MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Coupled Generative Adversarial Nets

Liu, M.-Y.; Tuzel, O.

TR2016-070 June 2016

Abstract

We propose the coupled generative adversarial nets (CoGAN) framework for generating pairs of corresponding images in two different domains. The framework consists of a pair of generative adversarial nets, each responsible for generating images in one domain. We show that by enforcing a simple weight-sharing constraint, the CoGAN learns to generate pairs of corresponding images without existence of any pairs of corresponding images in the two domains in the training set. In other words, the CoGAN learns a joint distribution of images in the two domains from images drawn separately from the marginal distributions of the individual domains. This is in contrast to the existing multi-modal generative models, which require corresponding images for training. We apply the CoGAN to several pair image generation tasks. For each task, the GoGAN learns to generate convincing pairs of corresponding images. We further demonstrate the applications of the CoGAN framework for the domain adaptation and cross-domain image generation tasks.

arXiv

Coupled Generative Adversarial Nets

Ming-Yu Liu
Mitsubishi Electric Research Labs (MERL),
Cambridge, MA 02139
mliu@merl.com

Oncel Tuzel
Mitsubishi Electric Research Labs (MERL),
Cambridge, MA 02139
oncel@merl.com

Abstract

We propose the coupled generative adversarial nets (CoGAN) framework for generating pairs of corresponding images in two different domains. The framework consists of a pair of generative adversarial nets, each responsible for generating images in one domain. We show that by enforcing a simple weight-sharing constraint, the CoGAN learns to generate pairs of corresponding images without existence of any pairs of corresponding images in the two domains in the training set. In other words, the CoGAN learns a joint distribution of images in the two domains from images drawn separately from the marginal distributions of the individual domains. This is in contrast to the existing multi-modal generative models, which require corresponding images for training. We apply the CoGAN to several pair image generation tasks. For each task, the GoGAN learns to generate convincing pairs of corresponding images. We further demonstrate the applications of the CoGAN framework for the domain adaptation and cross-domain image generation tasks.

1 Introduction

The pair image generation problem concerns learning a generative model that can generate pairs of corresponding images in two different domains such as the same character in different fonts, the same face with different attributes, or the same scene in different modalities. In addition to movie and game production, pair image generation finds applications in image transformation and domain adaptation. When training samples are given as pairs of corresponding images in the two domains (training samples are drawn from a joint distribution), several existing approaches [1, 2, 29, 30] can be applied. However, building a dataset with the correspondence images is often difficult. This is particularly true when the two domains are only related in an abstract sense. The dependency on correspondence greatly limits the applicability of the existing approaches.

To overcome the limitation, we propose the coupled generative adversarial nets (CoGAN) framework, which can learn a joint distribution of images for generating pairs of corresponding images in two domains unsupervisedly. Existence of corresponding images in the two domains in the training set is not required. It can learn the joint distribution from images drawn separately from the marginal distributions of the individual domains. The CoGAN framework is based on the generative adversarial nets (GAN) framework [3], which has been established as a viable solution for novel image generation. The CoGAN framework extends the GAN framework to the unsupervised pair image generation task.

The CoGAN consists of a pair of GANs. Each generates images in one domain. When trained separately, they generate unrelated images. We show that by enforcing a simple weight-sharing constraint, the CoGAN learns to capture the correspondence in the two domains in an unsupervised fashion. The CoGAN framework is inspired by the idea that deep neural networks learn a hierarchical feature representation. By enforcing the layers that decode high-level concepts in the two GANs to share the weights, it encourages the two GANs to decode the high-level semantics in the same way. The layers that decode low-level details then map the shared representation to images in different domains for fooling the respective discriminative models.

We apply the CoGAN framework to several pair image generation tasks. Through convincing visualization results and quantitative evaluations, we verify the effectiveness of the CoGAN framework. We also show its applications for the unsupervised domain adaptation and image transformation tasks.

2 Generative Adversarial Nets

The GAN framework consists of a generative model and a discriminative model. The objective of the generative model is to synthesize images resembling real images, while the objective of the discriminative model is to distinguish real images from synthesized ones. Both the generative and discriminative models are realized as multilayer perceptrons.

