LEARNING WITH MENTAL IMAGERY

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we propose deep convolutional generative adversarial networks (DCGAN) that learn to produce a "mental image" of the input image as internal representation of a certain category of input data distribution. This mental image is what the DCGAN 'imagines' that the input image might look like under ideal conditions. The mental image contains a version of the input that is iconic, without any peculiarities that do not contribute to the ideal representation of the input data distribution within a category. A DCGAN learns this association by training an encoder to capture salient features from the original image and a decoder to convert salient features into its associated mental image representation. Our new approach, which we refer to as a Mental Image DCGAN (MIDCGAN), learns features that are useful for recognizing entire classes of objects, and that this in turn has the benefit of helping single and zero shot recognition. We demonstrate our approach on object instance recognition and handwritten digit recognition tasks.

1 INTRODUCTION

Deep convolutional neural networks have had a revolutionary impact on machine learning and computer vision, yet we still fall dramatically short when it comes to learning in a manner that is most similar to people. Consider the way that children interact with objects when they are very young (James et al., 2014; Yu et al., 2009): during their interaction, children look at objects from many different perspectives. Eventually they build up a preference for certain viewpoints after examining objects for a long period of time. In this paper, we consider the question of what would happen if we were to train a deep convolutional generative adversarial network in the same manner. For this, we provide the mental image (i.e., an ideal representation), and then provide samples of many different variations of the input image. The Mental Image DCGAN (MIDCGAN) is trained to associate each of these samples in a specific input distribution back to the mental image. This association is learned using a GAN architecture (see Fig. 2) with a generator composed of an encoder and decoder. MIDCGAN trains the encoder to learn salient bottleneck features for each class while the decoder learns to generate a mental image from bottleneck features. We will show that MIDCGAN bottleneck features are better suited for learning than those features that are generated without the benefit of using a mental image.

Stated more formally, a typical learning task seeks to learn the data distribution, $p(x)$ mapping to a class or category label y or $p(y|x)$. The diversity of samples in the training data are limiting in the way the learner represents the category class internally. The MIDCGAN approach on the other hand provides a mental image $\tilde{x}$ as target to be learned and stored as representation of a category target distribution. The learner maps the input data distribution, $p(x)$, to this canonical representation of the category, $\tilde{x}$, i.e $p(\tilde{x}|x)$. During this mapping process, the MIDCGAN creates an internal bottleneck feature vector that is best representative of the input distribution mapping to the mental image.

We demonstrate the effectiveness of mental image DCGANs (MIDCGAN) on two different problems. First, we demonstrate this for handwritten digits as proof of concept. In this case, we assume that a helpful tutor has provided an ideal representation of the digit, so the mental image is a stencil (Fig. 3. When the MIDCGAN sees a digit, it is trained to think of how it might look if it were looking at the stencil. We mainly demonstrate the performance of MIDCGAN on instance based object recognition. In this case, the helpful tutor provides the system with an iconic view of the object. The MIDCGAN observes the objects from different viewpoints, but at each time it is trained to think of how the object might look like if it were looking at the object from an ideal or iconic perspective.

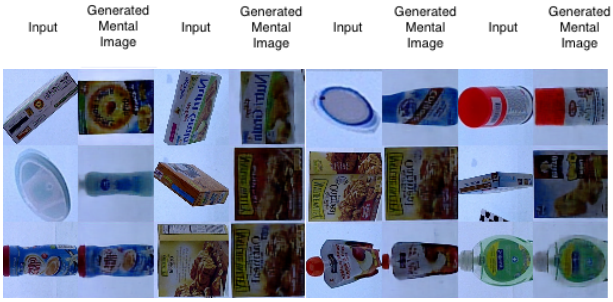

Figure 1: MIDCGAN input (left) and generated mental image (right). The images are selected from the BigBird database (Singh et al., 2014).

We evaluate this in three different ways. First, we evaluate quantitatively the usefulness of the bottleneck features from MIDCGAN on learning tasks (comparing to DCGAN features trained without a mental image). Second, we evaluate qualitatively MIDCGAN's ability to generate mental images. Finally, we evaluate MIDCGAN's ability to perform few shot recognition on objects whose mental image was not learned or transfer learning. More precisely how does MIDCGAN perform when asked to imagine what an object in its frontal view, when it has never seen the object before.

2 Related work

Original GAN and Derivatives: After the introduction of the original generative adversarial networks (GAN) (Goodfellow et al., 2014a), several architectures and improvements on training GANs were proposed such as improved GAN training (Salimans et al., 2016), categorical GAN Springenberg (2015), and InfoGAN (Chen et al., 2016).

Motivation to MIDCGAN: MIDCGAN is inspired partly by the work of image-to-image mapping (Isola et al., 2016) and image editing (Wang & Gupta, 2016) and inpainting using GANs (Pathak et al., 2016). Zhu et al. (Zhu et al., 2016b) also explored mapping 3D target from multiple 2D projections. We propose mapping a canonical target image from a single viewpoint from various inputs in terms of pose-invariance or style.

Partial Supervision with GAN for classification: GANs have been employed to learn expressive features for classification in unsupervised (Radford et al., 2015) and semi-supervised manner (Salimans et al., 2016) (Springenberg, 2015).

Rotation and Pose Invariance: Another closely related topic is invariance to Affine transforms. Spatial transformer networks (Jaderberg et al., 2015) combine localization and transformation layers to provide invariance. However, MIDCGAN learns this invariance implicitly by learning to map samples to canonical representation of the distribution. Pose invariance learning was also demonstrated (LeCun et al., 2004) on the NORB dataset. MIDCGAN handles pose invariance learning and exploits the canonical mapping to learn expressive features for either object class and instance inference without any constraint on the view point of objects.

Single-Shot Recognition: Single shot learning is of particular importance where there is limited training data. Held et al.(Held et al., 2016) proposed single-shot deep learning through multi-phase supervised pre-training. We approach the problem from a different perspective, where we map an instance of an object back to a canonical viewpoint using MIDCGAN. This allows single shot recognition performance on par to the pre-trained ones without actually being pre-trained on a larger similar dataset.

3 Methodology

DCGAN learns generative models using the joint adversarial training of a generator network and discriminator network. The generator maps a noise vector z to generated image using $\hat{x} = G_{\theta^G}(z)$ that increasingly represents the underlying target distribution $p(x)$ during training. The discrim-

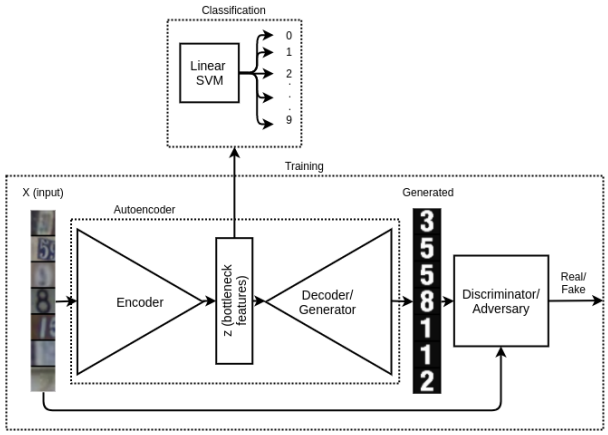

Figure 2: The MIDCGAN architecture built with either feedforward convolutional or ResNet blocks.

inator predicts whether a sample is from the real target data distribution, $p(x)$, or the generated data distribution, $p(\hat{x})$, and is represented by $D_{\theta D}(x, \hat{x})$ (Salimans et al., 2016). For the sake of simplicity, we refer to the generator as $G(z)$ and the discriminator as $D(x)$.

