv preprint arXiv:1807.07560*, 2018.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[5] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, page 201306572, 2013.

[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *Fifth International Conference on Learning Representations, ICLR*, 2017.

[8] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *Fifth International Conference on Learning Representations, ICLR*, 2017.

[9] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *Fifth International Conference on Learning Representations, ICLR*, 2017.

[10] SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016.

[11] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272, 2017.

[12] 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*, pages 2672–2680, 2014.

[13] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hao, Harri Valpola, and Juergen Schmidhuber. Tagger: Deep unsupervised perceptual grouping. In *Advances in Neural Information Processing Systems*, pages 4484–4492, 2016.

[14] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pages 6691–6701, 2017.

[15] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[16] 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*, pages 6626–6637, 2017.

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *Fifth International Conference on Learning Representations, ICLR*, 2017.

[18] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. *arXiv preprint*, 2018.

[19] 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 *CVPR*, 2017.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Third International Conference on Learning Representations, ICLR*, 2015.

[21] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, 2014.

[22] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[23] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The gan landscape: Losses, architectures, regularization, and normalization. *arXiv preprint arXiv:1807.04720*, 2018.

[24] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[25] Charlie Nash, SM Ali Eslami, Chris Burgess, Irina Higgins, Daniel Zoran, Theophane Weber, and Peter Battaglia. The multi-entity variational autoencoder. *NIPS Workshops*, 2017.

[26] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[27] Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

[28] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

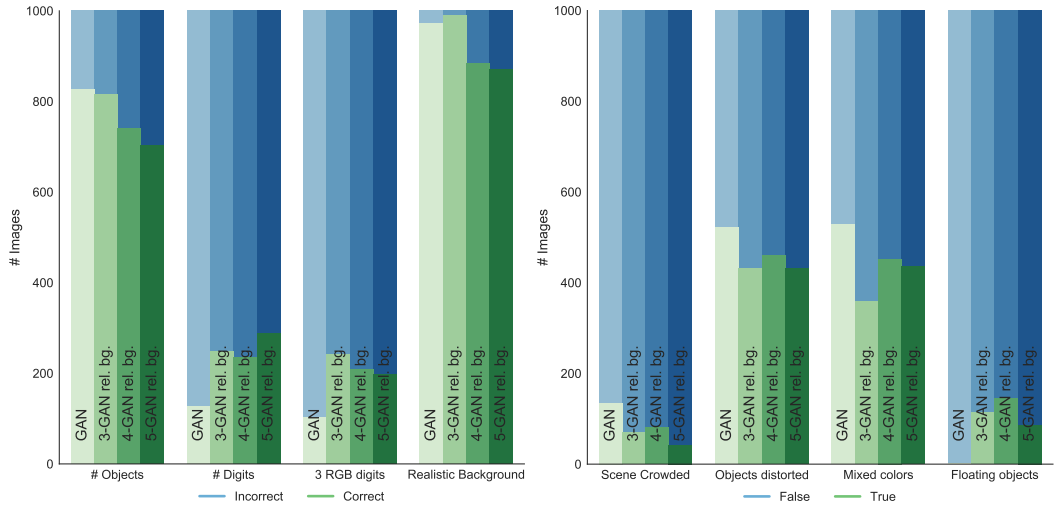
[30] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[31] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. In *Fifth International Conference on Learning Representations, ICLR*, 2017.

[32] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

A Additional Experiment Results

A.1 Human Study - Properties



(a) Properties on CIFAR10 + MM

(b) Properties on CLEVR

Figure 6: Results of human evaluation. Properties of generated images by *k-GAN* (k=3,4,5) and *GAN* on *CIFAR10 + MM* (a) and CLEVR (b). Note that on CLEVR all evaluated properties are undesirable, and thus a larger number of "False" responses is better.

On *CIFAR10 + MM* (Figure 6a) it appears that *GAN* is able to accurately generate the correct number of objects, although the addition of background makes it difficult to provide a comparison in this case. On the other hand if we look at the number of digits, then we find that *k-GAN* outperforms *GAN* by the same margin, as one would expect compared to the results in Figure 5b.

In comparing the generated images by *k-GAN* and *GAN* on CLEVR (Figure 6b) we noticed that the former generated more crowded scenes (containing multiple large objects in the center), and more frequently generated objects with distorted shapes or mixed colors. On the other hand we found cases in which *k-GAN* generated scenes containing "flying" objects, a by-product of the fixed order in which we apply (3). We asked humans to score images based on these properties, which confirmed these observations (see Figure 6b), although some differences are small.

Figure 7 reveals the distribution of the number of (geometric) objects in the images generated by *k-GAN* and *GAN* on CLEVR.

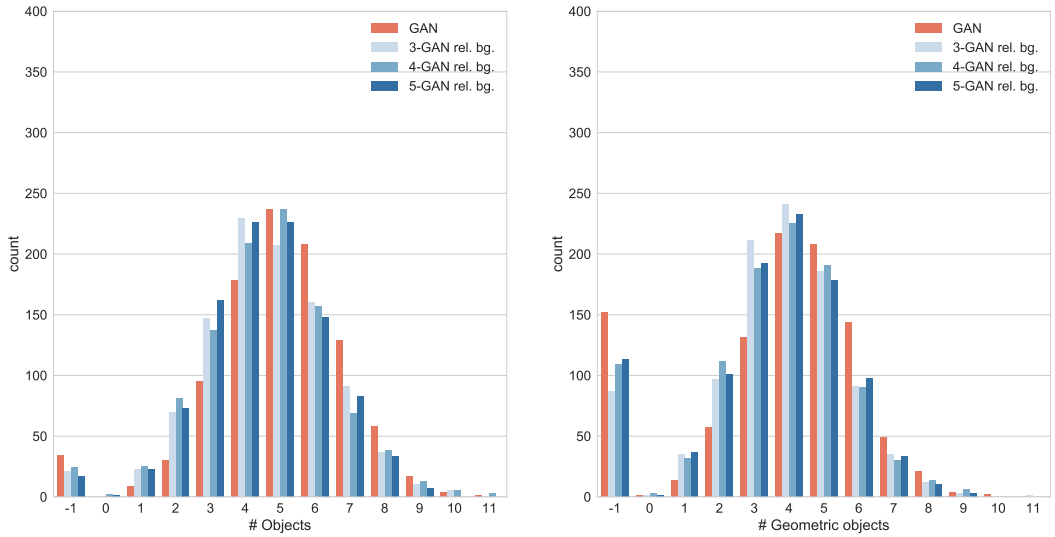
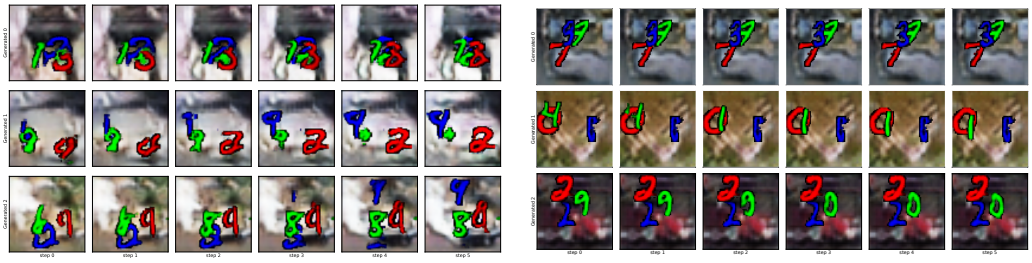


Figure 7: Results of human evaluation. Number of (geometric) objects in generated images by *k-GAN* (k=3,4,5) and *GAN* on CLEVR. A value of -1 implies a majority vote could not be reached.