Let $\mathbf{x}$ be a natural image drawn from a distribution, p_X, and $\mathbf{z}$ be a random vector in $\mathbb{R}^d$. Note that we only consider that $\mathbf{z}$ is from a uniform distribution with a support of $[-1\ \ 1]$ for each dimension, but different distributions such as a multivariate normal distribution can be used as well. Let g and f be the generative and discriminative models, respectively. The generative model takes $\mathbf{z}$ as input and outputs an image, $g(\mathbf{z})$, that has the same support as $\mathbf{x}$. Denote the distribution of $g(\mathbf{z})$ as p_G. The discriminative model estimates the probability that an input image is drawn from p_X. Ideally, $f(\mathbf{x}) = 1$ if $\mathbf{x} \sim p_X$ and $f(\mathbf{x}) = 0$ if $\mathbf{x} \sim p_G$. The GAN framework corresponds to a minimax two-player game, and the generative and discriminative models can be trained jointly via solving

$$\max_g \min_f V(f, g) \equiv E_{\mathbf{x} \sim p_{\mathbf{X}}}[-\log f(\mathbf{x})] + E_{\mathbf{z} \sim p_{\mathbf{Z}}}[-\log(1 - f(g(\mathbf{z})))]. \tag{1}$$

In practice (1) is solved by alternating the following two gradient update steps:

$$\text{Step 1: } \boldsymbol{\theta}_f^{t+1} = \boldsymbol{\theta}_f^t - \lambda^t \nabla_{\boldsymbol{\theta}_f} V(f^t, g^t), \qquad \text{Step 2: } \boldsymbol{\theta}_g^{t+1} = \boldsymbol{\theta}_g^t + \lambda^t \nabla_{\boldsymbol{\theta}_g} V(f^{t+1}, g^t)$$

where $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_g$ are the parameters of f and g, λ is the learning rate, and t is the iteration number.

Goodfellow et al. [3] show that, given enough capacity to f and g and sufficient training iterations, the distribution, p_G, converges to p_X. In other words, from a random vector, $\mathbf{z}$, the network g can synthesize an image, $g(\mathbf{z})$, that resembles that drawn from the true distribution, p_X.

3 Coupled Generative Adversarial Nets

The CoGAN framework as illustrated in Figure 1 is designed for synthesizing pairs of corresponding images in two different domains. It consists of a pair of GANs—GAN1 and GAN2; each is responsible for synthesizing images in one domain. During training, we force them to share a subset of network parameters. This weight-sharing constraint results in that the two GANs learn to synthesize pairs of corresponding images in an unsupervised fashion. We note that the framework can be easily extended to generating corresponding images in multiple domains by adding more GANs.

Generative Models: Let $\mathbf{x}_1$ be an image drawn from the distribution of the 1st domain, $\mathbf{x}_1 \sim p_{X_1}$. Let $\mathbf{x}_2$ be an image drawn from the distribution of the 2nd domain, $\mathbf{x}_2 \sim p_{X_2}$. Let g_1 and g_2 be the generative models of GAN1 and GAN2, which individually map a random vector input $\mathbf{z}$ to images that have the same support as $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. Denote the distributions of $g_1(\mathbf{z})$ and $g_1(\mathbf{z})$ by p_{G_1} and p_{G_2}. Similar to the GAN framework, both g_1 and g_2 are realized as multilayer perceptrons:

$$g_1(\mathbf{z}) = g_1^{(m_1)}\big(g_1^{(m_1-1)}\big(\ldots\ g_1^{(2)}\big(g_1^{(1)}(\mathbf{z})\big)\big)\big), \quad g_2(\mathbf{z}) = g_2^{(m_2)}\big(g_2^{(m_2-1)}\big(\ldots\ g_2^{(2)}\big(g_2^{(1)}(\mathbf{z})\big)\big)\big)$$

where $g_1^{(i)}$ and $g_2^{(i)}$ are the ith layers of g_1 and g_2 and m_1 and m_2 are the numbers of layers in g_1 and g_2. Note that m_1 need not equal m_2. Also note that the support of $\mathbf{x}_1$ need not equal to that of $\mathbf{x}_2$.

Through layers of perceptron operations, the generative models gradually decode information from more abstract concepts to more material details. The bottom layers decode high-level semantics and the top layers decode low-level details. Note that this information flow is opposite to that in a discriminative deep neural network [4] where the bottom layers extract low-level features while the top layers extract high-level features.