DCGANs have been applied to a variety of domains including realistic image generation, unsupervised and semi-supervised learning (Radford et al., 2015), and image-to-image translation (Isola et al., 2016). We have built on the success of these applications with MIDCGAN, training the generative network using both l_2 loss and the adversarial discriminative loss so that it learns to map sample inputs from the training set to the associated mental image.

Another important basic distinction between regular DCGAN and MIDCGAN is that the generator network is not generating samples from a random noise vector, z, rather they are derived from the encoder of an autoencoder (Pathak et al., 2016). This enables MIDCGAN to encode important features from the input distribution, $p(x)$, into a bottleneck feature vector denoted as $\bar{z}$. This feature vector is later used for learning as in 3.3.

3.1 ARCHITECTURE

The MIDCGAN architecture is based on a GAN architecture with the generator replaced by a full autoencoder (i.e., encoder-decoder) (Pathak et al., 2016). Given an input selected from a distribution $p(x)$, the encoder network produces a representation of the object in the form of a bottleneck feature vector, $\bar{z}_n$, of length n. The decoder network then takes the output of the encoder, $\bar{z}_n$, and produces the a sample from the generated distribution, $p(\hat{\tilde{x}})$. Here, $\tilde{x}$ is the mental image of x.

The goal of MIDCGAN is to associate the input image back to the mental image. For this, the encoder network focuses on emphasizing weights that are meaningful to map the input sample to the mental image and in the process the bottleneck learns the essential features that are useful in recognition tasks. The novel introduction of the mental image forces the network to focus on features that are common to most samples of that specific category. The network learns features that could be useful for classification implicitly without getting distracted to map the input peculiarities that are not important for classification. In this, we take the semantics of the image that have been extracted by the encoding function and reproduce what the image looks like from what an ideal perspective (i.e., a mental image). To train, the generated mental image is given to the discriminator together with the true mental images we want the network to produce to form the joint reconstruction and adversarial loss discussed in Section 3.2.

Fig. 2 shows an overview of the MIDCGAN architecture. The convolutional blocks in all the three components of MIDCGAN (encoder, decoder and discriminator) take two forms: either simple convolutional blocks that contain a sequence of convolution-batch normalization-activation (we call this simple network) or n residual blocks. We use $n = 5$ double convolutional BN-activation-convolution residual units (He et al., 2016) in each of the convolution blocks. Our experiments compare these different architectures. The discriminator is of two types: the regular AlexNet style

discriminator (similar to most DCGANs) and a discriminator that has a feature extraction pipeline that is similar to the encoder with an added discriminator layer (we call this separate discriminator).

3.2 THE JOINT AUTOENCODER AND ADVERSARIAL LOSS

MIDCGAN is trained using joint adversarial and generator l_2 losses. In a regular autoencoder, the input and the target are the same and since there is no guidance for the encoder to select a specific mode of the multimodal underlying data distribution, $p(x)$. Therefore, the autoencoder tends to average modes resulting in a blurry average image (Pathak et al., 2016). By adding the mental image as a target in the l_2 loss component, it forces the network to produce a specific mode of the target distribution, $p(\tilde{x})$, instead (Goodfellow et al., 2014a; Pathak et al., 2016). Therefore, MIDCGAN generates images that are sharper and closer to the target earlier in the training than an equivalent regular DCGAN while at the same time learning useful classification features via its mapping of the input sample to the mental image. The l_2 loss is mainly responsible for the mapping of the input to the canonical mental image while the adversarial loss contributes additional supervision of how close the mapping is to the canonical image of that category. The mental image and the adversarial loss together force the network to choose a particular mode (close to real mental image) of the target distribution, $p(\tilde{x})$.

3.2.1 AUTOENCODER RECONSTRUCTION LOSS

Here we used a normalized l_2 loss similar to (Pathak et al., 2016) without the masking. We have also experimented with both l_1 and l_2 losses and saw no significance difference. The l_2 loss is given by the squared difference between the target mental image, $\tilde{x}$, and the generated mental image, $G(z)$. Substituting encoded features from input data distribution, $p(x)$, for z using the encoder, $E(x)$ results in Eq. 1

$$L_{l_2}(x, \tilde{x}) = \mathop{\mathbb{E}}_{x \in p(x), \tilde{x} \in p(\tilde{x})} ||\tilde{x} - G(E(x))||_2^2 \tag{1}$$

3.2.2 ADVERSARIAL LOSS

According to (Goodfellow et al., 2014a), the regular adversarial two-player minmax game between the discriminator, $D(x)$, and the generator, $G(z)$ is given by Eq. 2.

$$\mathop{minmax}_{G \quad D} \mathop{\mathbb{E}}_{x \in p(x)} [log(D(x))] + \mathop{\mathbb{E}}_{z \in p(z)} [log(1 - D(G(z)))] \tag{2}$$

The discriminator maximizes both terms while the generator minimizes the second term. There are several things to consider with respect to adversarial loss and MIDCGAN. First, the input distribution, $p(x)$ and the target distribution, $p(\tilde{x})$, are not the same due to the mental image mapping in MIDCGAN and hence the first term of Eq. 2 is maximized by the discriminator using samples from the real target distribution or mental image, $p(\tilde{x})$. There is also no prior distribution of the bottleneck. In a regular GAN, z is sampled from a prior noise distribution $p(z)$. In MIDCGAN, $\bar{z}$ comes from the encoder and hence is learned. These modifications and the fact that the discriminator is trying to maximize an objective results in an adversarial loss component of MIDCGAN as given by Eq. 3 representing the encoder as $E(x)$. In Eq. 3, $p(x)$ represents the actual data distribution of the dataset input to the encoder while $p(\tilde{x})$ represents the target canonical mental image distribution.

$$L_{adv}(x, \tilde{x}) = \mathop{\mathbb{E}}_{\tilde{x} \in p(\tilde{x})} [log(D(\tilde{x}))] + \mathop{\mathbb{E}}_{x \in p(x)} [log(1 - D(G(E(x))))] \tag{3}$$

In practice (Goodfellow et al., 2014a)(Pathak et al., 2016), this loss is implemented by training the discriminator and the encoder-generator pair using alternating SGD. The total joint loss used to train the encoder-decoder generator pair is the weighted sum of the Eq. 1 and Eq. 3 and is given as Eq. 4.

$$L_{total} = \lambda_{l_2} L_{l_2} + \lambda_{adv} L_{adv} \tag{4}$$

We experimented with various λ weights and observed that it is dataset dependent. We found the set $(0.2, 0.8)$ worked well for MNIST while $(1.0, 1.0)$ worked well for the remaining experiments.