A.2 Latent Traversal

We explore the degree to which the relational structure affects our initial independence assumption about objects. If it were to cause the latent representations to be fully dependent on one another then our approach would no longer be compositional in the strict sense. Note that although we have a clear intuition in how this mechanism should work, there is no corresponding constraint in the architecture. We conduct an experiment in which we traverse the latent space of a single latent vector in *k-GAN rel.*, by adding a random vector to the original sample with fixed increments and generating an image from the resulting latent vectors. Several examples can be seen in Figure 8b. In the first row it can be seen that as we traverse the latent space of a single component the blue digit 9 takes on the shape of a 3, whereas the visual presentation of the others remain unaffected. Similarly in the second and third row the green digits are transformed, while other digits remain fixed. Hence, by disentangling objects at a representational level the underlying representation is more robust to common variations in image space.

We observe this behavior for the majority of the generated samples, confirming to a large degree our own intuition of how the relational mechanism should be utilized. When we conduct the same latent traversal on the latent space of *GAN* for which the information encoding different objects is entangled, it results in a completely different scene (see Figure 8a).



(a) GAN (b) 5-GAN rel. bg.

Figure 8: Three generated images by a) *GAN* and b) *5-GAN rel. bg.*, when traversing the latent space of a single (object) generator at different increments. On the right it can be seen that in each case only a single digit is transformed, whereas the visual presentation of the others remains unaffected. In the case of GAN (left) the entire scene changes.

A.3 FID Study

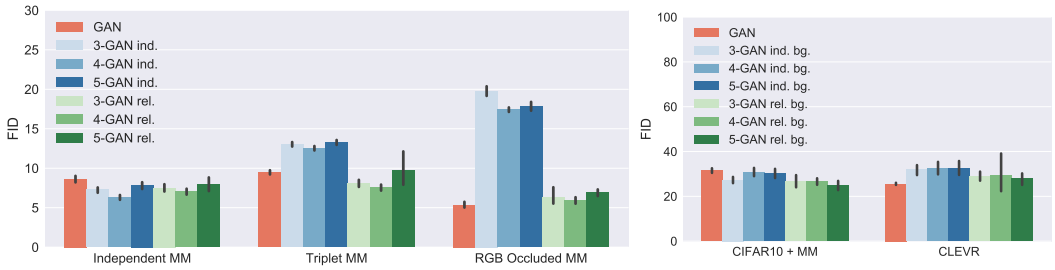


Figure 9: The best FID obtained by *GAN* and *k-GAN* following our grid search. The best configurations were chosen based on the smallest average FID (across 5 seeds). Standard deviations across seeds are illustrated with error bars.

We train *k-GAN* and *GAN* on each dataset, and compare the FID of the models with the lowest average FID across seeds. On all datasets but CLEVR we find that *k-GAN* compares favorably to our baseline, although typically by a small margin. A break-down of the FID achieved by different variations of *k-GAN* reveals several interesting observations (Figure 9). In particular, it can be observed that the lowest FID on *Independent MM* is obtained by *4-GAN* without relational structure. This is surprising as each component is strictly independent and therefore *4-GAN ind.* is unable to consistently generate 3 digits. Indeed, if we take a look at the generated samples in Figure 11, then we frequently observe that this is the case. It suggests that FID is unable to account for these properties of the generated images, and renders the small FID differences that we observed inconclusive. Figure 9 does reveal some large FID differences across the different variations of *k-GAN* on *Triplet MM*, and *RGB Occluded MM*. It can be observed that the lack of a relational mechanism on these datasets is prohibitive (as one would expect), resulting in poor FID for *k-GAN ind.* Simultaneously it confirms that the relational mechanism is properly utilized when relations are present.

A.3.1 Best configurations

Table 1 reports the best hyper-parameter configuration for each model that were obtained following our grid search. Configurations were chosen based on the smallest average FID (across 5 seeds) as reported in Figure 9. Each block corresponds to a dataset (from top to bottom: *Independent MM*, *Triplet MM*, *RGB Occluded MM*, *CIFAR10 + MM*, *CLEVR*)

model	gan type	norm.	penalty	blocks	heads	share	bg. int.	β_1	β_2	λ
GAN	NS-GAN	spec.	none	x	x	x	x	0.5	0.999	10
3-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
4-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
5-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
3-GAN rel.	NS-GAN	spec.	WGAN	1	1	no	x	0.9	0.999	1
4-GAN rel.	NS-GAN	spec.	WGAN	1	1	no	x	0.9	0.999	1
5-GAN rel.	NS-GAN	spec.	WGAN	2	1	no	x	0.9	0.999	1
GAN	NS-GAN	spec.	none	x	x	x	x	0.5	0.999	1
3-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
4-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
5-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
3-GAN rel.	NS-GAN	spec.	WGAN	1	1	no	x	0.9	0.999	1
4-GAN rel.	NS-GAN	spec.	WGAN	1	2	no	x	0.9	0.999	1
5-GAN rel.	NS-GAN	spec.	WGAN	2	1	no	x	0.9	0.999	1
GAN	NS-GAN	spec.	none	x	x	x	x	0.5	0.999	1
3-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
4-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
5-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
3-GAN rel.	NS-GAN	none	WGAN	2	2	yes	x	0.9	0.999	1
4-GAN rel.	NS-GAN	none	WGAN	2	2	no	x	0.9	0.999	1
5-GAN rel.	NS-GAN	none	WGAN	2	2	yes	x	0.9	0.999	1
GAN	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
3-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
4-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
5-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
3-GAN rel. bg.	NS-GAN	none	WGAN	2	1	yes	yes	0.9	0.999	1
4-GAN rel. bg.	NS-GAN	none	WGAN	2	1	yes	yes	0.9	0.999	1
5-GAN rel. bg.	NS-GAN	none	WGAN	2	2	yes	no	0.9	0.999	1
GAN	WGAN	none	WGAN	x	x	x	x	0.9	0.999	1
3-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
4-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
5-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
3-GAN rel. bg.	NS-GAN	none	WGAN	2	1	no	yes	0.9	0.999	1
4-GAN rel. bg.	NS-GAN	none	WGAN	1	2	no	no	0.9	0.999	1
5-GAN rel. bg.	NS-GAN	none	WGAN	2	2	no	no	0.9	0.999	1

Table 1: The best hyper-parameter configuration for each model as were obtained after conducting a grid search. Configurations were chosen based on the smallest average FID (across 5 seeds).

B Experiment Details

B.1 Model specifications

The generator and discriminator neural network architectures in all our experiments are based on DCGAN [26].

Object Generators *k-GAN ind.* introduces $K = k$ copies of an object generator (i.e. tied weights, DCGAN architecture) that each generate and image from an independent sample of a 64-dimensional *UNIFORM(-1, 1)* prior $P(Z)$.

Relational Structure When a relational stage is incorporated (*k-GAN rel.*) each of the $\boldsymbol{z}_i \sim P(Z)$ are first updated, before being passed to the generators. These updates are computed using one or more *attention blocks* [32], which integrate Multi-Head Dot-Product Attention (MHDPA; [29]) with a post-processing step. A single head of an attention block updates $\boldsymbol{z}_i$ according to (2).

In our experiments we use a single-layer neural network (fully-connected, 32 ReLU) followed by LayerNorm [3] for each of MLP^{query}, MLP^{key}, MLP^{value}. We implement MLP^{up} with a two-layer neural network (each fully-connected, 64 ReLU), and apply LayerNorm after summing with $\boldsymbol{z}_i$. Different heads in the same block use different parameters for MLP^{query}, MLP^{key}, MLP^{value}, MLP^{up}. If multiple heads are present, then their outputs are concatenated and transformed by a single-layer neural network (fully-connected, 64 ReLU) followed by LayerNorm to obtain the new $\hat{\boldsymbol{z}}_i$. If the relational stage incorporates multiple attention blocks that iteratively update $\boldsymbol{z}_i$, then we consider two variations: using unique weights for each MLP in each block, or sharing their weights across blocks.