Based on the idea that a pair of corresponding images in two domains share the same high-level semantics, we force the bottom layers of g_1 and g_2 to have identical structure and share the weights. That is $\boldsymbol{\theta}_{g_1^{(i)}} = \boldsymbol{\theta}_{g_2^{(i)}}$, for $i = 1, 2, ..., k$ where k is the number of shared layers, and $\boldsymbol{\theta}_{g_1^{(i)}}$ and $\boldsymbol{\theta}_{g_2^{(i)}}$

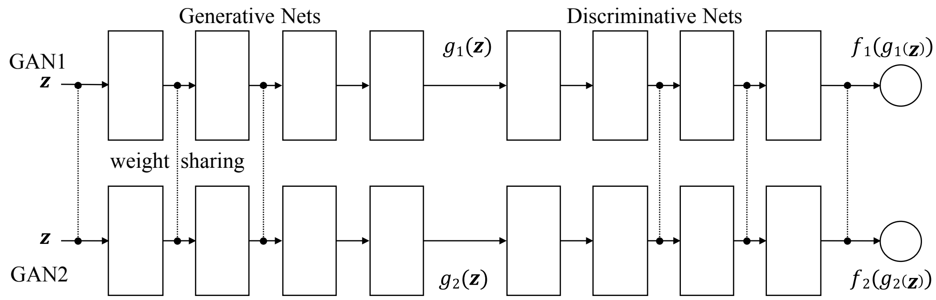

Figure 1: The proposed CoGAN framework consists of a pair of generative adversarial nets (GAN): GAN1 and GAN2. Each has a generative model that can synthesize realistic images in one domain and a discriminative model that can classify whether an input image is a real image or a synthesized image in the domain. We tie the weights of the first few layers (responsible for decoding high-level semantics) of the generative models, g_1 and g_2. We also tie the weights of the last few layers (responsible for encoding high-level semantics) of the discriminative models, f_1 and f_2. This weight-sharing constraint forces the GAN1 and GAN2 to learn to synthesize pairs of corresponding images in the two domains, where the correspondence is defined in the sense that the two images share the same high-level abstraction but have different low-level realizations.

are the parameters of $g_1^{(i)}$ and $g_2^{(i)}$, respectively. This weight-sharing constraint forces the high-level semantics are decoded in the same way in g_1 and g_2. No constraints are enforced on the top layers. They can materialize the shared high-level representation differently in each domain.

Discriminative Models: Let f_1 and f_2 be the discriminative models of GAN1 and GAN2 given by

$$f_1(\mathbf{x}_1) = f_1^{(n_1)}\big(f_1^{(n_1-1)}\big(\ldots\ f_1^{(2)}\big(f_1^{(1)}(\mathbf{x}_1)\big)\big)\big),\ f_2(\mathbf{x}_2) = f_2^{(n_2)}\big(f_2^{(n_2-1)}\big(\ldots\ f_2^{(2)}\big(f_2^{(1)}(\mathbf{x}_2)\big)\big)\big)$$

where $f_1^{(i)}$ and $f_2^{(i)}$ are the ith layers of f_1 and f_2 and n_1 and n_2 are the numbers of layers. The discriminative models map an input image to a probability score, estimating the likelihood that the input is drawn from a true data distribution. The bottom layers of the discriminative models extract low-level features, while the top layers extract high-level features. Because the input images are realizations of the same high-level semantics in two different domains, we force f_1 and f_2 to have the same top layers, which is achieved by sharing the weights of the top layers via $\boldsymbol{\theta}_{f_1^{(n_1-i)}} = \boldsymbol{\theta}_{f_2^{(n_2-i)}}$, for $i = 0, 1, ..., l-1$ where l is the number of weight-sharing layers in the discriminative models, and $\boldsymbol{\theta}_{f_1^{(i)}}$ and $\boldsymbol{\theta}_{f_2^{(i)}}$ are the network parameters of $f_1^{(i)}$ and $f_2^{(i)}$, respectively..

Learning: The CoGAN framework corresponds to a constrained minimax game given by

$$\max_{g_1,g_2}\min_{f_1,f_2} V(f_1,f_2,g_1,g_2),\ \text{subject to}\quad \begin{aligned} &\boldsymbol{\theta}_{g_1^{(i)}} = \boldsymbol{\theta}_{g_2^{(i)}}, && \text{for } i = 1,2,...,k \\ &\boldsymbol{\theta}_{f_1^{(n_1-i)}} = \boldsymbol{\theta}_{f_2^{(n_2-i)}}, && \text{for } i = 0,1,...,l-1 \end{aligned} \tag{2}$$

where the value function V is given by

$$\begin{aligned} V(f_1,f_2,g_1,g_2) = &\ E_{\mathbf{x}_1\sim p_{\mathbf{X}_1}}[-\log f_1(\mathbf{x}_1)] + E_{\mathbf{z}\sim p_{\mathbf{Z}}}[-\log(1-f_1(g_1(\mathbf{z})))] \\ &+ E_{\mathbf{x}_2\sim p_{\mathbf{X}_2}}[-\log f_2(\mathbf{x}_2)] + E_{\mathbf{z}\sim p_{\mathbf{Z}}}[-\log(1-f_2(g_2(\mathbf{z})))]. \end{aligned} \tag{3}$$