The training of MIDCGAN involves three updates to the parameters of the three networks as shown in Algorithm 1. The convergence of MIDCGAN happens often early due to the dual mode selection because of the mental image and the adversarial loss and hence the discriminator could tell a real target sample quickly and the real component of the adversarial loss does not change significantly after a certain epoch as the generator tries to balance with the fake component of the adversarial loss.

Algorithm 1 Training MIDCGAN

$\theta^E, \theta^G, \theta^D \leftarrow HeNormal$ He et al. (2016) ▷ initialize encoder, decoder/generator and discriminator params
repeat
  $X, Y \leftarrow$ shuffled mini-batch ▷ X is the input image while Y is the mental image
  $Z \leftarrow Encoder(X)$
  $\tilde{X} \leftarrow Decoder(Z)$
  $L_{adv} \leftarrow log(D(\tilde{X})) + log(1 - D(\hat{\tilde{X}}))$ ▷ Compute adversarial loss for real and fake
  $\theta^D \leftarrow \theta^D - \nabla_{\theta^D}(L_{adv})$ ▷ Update Discriminator params with the adversarial loss gradients
  $l_2 \leftarrow ||\tilde{X} - \hat{\tilde{X}}||_2^2$
  $L_T \leftarrow \lambda_{l_2} l_2 + \lambda_{adv} L_{adv}$ ▷ total loss as fraction of adversarial and l_2 losses
  $\theta^E \leftarrow \theta^E - \nabla_{\theta^E}(L_T)$ ▷ Update Encoder params with the total loss gradients
  $\theta^G \leftarrow \theta^G - \nabla_{\theta^G}(L_T)$ ▷ Update Decoder/Generator params with total loss gradients
until number of epochs

3.3 Bottleneck Features for Classification

In our observations, the bottleneck features z represent important aspects of the input distribution, $p(x)$. We demonstrate how these features can be used for later classification. Once MIDCGAN is trained, we need only the encoder to generate the bottleneck features $\bar{z}$. To classify, we extract features on training/testing samples from the dataset distribution, $p(x)$. These features are then given to an l_2 SVM to perform classification.

4 Results and Discussion

In this section, we will present and discuss the four major experiments. In the first two, MNIST and SVHN are evaluated using stencils targets as mental images. In the final two, we evaluate object instance recognition using the Big Berkeley Instance Recognition Database (BigBIRD) and the University of Washington Kinects Objects Dataset. In the first experiment, we learn mental images for each of the objects. In the second experiment, we use the features learned previously to recognize an entirely new database. In all experiments, we show that learning in this manner outperforms features learned from a typical DCGAN architecture.

4.1 MNIST with Stencils as 'Mental Images'

MIDCGAN maps every kind of handwritten digit in the MNIST dataset to stenciled digits as mental images since they represent typical iconic representation of each digit. In this experiment, the encoder, decoder and the separate discriminator were all ResNet architecture with each convolution block replaced by 5 dual-convolutional residual units. Therefore, the separate discriminator is much deeper and more expressive than the more commonly used AlexNet discriminator. In Table 1, we compare different architectures trained with a bottleneck size of 256 against the state-of-the-art (Springenberg, 2015)(Makhzani et al., 2015)(Salimans et al., 2016) with a different number of labeled training samples. MIDCGAN with simple feedforward blocks and separate discriminator (meaning the discriminator weights are not shared with the encoder) outperforms others with fewer training samples and comparable performance with more training samples. A discriminator with

Table 1: MNIST test error (%) with n labeled examples for different architectures at nBn=256. Note: SS is simple feedforward CNN encoder/decoder/discriminator and separate discriminator and RS is residual CNN encoder/decoder/discriminator and separate discriminator

Method/ Arch	**Examples per Class**					**All**
	10	**20**	**50**	**100**	**200**	
DCGAN Alexnet	53.39 ± 4.22	47.1 ± 3.63	34.44 ± 2.13	24.55 ± 1.56	19.54 ± 1.1	8.17
Springenberg (2015)	-	-	-	1.39 ± 0.28	-	**0.48**
Makhzani et al. (2015)	-	-	1.90 ± 0.1	-	-	0.85
Salimans et al. (2016)	-	16.77 ± 4.52	2.21 ± 1.36	**0.93** ± **0.07**	**0.9** ± **0.04**	-
MIDCGAN SS	**1.51** ± **0.27**	**1.22** ± **0.08**	**1.13** ± **0.09**	1.07 ± 0.08	0.99 ± 0.11	0.82
MIDCGAN RS	4.58 ± 2.65	1.72 ± 0.69	1.48 ± 0.43	1.17 ± 0.13	1.03 ± 0.07	0.68

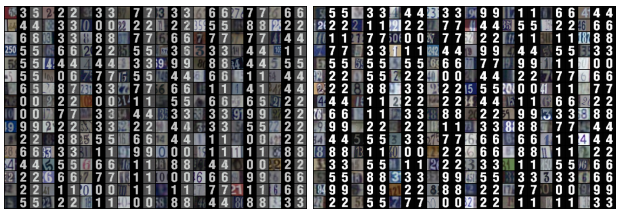

Figure 3: Reconstruction of stencil digits from SVHN samples at epochs 3(left) and 2499 (right). Sample, target and generated, consecutively.

shared weights with the encoder performs slightly worse allowing a possibility of almost half parameter reduction with minimal performance impact. We finally evaluate results with all examples labeled, and obtained our best results with a bottleneck size of 1024 with a separate discriminator of **0.53%** error. This is quite an improvement from 0.68% error reported in Table 1 with bottleneck size of 256.

4.2 SVHN WITH STENCILS AS 'MENTAL IMAGE'

MIDCGAN with simple feedforward blocks and AlexNet style discriminator outperforms all other architectures we experimented with for the SVHN dataset. As shown in Table 2, MIDCGAN produced the best performance among all the methods with a wide range of number of training samples.

Table 2: SVHN test error (%) with n labeled examples for different architectures at nBn=2048.

Method/ Arch	**Examples per Class**							
	10	**20**	**50**	**100**	**200**	**500**	**1000**	**2000**
Makhzani et al. (2015)	-	-	-	-	-	-	17.70 ± 0.3	-
Salimans et al. (2016)	-	-	-	-	-	**18.44** ± **4.8**	8.11 ± 1.3	**6.16** ± **0.58**
MIDCGAN SA	**17.8** ± **4.11**	**13.8** ± **1.8**	**9.1** ± **0.6**	**8.4** ± **0.68**	**7.89** ± **0.39**	-	**7.15** ± **0.26**	-

Table 3: Object recognition accuracy using different databases and approaches.