Background Generation When a background generator is incorporated (eg. *k-GAN rel. bg*) it uses the same DCGAN architecture as the object generators, yet maintains its own set of weights. It receives as input its own latent sample $\boldsymbol{z}_b \sim P(Z_b)$, again using a *UNIFORM(-1, 1)* prior, although one may in theory choose a different distribution. We explore both variations in which $\boldsymbol{z}_b$ participates in the relational stage, and in which it does not.

Composing In order to obtain the final generated image, we need to combine the images generated by each generator. In the case of *Independent MM* and *Triplet MM* we simply sum the outputs of the different generators and clip their values to $(0, 1)$. On *RGB Occluded MM* we combine the different outputs using alpha compositing, with masks obtained by thresholding the output of each generator at 0.1. On *CIFAR10 + MM* and *CLEVR* we require each of the object generators to generate an additional alpha channel by adding an additional feature map in the last layer of the generator. These are then combined with the generated background (opaque) using alpha compositing, i.e. through repeated application of (3).

B.2 Hyperparameter Configurations

Each model is optimized with ADAM [20] using a learning rate of 0.0001, and batch size 64 for $1\,000\,000$ steps. Each generator step is followed by 5 discriminator steps, as is considered best practice in training GANs. Checkpoints are saved at every $20\,000^{th}$ step and we consider only the checkpoint with the lowest FID [16] for each hyper-parameter configuration. FID is computed using $10\,000$ samples from a hold-out set.

Baseline We conduct an extensive grid search over 48 different GAN configurations to obtain a strong GAN baseline on each dataset. It is made up of hyper-parameter ranges that were found to be successful in training GANs on standard datasets [23].

We consider [SN-GAN [12] / WGAN [1]], using [NO / WGAN [15]] gradient penalty with λ [1 / 10]. In addition we consider these configurations [WITH / WITHOUT] spectral normalization [24]. We consider [(0.5, 0.9) / (0.5, 0.999) / (0.9, 0.999)] as (β_1, β_2) in ADAM. We explore 5 different seeds for each configuration.

k-GAN We conduct a similar grid search for the GANs that incorporate our proposed structure. However, in order to maintain a similar computational budget to our baseline we consider a *subset*

of the previous ranges to compensate for the additional hyper-parameters of the different structured components that we would like to search over.

In particular, we consider SN-GAN with WGAN gradient penalty, with a default λ of 1, [WITH / WITHOUT] spectral normalization. We use (0.9, 0.999) as fixed values for (β_1, β_2) in ADAM. Additionally we consider K = [3 / 4 / 5] copies of the generator, and the following configurations for the relational structure:

- Independent
- Relational (1 block, no weight-sharing, 1 head)
- Relational (1 block, no weight-sharing, 2 heads)
- Relational (2 blocks, no weight-sharing, 1 head)
- Relational (2 blocks, weight-sharing, 1 head)
- Relational (2 blocks, no weight-sharing, 2 heads)
- Relational (2 blocks, weight-sharing, 2 heads)

This results in 42 hyper-parameter configurations, for which we each consider 5 seeds. We do not explore the use of a background generator on the non-background datasets. Correspondingly, we *only* explore variations that incorporate the background generator on the background datasets. In the latter case we search over an additional hyper-parameter that determines whether the latent representation of the background generator should participate in the relational stage or not.

B.3 Human Study

We asked human raters to compare the images generated by *k-GAN* ($k = 3, 4, 5$) to our baseline on *RGB Occluded MM*, *CIFAR10 + MM* and *CLEVR*, using the configuration with a background generator for the last two datasets. For each model we select the 10 best hyper-parameter configurations, from which we each generate 100 images. We conduct two different studies 1) in which we compare images from *k-GAN* against *GAN* and 2) in which we asked raters to answer questions about the content (properties) of the images.

Comparison We asked reviewers to compare the quality of the generated images. We asked up to three raters for each image and report the majority vote or "none" if no decision can be reached.

Properties For each dataset we asked (up to three raters for each image) the following questions.

On *RGB Occluded MM* we asked:

1. How many [red, blue, green] shapes are in the image? Answers: [0, 1, 2, 3, 4, 5]
2. How many are recognizable as digits? Answers: [0, 1, 2, 3, 4, 5]
3. Are there exactly 3 digits in the picture, one of them green, one blue and one red? Answers: Yes / No

On *CIFAR10 + MM* we asked these same questions, and additionally asked:

4. Does the background constitute a realistic scene? Answers: Yes / No

On *CLEVR* we asked the following set of questions:

1. How many shapes are in the image? Answers: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2. How many are recognizable as geometric objects? Answers: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3. Are there any objects with mixed colors (eg. part green part red)? Answers: Yes / No
4. Are there any objects with distorted geometric shapes?: Answers: Yes / No
5. Are there any objects that appear to be floating? Answers: Yes / No
6. Does the scene appear to be crowded? Answers: Yes / No

C Overview of Real and Generated Samples

Generated samples are reported for the best (lowest FID) structured model, as well as the best baseline model for each dataset.