Dataset	Method/ Arch	Pre-training	Examples per Class				
			50	25	10	5	1
BigBIRD	MIDCGAN	Self-Supervised	**98.19 ± 0.14**	**95.45 ± 0.39**	**87.4 ± 0.59**	**79.8 ± 0.89**	**52.24 ± 0.64**
BigBIRD	DCGAN	Unsupervised	65.55 ± 0.42	54.7 ± 0.45	43.4 ± 0.54	32.6 ± 0.73	14.1 ± 0.89
RGBD	MIDCGAN	Transfer	**61.7 ± 0.25**	**59.7 ± 0.61**	**55.99 ± 0.54**	**51.4 ± 0.36**	37.6 ± 0.8
RGBD	DCGAN	Transfer	43.33 ± 0.5	40.34 ± 0.51	37.5 ± 0.46	32.1 ± 0.35	20 ± 0.8
RGBD	CNN Held et al. (2016)	Supervised	-	-	-	-	**63.9**

4.3 Object Instance Recognition

In object instance recognition, rather than identifying a general class of objects (eg., a bottle), we instead identify the specific instance of the object (e.g., a coke bottle). This problem is complicated by subtle differences between different instances, which are also sometimes very dependent on pose of the object. We evaluate results on two different datasets. The first is the Big Berkley Instance Recognition Dataset (BigBIRD), which has 125 instances of different objects (Singh et al., 2014). The objects include typical households items such as boxes of food, water bottles, shampoo, etc. Many instances are highly similar in appearance. For example, there are 8 different types of cereal bars made by the same company, some of which are extremely similar in packaging and appearance (i.e., "big chewy chocolate chip" vs. "big chewy peanut butter"). The dataset shows objects from 5 different viewing angles and 360 degrees of rotation sampled at approximately every 3 degrees. With MIDCGAN, recognition works as follows. For the sake of consistency, the mental image is the first level of pose, first instance of the object that is seen. MIDCGAN is able to see the object from a number of different poses, with each being mapped back to the mental image. When presented with a test image, MIDCGAN generates a mental image of the frontal view of the object.

From the perspective of the linear SVM, The BigBIRD dataset is a 125 way classification problem, where each class is a different object instance. We split the data with a stratified shuffle of all of the data into a training set with 60,000 images, a test set with 7,500 images and a validation set with 7,500 images. A 125-way linear SVM is trained for object instance recognition using bottleneck features. The results of this experiment are in the top of Table 3. In this case, we fix the bottleneck size at 4096 and use the separate discriminator. The loss function assigns a weight of 0.2 to l_2 loss and 0.8 to adversarial loss to capture the general details of the object without focusing on the specifics of how the objects appear. It's clear that the MIDCGAN features perform especially well at object instance recognition. With only a single instance of each class (i.e., *single shot recognition*), MIDCGAN recognizes objects 52.2% of the time, compared to 14% of the time for DCGAN. More interestingly, in some cases only the back or the top of a box is used to train the SVM – it is still able to recognize objects from dramatically different viewpoints!

A related question is how well do these features generalize to objects that have never been seen before. That is, what if we were unable to train the MIDCGAN using the objects in question, but instead only have a general knowledge of how mental images with objects of different classes. In this rather challenging problem (sometimes referred to as transfer learning), as we have not seen the objects a priori, but instead use the features learned on a different dataset. To evaluate in this case, we make use of a second dataset, the University of Washington Kinect Object Dataset (hence force referred to as RGBD) (Lai et al., 2011) (this dataset does have a depth component, but we only make use of the color components). RGBD has 300 different object instances grouped into 51 classes. The dataset was collected at 3 different levels of pitch, and a full 360 degree pan, sampled to generate 250 different poses per level of pan. It's interesting to note that these objects are also household type of objects, but this dataset contains a lot of images that are nowhere near the type of images from BigBIRD. For example, this has images that are not present in BigBIRD such as

heads of garlic, bok choy, apples, etc. The transfer results are shown in the bottom of Table 3. Here we follow the same general criteria established in (Held et al., 2016), which used one level of pitch ($30°$) to train, and another ($45°$) to evaluate. Here, we see that the performance for single shot recognition is not as good as it was in the previous experiment (37.9%), and with 10 labeled images we recognize objects correctly 52% of the time). The difference in performance between MIDCGAN and DCGAN suggests that MIDCGAN learns features that are beneficial for learning different classes of objects.

5 CONCLUSION

We have demonstrated the use of mental model autoencoder-based DCGAN (MIDCGAN) on digit classification (MNIST), and object recognition on both the Kinects Objects Dataset (RGB-D) and the Bigbird Dataset. Although MIDCGAN was demonstrated on representative viewpoints, the selection of the mental target image could be arbitrary. In cases when MIDCGAN was not sure about the object, the generated image did not resemble the actual representative image, or important details about the representative image were omitted. In essence, it could potentially provide another way to determine the confidence in the prediction of the object class.

MIDCGAN can be used on robotics platform with real images in the wild. Given the growing interest in active object manipulation and recognition in the robotics community (Browatzki et al., 2012). Incorporating real manipulation of objects as separate modality with MIDCGAN could permit robots to learn about objects in their environment with minimal supervision. The mental images described in this paper map very well to the concept of prototype learning in the cognitive science community. Several improvements could be made to MIDCGAN in the future. First, we have generally assumed the availability of a tutor to select a mental image. Autonomous selection of the mental images is an inherent extension to allow semi-supervised and fully unsupervised MIDCGAN training. Alternatively, prototype images can be selected in some systematic manner using some heuristics related to the objects (i.e., a cup holds liquid so prototype is face up, the handle is to be grabbed so that should be visible).

REFERENCES

H. Larochelle O. Winther A. Larsen, S. Sonderby. Autoencoding beyind pixels using a learned similarity metric, 2016. URL `https://arxiv.org/pdf/1512.09300v2.pdf`.

Andrew Brock, Theodore Lim, JM Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

Björn Browatzki, Vadim Tikhanoff, Giorgio Metta, Heinrich H Bülthoff, and Christian Wallraven. Active object recognition on a humanoid robot. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 2021–2028. IEEE, 2012.

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*, pp. 2172–2180, 2016.

Antonia Creswell and Anil Anthony Bharath. Inverting the generator of a generative adversarial network. *arXiv preprint arXiv:1611.05644*, 2016.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Ruohan Gao, Dinesh Jayaraman, and Kristen Grauman. Object-centric representation learning from unlabeled videos. *Asian Conference on Computer Vision (ACCV)*, 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, 2014a.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645. Springer, 2016.

David Held, Silvio Savarese, and Sebastian Thrun. Deep lerning for single-view instance recognition. *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neeural Information Processing Systems (NIPS)*, pp. 2017–2025, 2015.

Karin H James, Susan S Jones, Linda B Smith, and Shelley N Swain. Young children's self-generated object views and object recognition. *Journal of Cognition and Development*, 15(3): 393–401, 2014.

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1817–1824. IEEE, 2011.

Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pp. II–97. IEEE, 2004.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436, 2015.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544, 2016.

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

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. 2226–2234, 2016.

Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 509–516. IEEE, 2014.

Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2015.

Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision*, pp. 318–335. Springer, 2016.

Raymond Yeh, Chen Chen, Teck Yian Lim, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016.

Chen Yu, Linda B Smith, Hongwei Shen, Alfredo F Pereira, and Thomas Smith. Active information selection: Visual attention through the hands. *IEEE Transactions on Autonomous Mental Development*, 1(2):141–151, 2009.

Shuangfei Zhai, Yu Cheng, Rogerio Feris, and Zhongfei Zhang. Generative adversarial networks as variational training of energy based models. *arXiv preprint arXiv:1611.01799*, 2016.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pp. 597–613. Springer, 2016a.

Zhuotun Zhu, Xinggang Wang, Song Bai, Cong Yao, and Xiang Bai. Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, 204:41 – 50, 2016b.

A Supplemental Material

This section presents the details of the MIDCGAN architecture, training parameters and generated images results for the different architectures, training conditions and datasets.

A.1 Architectural Details

Both MIDCGAN and DCGAN used in this experiment had similar architectures except the targets are different in each case. The architecture is composed of 3 networks, i.e. the encoder, the decoder and the discriminator. The encoder and decoder together form the autoencoder embedded in the DCGAN that serves as the generator. All the three networks (with the exception of the AlexNet discriminator) are built around 3 major convolutional blocks. When these 3 major convolutional blocks are single convolution followed by batch normalization and activation each, the resulting architecture is referred to as simple. This architecture is equivalent to most DCGANs that are based on the AlexNet architecture for each network in the GAN. We then replaced these convolution blocks with 3 residual units each and we referred to the resulting architecture as resnet. Each residual unit in turn is double convolutional blocks proceeded by batch normalization and activation and are connected by a shortcut (residual) connection.

The encoder has single identity convolutions at the beginning and end for just channel transformation. The network also has a dense layer at the end that converts the spatially flattened pooled features into a bottleneck of features suitable for classification. The bottleneck features were parameterized by the bottleneck size of nBn. As presented in the results section, we have experimented with various bottleneck sizes such as 256, 512, 1024 and 2048.

The decoder has two dense layers that transform the input bottleneck feature size of nBn to N that is suitable to transform back to spatial planes that were at the end of the encoder before feature transformation using the reshape layer. The 3 major convolutional blocks in the decoder are implemented together with upsampling instead of converting them to transposed convolution or deconvolution because we wanted to keep the 3 main convolution block the same for all the 3 networks for consistency. The decoder also has a single identity convolution at the output for channel transformation.

The discriminator is a bit different as there were three kinds of discriminators that were explored in this paper. The separate discriminator was essentially the same as the encoder except it adds binary discrimination layer after the feature transformation. The shared discriminator went further and shares weights with the discriminator with the exception of the last discrimination layer. The third discriminator was the AlexNet discriminator that was similar to most discriminator architectures in GAN literature. We added this discriminator for fair comparison with literature.

All the convolutions in all the architectures were 4 x 4 convolutions. Downsampling is implemented by a stride of size 2 at the beginning convolution of each of the 3 major convolutional blocks in the encoder and the discriminator. Beginning and ending convolutions in each network were using $tanh$ activations while internal convolutions used $LeakyReLU$ with parameter of 0.2. The number

of filters in the internal convolutional blocks were all 32 for all the three networks for MNIST training while there were 64 filters for BigBird training.

A.2 TRAINING PARAMETERS

The MNIST dataset were trained on the 60, 000 training images while the BigBird dataset was divided into train and validation using 80% - 20% split. The 20% split was further divided equally into validation and testing sets. We employed the Adam optimizer for both discriminator and generator training. For MNIST the learning rate for both discriminator and generator were 0.002 with the discriminator beta1 set at 0.5 and generator beta1 set at 0.9. For BigBird training, we used learning rate of 0.0001 and beta1 parameter of 0.9 for both discriminator and generator. The maximum epochs for each training were 2500 with best model saved using the total validation generator loss. However, in practice, we saw that most of the networks converged much earlier usually after few hundred epochs.

A.3 NOTES ON IMPROVEMENTS FOR CONVERGENCE

Compared to Goodfellow et al.(Salimans et al., 2016), we have not applied advanced techniques such as feature matching, minibatch discrimination, historical averaging and virtual batch normalization to improve convergence and performance. We rather focused on basic techniques to stabilize DCGAN and ResNet training. Training a DCGAN involves ideally finding the Nash equilibrium point of the two models (discriminator and generator networks) that are competitive in adversarial fashion. In practice, it is difficult to find the Nash equilibrium point for adversarial training where the loss function is non-convex (Salimans et al., 2016). Therefore, several heuristic techniques are employed to make the training of DCGANs converge using gradient descent back propagation.

We would like to emphasize that, in this work, we opt to not include the more complicated heuristics suggested by (Salimans et al., 2016) to assess the performance of our networks in a fair manner so as to not add confounding factors to the training. We believe that if we add all the improvements suggested, it would be difficult to isolate the contribution of the architectural innovations and the main idea of using mental imagery for the generative reasoning. It would be hard to tell if the performance increase are from the network and mental image canonical learning or the improvements. Moreover, it would also take away from the focus of this paper, which introduces learning from mental images. We utilized mainly standard GAN training techniques such as one-sided label smoothing (Salimans et al., 2016), regular batch normalization after each convolutional block in the case of the simple architecture or after each residual block in the case of ResNet version of the networks. We also employed l2 regularization for the ResNet units inside the ResNet blocks to regularize the redundant aggressive learning as the network gets deeper and deeper. We have not employed this regularization to the shallower simple networks.

A.4 MNIST ADDITIONAL RESULTS

Generated samples from MNIST dataset are shown in Fig. 4 at epoch 7 (left) and epoch 1500 (right) with a bottleneck of size 256. It is evident that at an earlier epoch, the generator confuses several digits and outputs more than one mode in one generated image (the first digit being 7 overlaid on top of 4. However, with more training epochs the generator learns to separate out only the target mode (right).

Both the bottleneck size and the type of discriminator architecture have a direct impact on the descriptiveness of the bottleneck features, as shown in Table 4. To explore this relationship, we chose 200 training images as this permits a fair comparison against (Salimans et al., 2016). The results for this experiment are in Table 4. Our best result came with a bottleneck size of 1024 and a more complicated ResNet discriminator. Note that in general, more complicated (and descriptive) discriminators produce better results with larger bottleneck sizes whereas simpler discriminators produce better results with smaller bottleneck sizes. We hypothesize that this is because simpler discriminators produce simpler features that are easier to capture with fewer neurons.

In Table 5, we compare performance with different training sample sizes and different types of discriminators. Goodfellow et al. (Salimans et al., 2016) showed classification results with as few as 20 training samples; we show results down to 10 samples per class (i.e., one sample per class). The

Figure 4: Reconstruction of stencil digits from MNIST samples at epochs 7(left) and 1500 (right). Sample, target and generated, consecutively.