C.1 Independent Multi MNIST

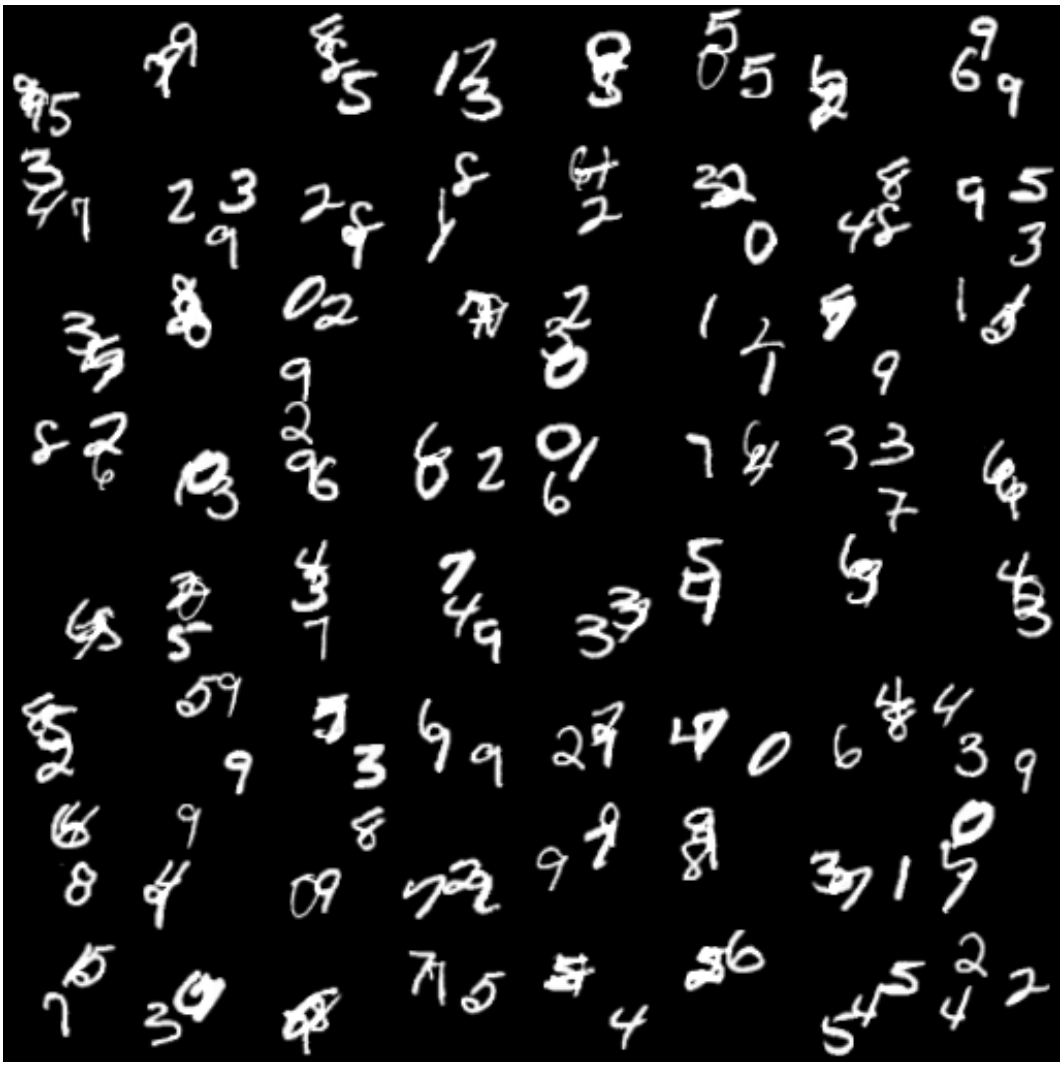


Figure 10: Real

C.1.1 Structured GAN

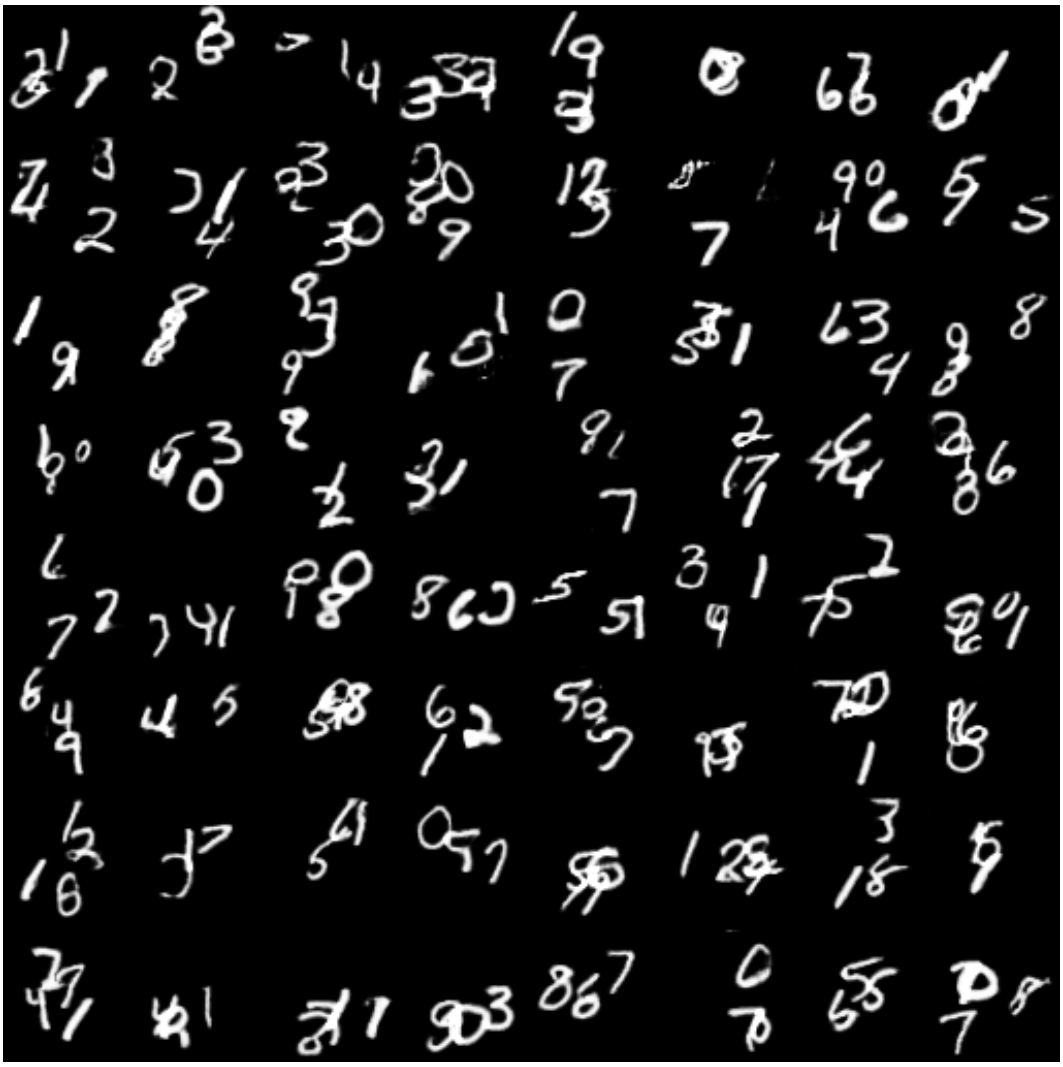


Figure 11: 4-GAN ind. with spectral norm and WGAN penalty

C.1.2 GAN

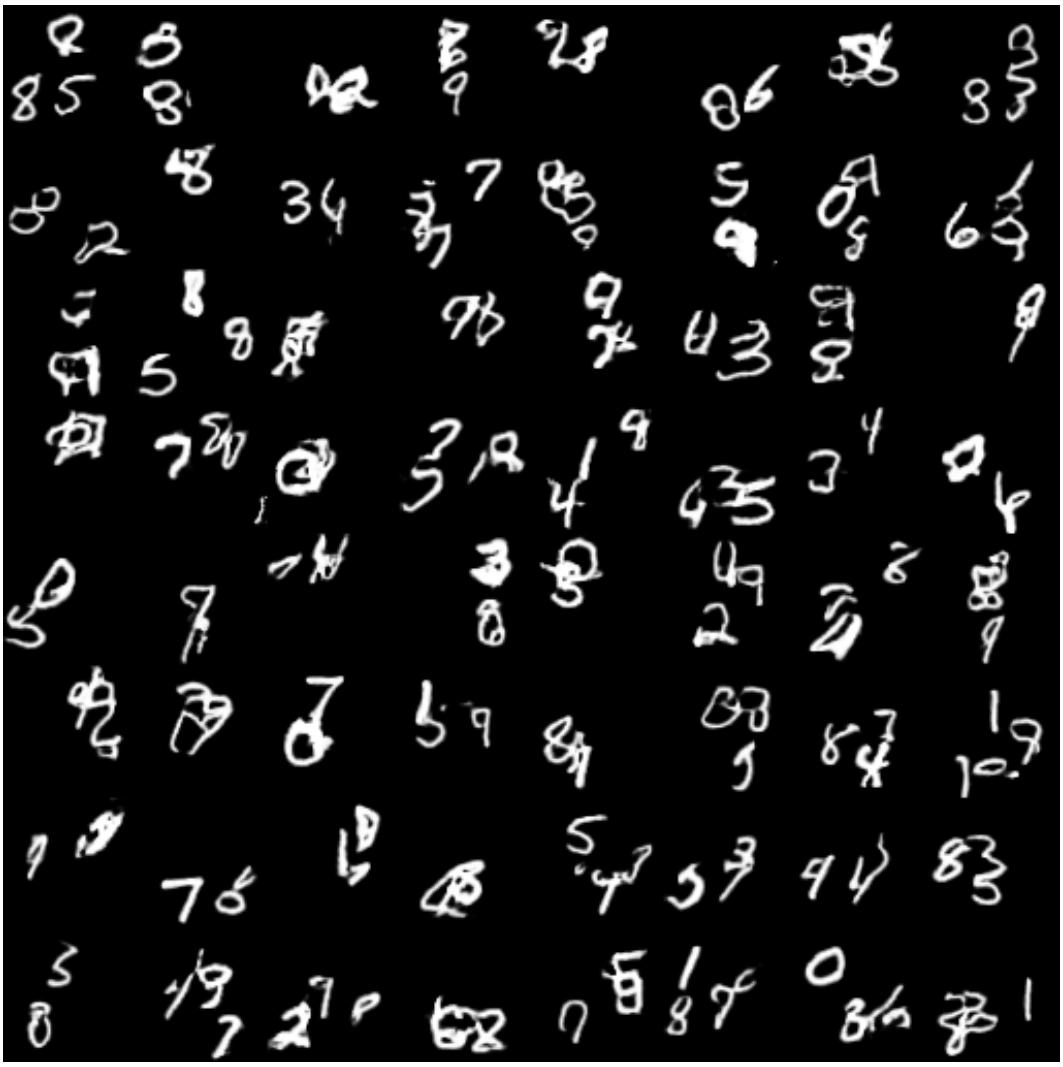


Figure 12: NS-GAN with spectral norm

C.2 Triplet Multi MNIST

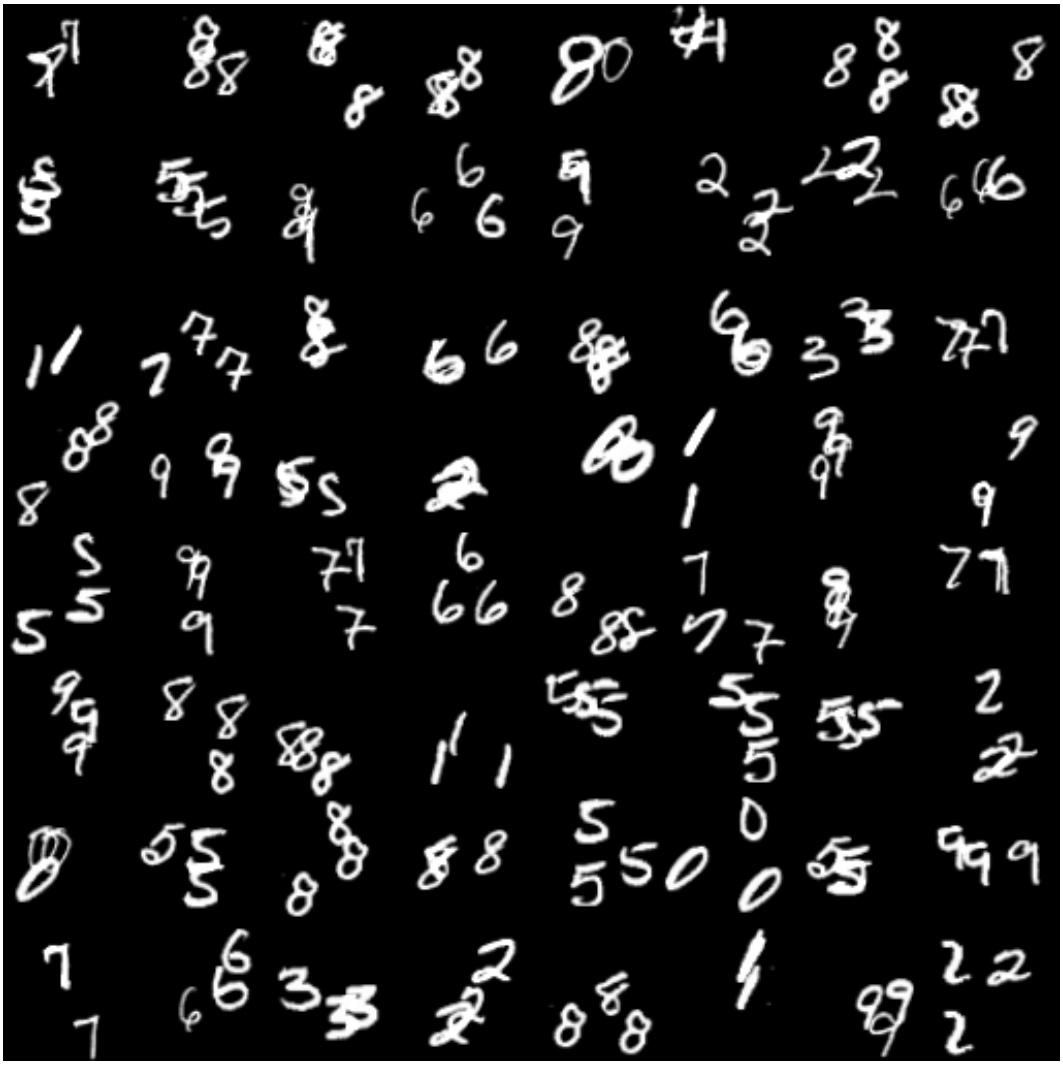


Figure 13: Real

C.2.1 Structured GAN

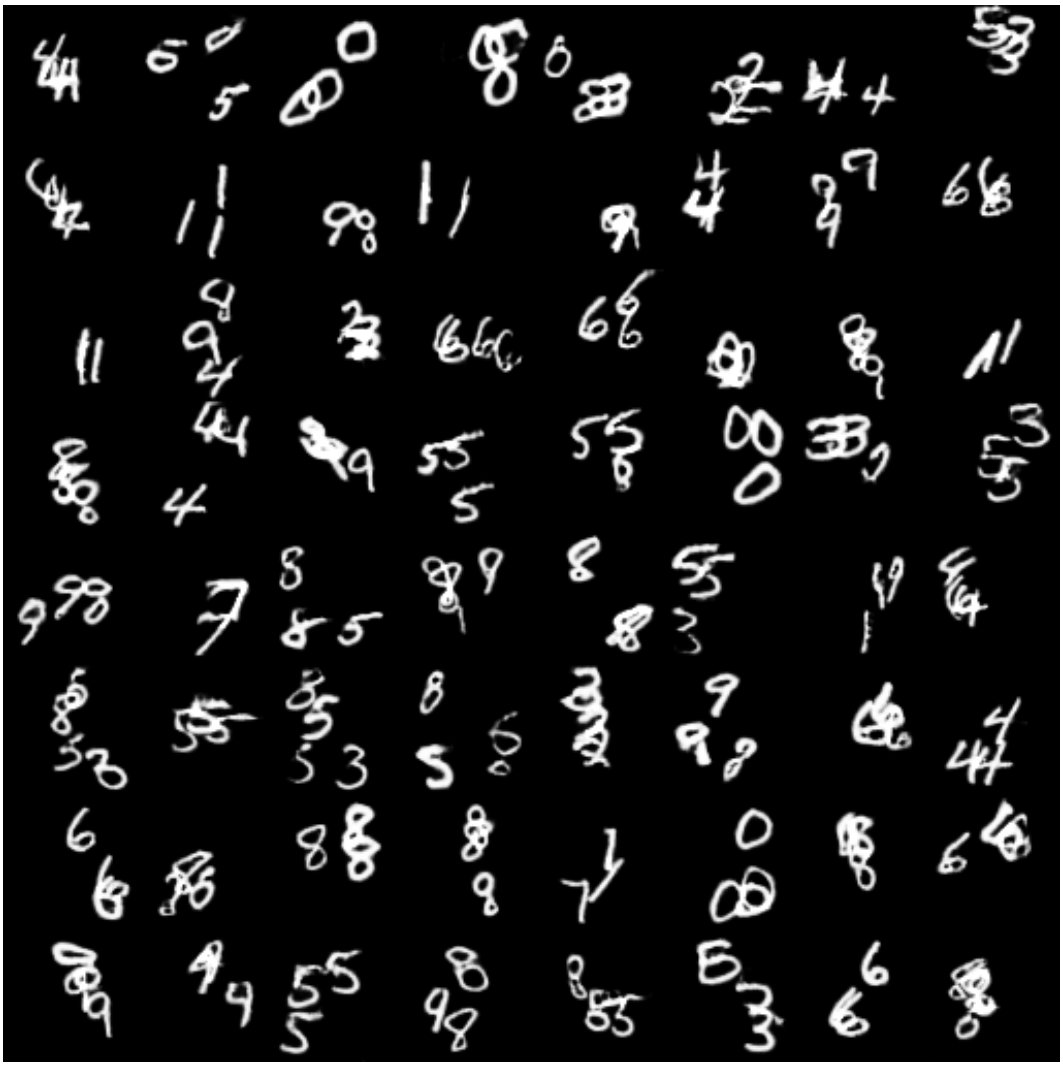


Figure 14: 4-GAN rel. (1 block, 2 heads, no weight sharing) with spectral norm and WGAN penalty