Table 4: Effect of bottleneck size on performance and comparison with the state-of-the-art at n=200.

Arch	**nBn**				
	256	**512**	**1024**	**2048**	**4096**
Separate	**1.025**	1.126	**0.844**	**0.907**	**0.927**
Alexnet	1.049	**0.951**	1.387	1.302	1.155
Salimans et al. (2016)	x	x	0.9	x	x

results indicated that the more descriptive discriminator (separate) dominated the simpler (AlexNet). Although distinctive labels were not provided to MIDCGAN, clearly the features learned through mental image generation are well suited for this task.

A.5 SVHN Additional Results

Table 6 shows the effect of bottleneck feature size on MIDCGAN performance using the SVHN dataset.

Table 5: Test error (%) on MNIST at bottleneck size of 1024 for the two dicriminator types and state-of-the-art.

Arch	**n**			
	10	**20**	**50**	**100**
Separate	**2.832**	**1.209**	**1.103**	1.017
Alexnet	8.045	4.473	2.426	1.727
Salimans et al. Salimans et al. (2016)	x	16.77	2.21	**0.93**

Table 6: Effect of bottleneck size on performance and comparison with the state-of-the-art at n=1000.

Method/ Arch	nBn			
	256	**512**	**1024**	**2048**
Salisman et. al.	-	-	**8.11**	-
MIDCGAN SS	17.59	23.49	-	-
MIDCGAN SA	**11.5**	**10.46**	10.82	**7.15**

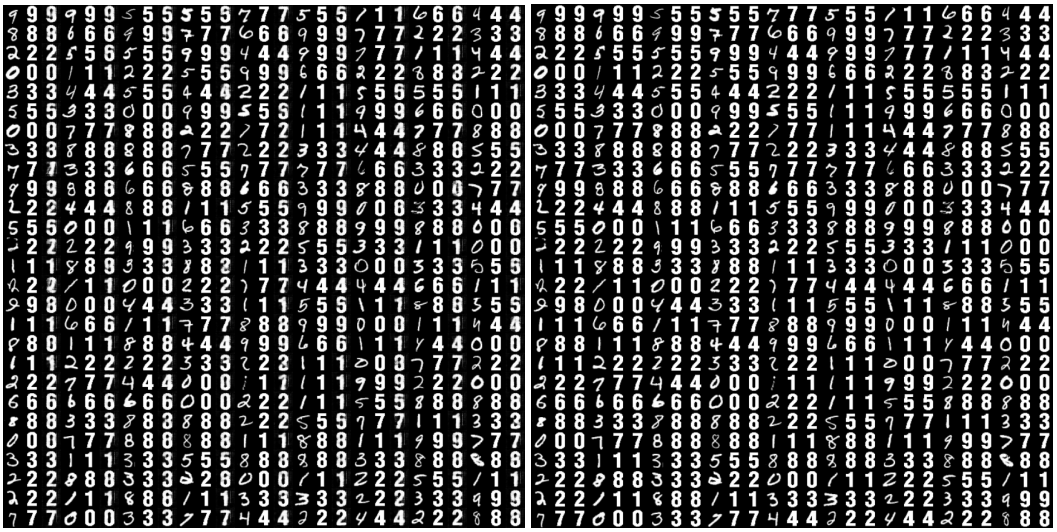

Figure 5: MIDCGAN generated images (every third column) for test set of MNIST after the first epoch (left) and the last epoch (right). Every first column is the input, every second column is the target and every third column is the generated image.

A.6 Additional Generated Images

In this supplementary material, we added additional generated images for qualitative comparisons of MIDCGAN with the regular DCGAN counterpart. We are adding validation set reconstruction images early on the training and after the training had finished to show the progression of quality of the reconstructed images.

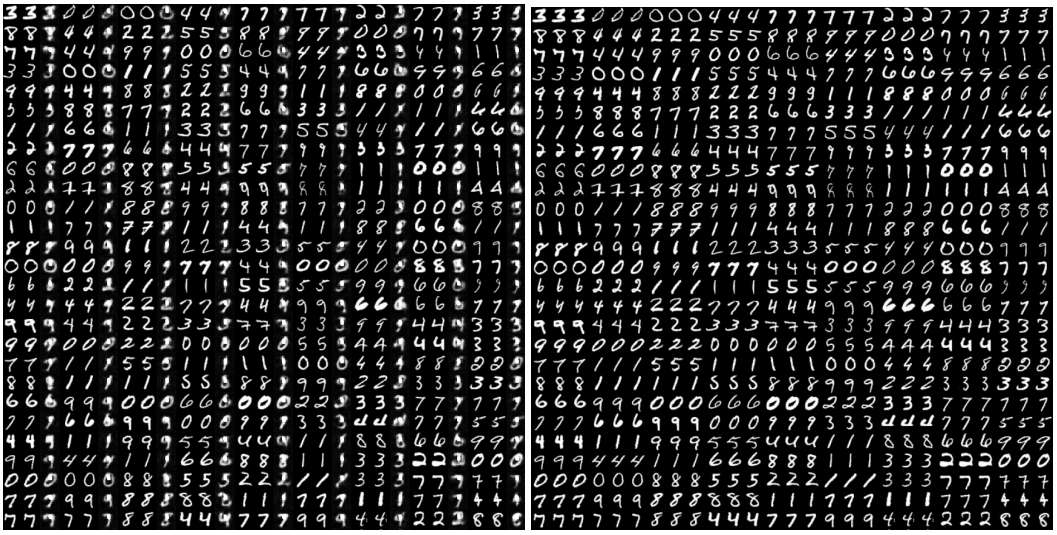

Figure 6: DCGAN generated images (every third column) for test set of MNIST after the first epoch (left) and the last epoch (right). Every first column is the input, every second column is the target and every third column is the generated image.

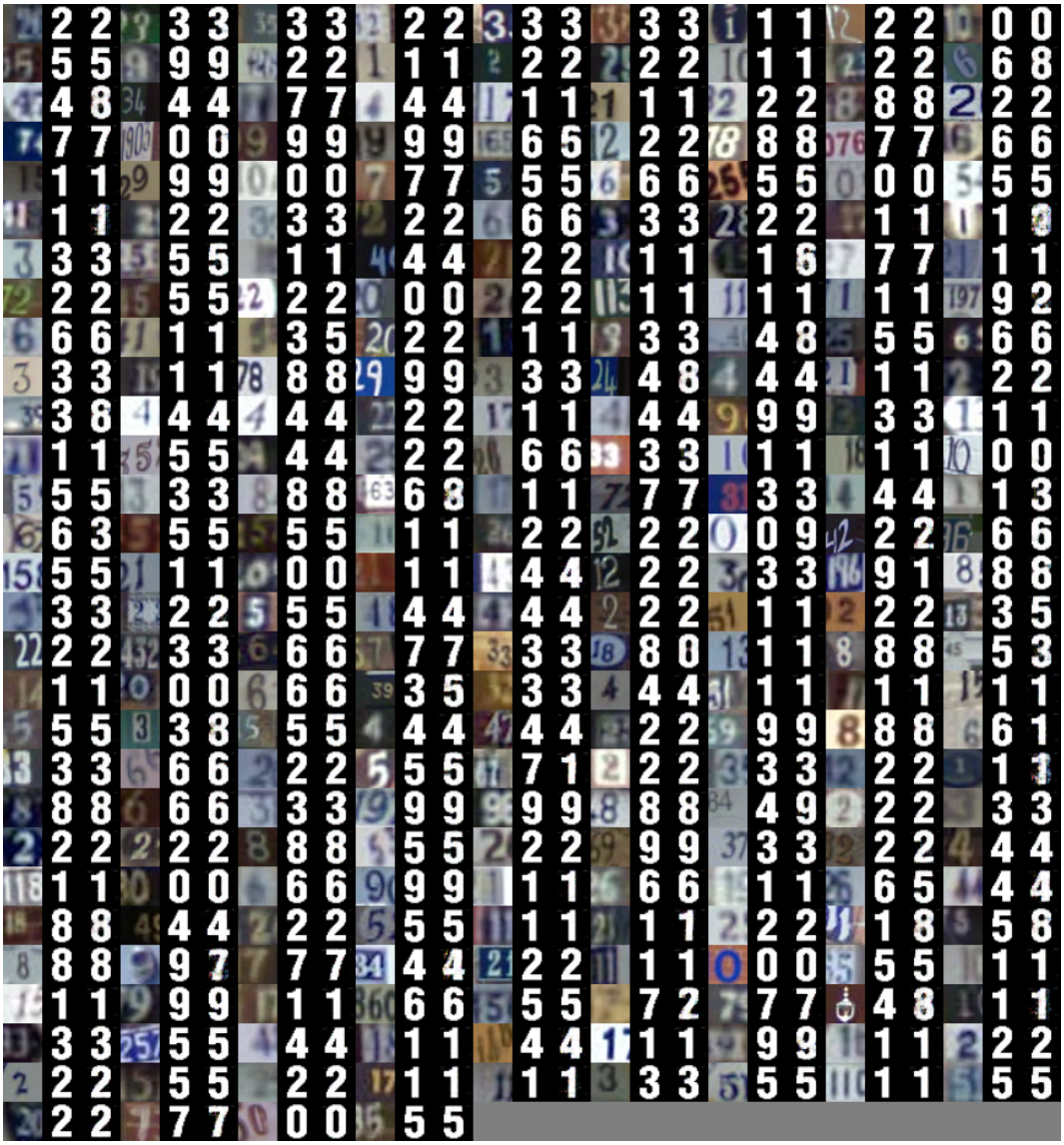

Figure 7: MIDCGAN generated images (every third column) for test set of SVHN. Every first column is the input, every second column is the target and every third column is the generated image.

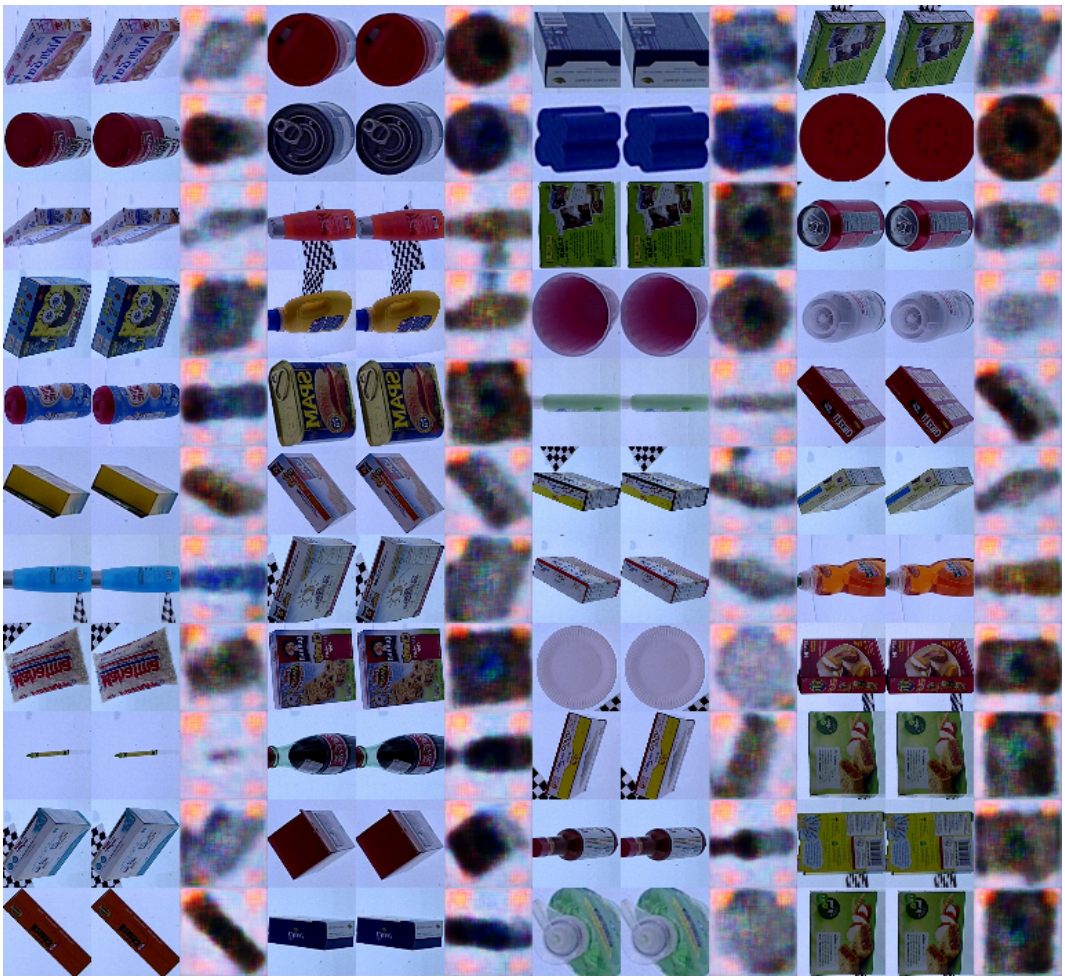

Figure 8: DCGAN generated images (every third column) for validation set of BigBird after the first epoch (last epoch is shown in the next figure). Every first column is the input, every second column is the target and every third column is the generated image.