C.2.2 GAN

Figure 15: NS-GAN with spectral norm

C.3 RGb-Occluded Multi MNIST

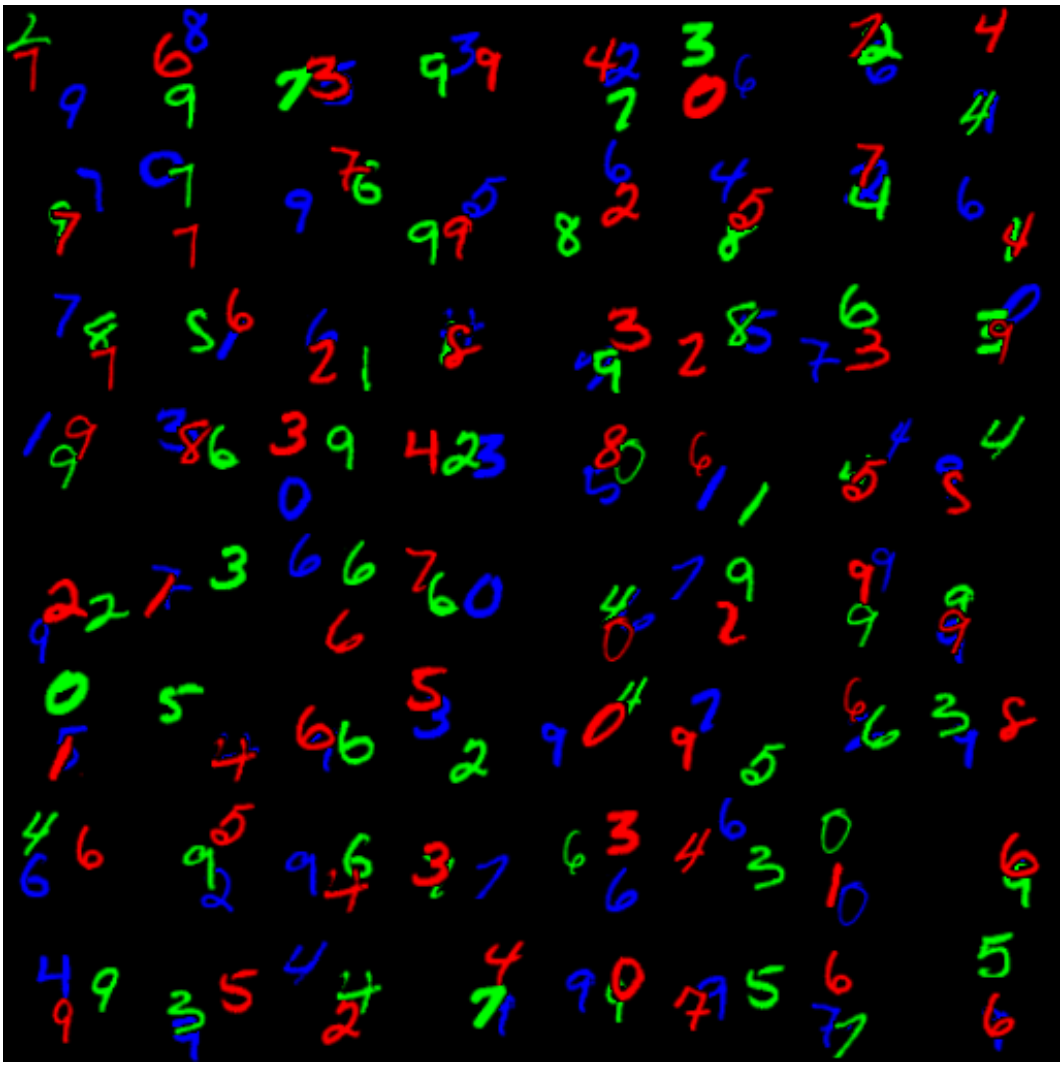


Figure 16: Real

C.3.1 Structured GAN

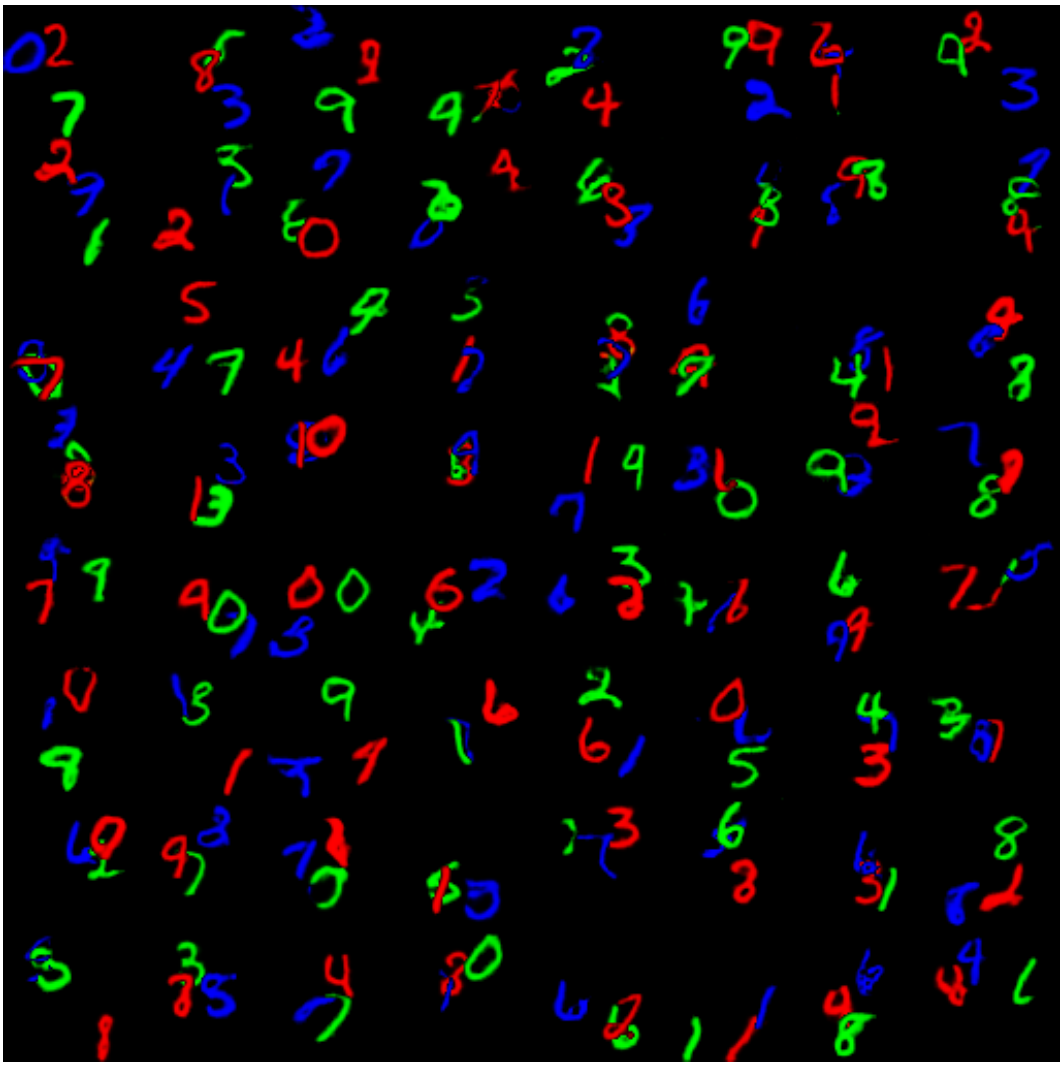


Figure 17: 3-GAN rel. (2 blocks, 2 heads, no weight sharing) with spectral norm and WGAN penalty