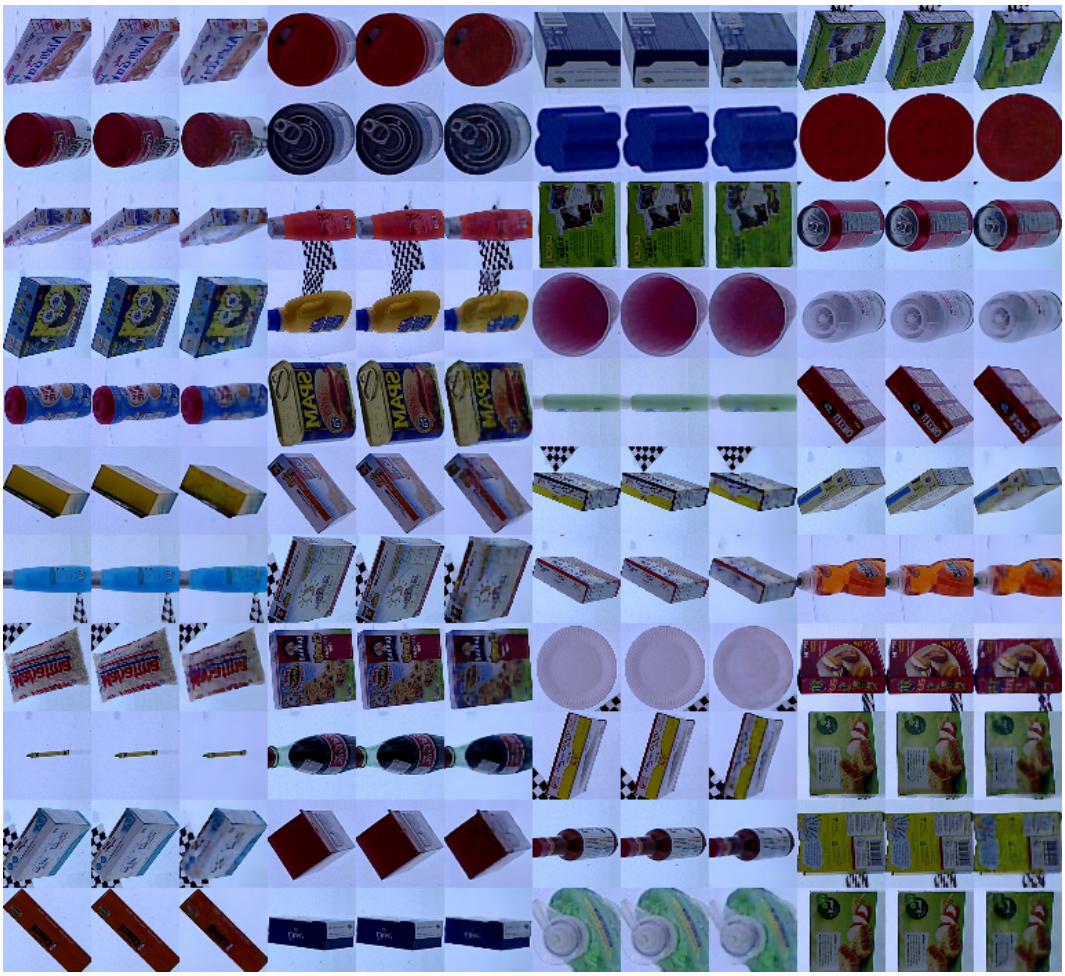

Figure 9: DCGAN generated images (every third column) for validation set of BigBird after the last epoch (first epoch is shown in the previous figure). Every first column is the input, every second column is the target and every third column is the generated image.

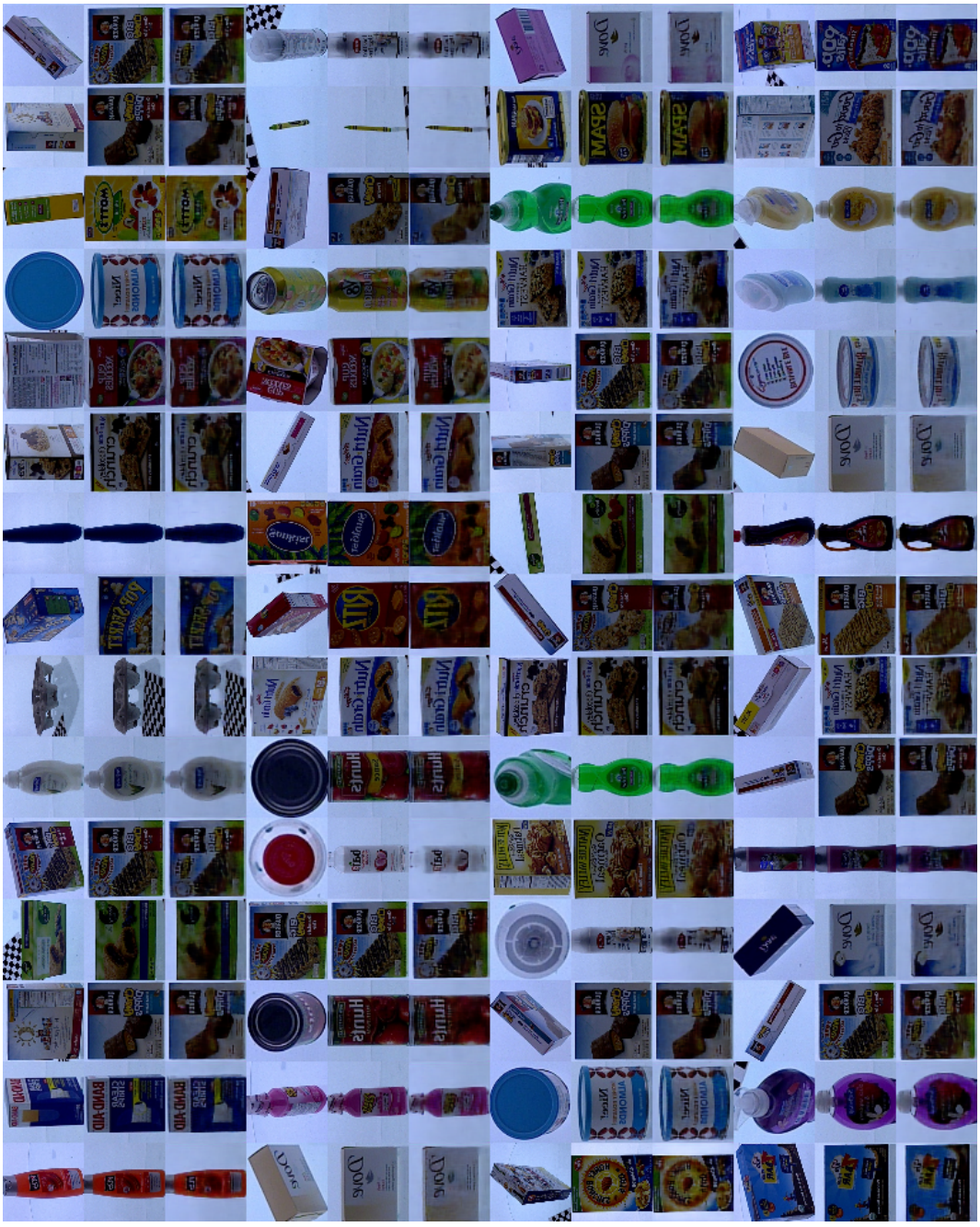

Figure 10: MIDCGAN generated images (every third column) for test set of BigBird after the last epoch. Every first column is the input, every second column is the target and every third column is the generated image.