C.3.2 GAN

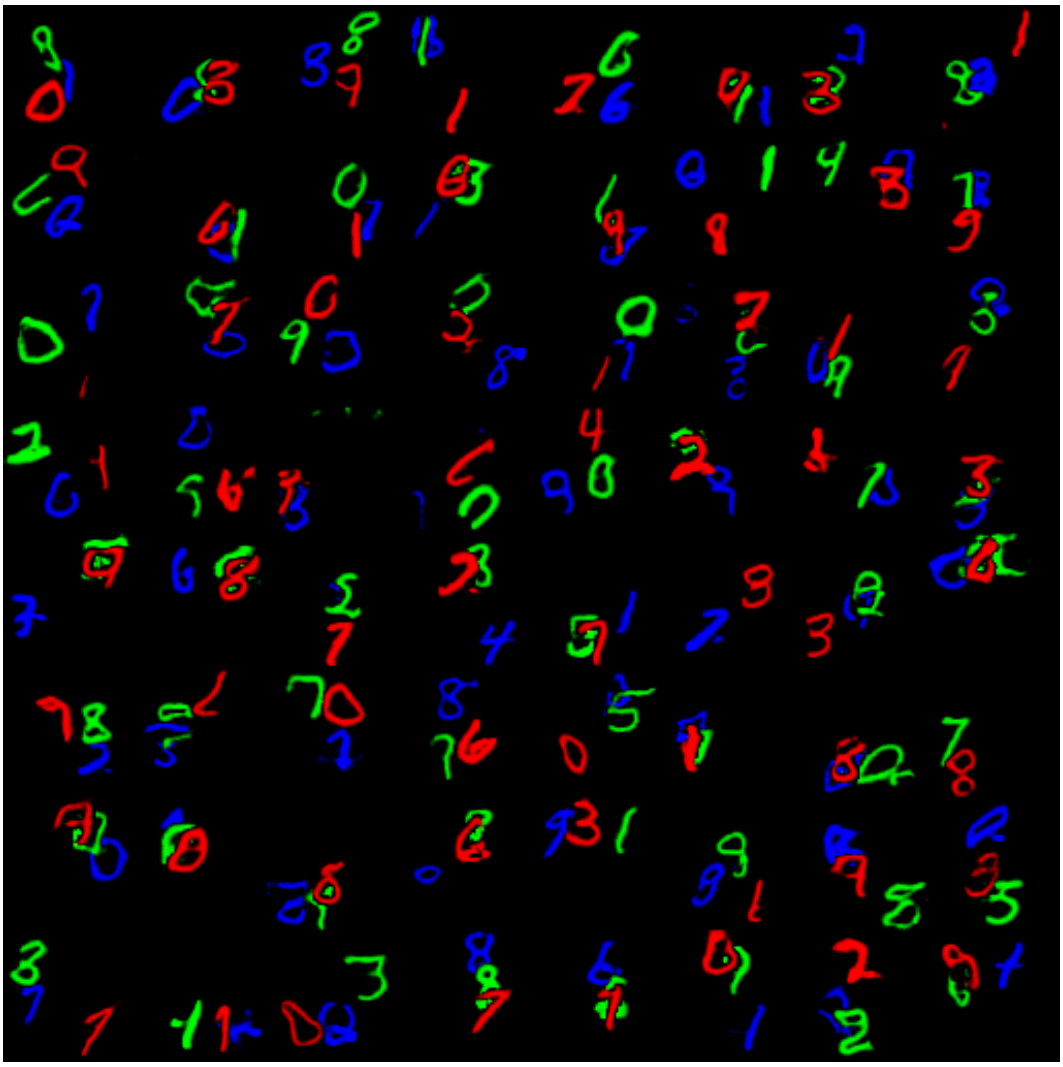


Figure 18: NS-GAN with spectral norm

C.4 RGB-Occluded Multi MNIST + CIFAR10

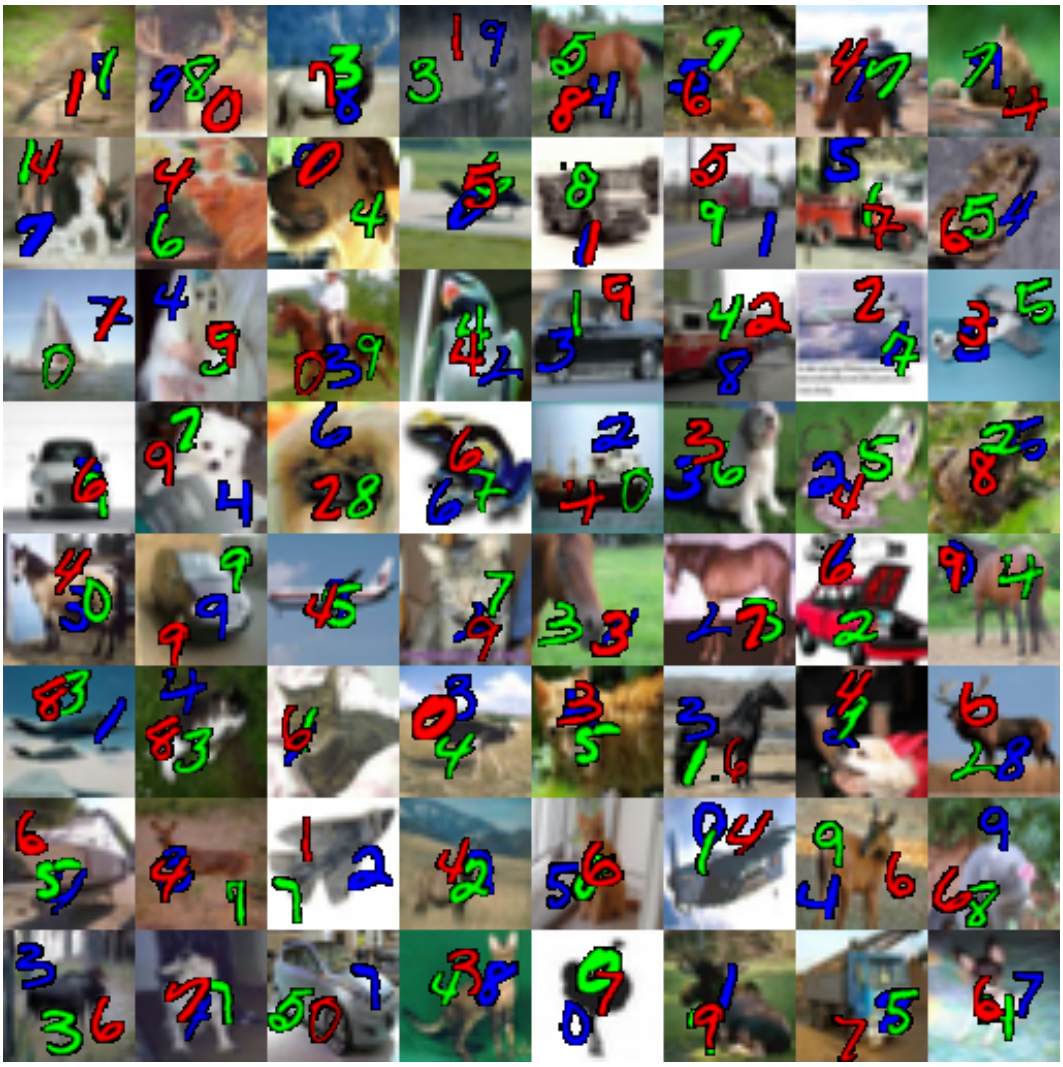


Figure 19: Real

C.4.1 Structured GAN

Figure 20: 5-GAN rel. (1 block, 2 heads, no weight sharing) bg. (no interaction) with WGAN penalty

C.4.2 GAN

Figure 21: WGAN with WGAN penalty

C.5 CLEVR

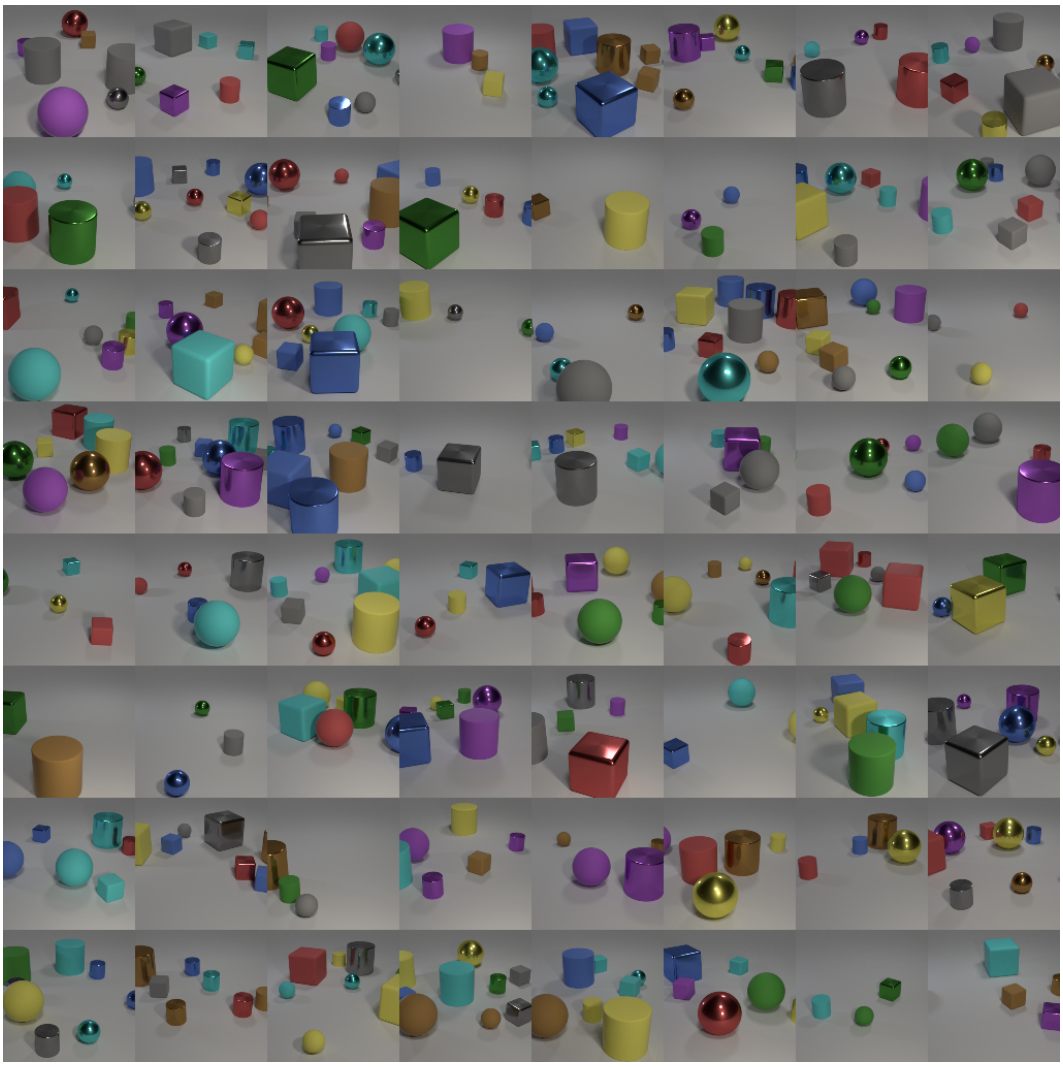


Figure 22: Real

C.5.1 Structured GAN

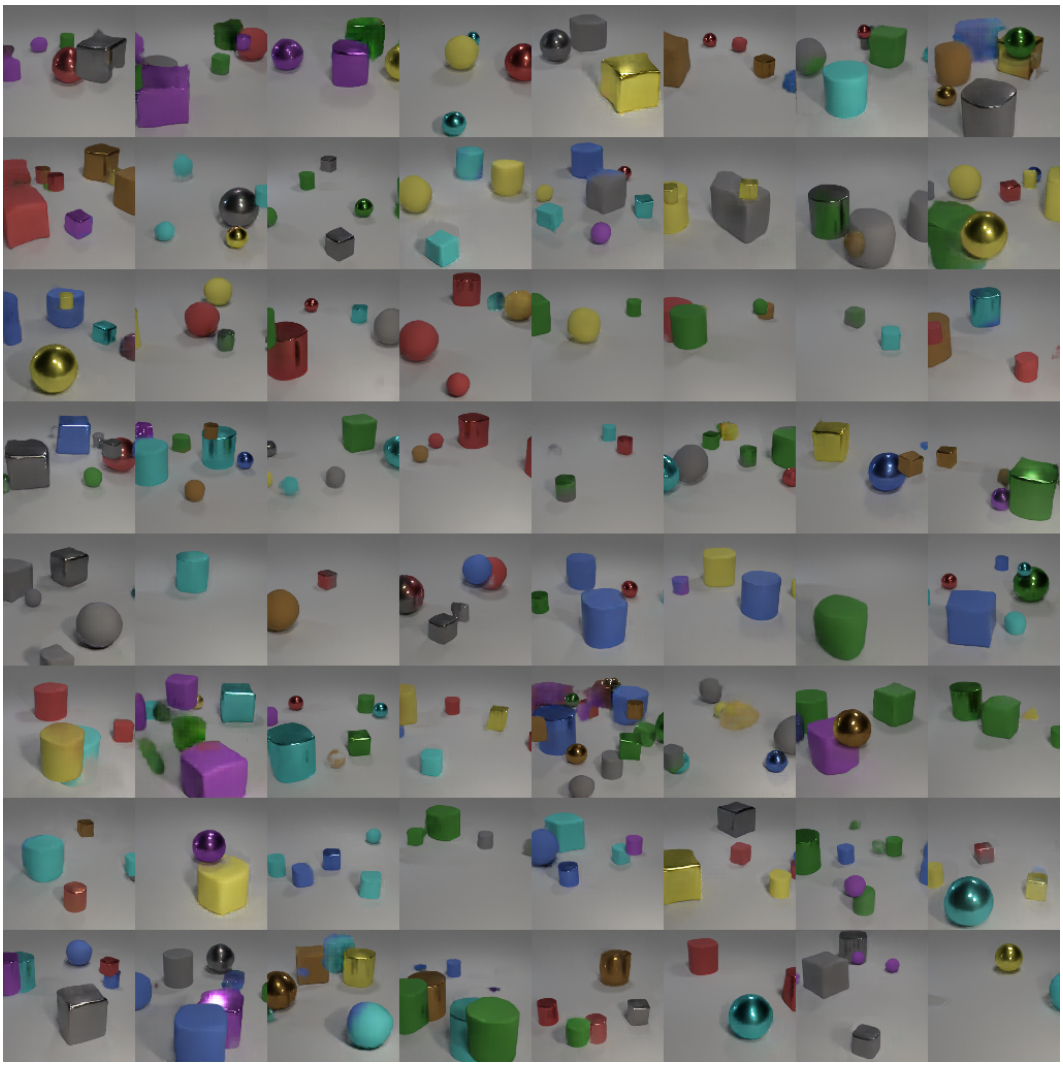


Figure 23: 3-GAN rel. (2 heads, 2 blocks, no weight sharing) bg. (with interaction) with WGAN penalty.

C.5.2 GAN

Figure 24: WGAN with WGAN penalty