In the game, there are two teams and each team has two players. The generative models, g_1 and g_2, form a team and work together for synthesizing a pair of images in two different domains for confusing the discriminative models, f_1 and f_2. The discriminative models try to differentiate images drawn from the training data distribution in the respective domains from those drawn from the respective generative models. The collaboration is established from the weight-sharing constraint. Similar to the GAN framework, the CoGAN learning can be achieved by the back-propagation algorithm with the alternating gradient update scheme. The details of the learning algorithm are given in the supplementary materials.

Remarks: In the CoGAN learning, the training samples are from the marginal distributions, p_{X_1} and p_{X_2}. It does not rely on samples from the joint distribution, p_{X_1,X_2}, where corresponding images in the two domains would be available. Our main contribution is in showing that with just samples separately drawn from the marginal distributions, the CoGAN learns a joint distribution that can be used to generate pairs of corresponding images in the two domains. Both the weight-sharing

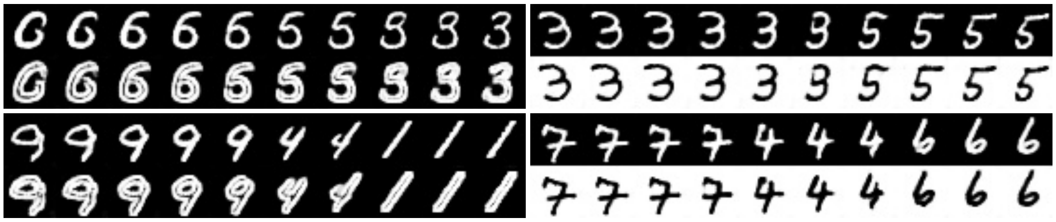

Figure 2: Left: pair generation of digit and corresponding edge images (Task $\mathbb{A}$). Right: pair generation of digit and corresponding negative images (Task $\mathbb{B}$). Each of the top and bottom pairs was generated using the same input noise. We visualized the results by traversing in the input space.

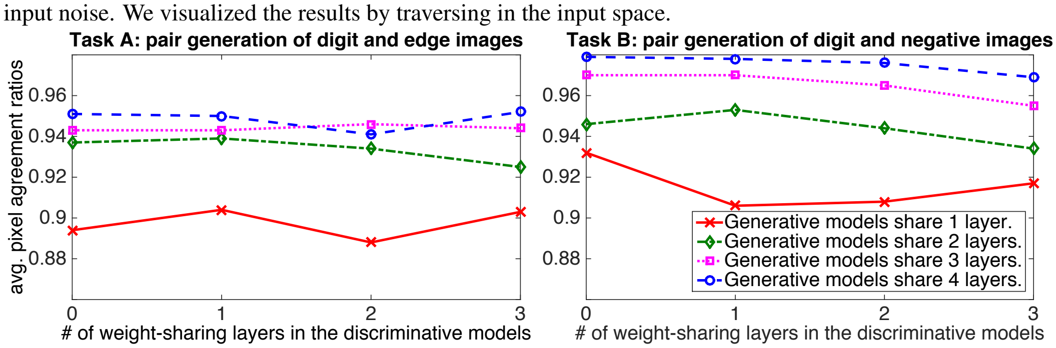

Figure 3: The figures plot the avergae pixel agreement ratios of the CoGANs with different weight-sharing configurations for Task $\mathbb{A}$ and $\mathbb{B}$. The larger the pixel agreement ratio the better the pair generation performance. We found that the performance was positively correlated with the number of weight-sharing layers in the generative models but was uncorrelated to the number of weight-sharing layers in the discriminative models.

constraint and adversarial training are crucial for enabling this capability. Unlike the autoencoder training [29], which encourages a generated pair of images to be *identical* to the target pair of corresponding images in the two domains for minimizing the reconstruction loss[1], the adversarial training only encourages the generated pair of images to be *individually resembling to* the images in the respective domains. With this more relaxed adversarial training objective, the weight-sharing constraint can then kick in for the unsupervised learning task. The generative models must use the capacity more efficiently for fooling the discriminative models, and the most efficient way of using the capacity for generating a pair of realistic images in two domains is to generate a pair of *corresponding* images since the neurons responsible for decoding high-level semantics can be shared.

4 Experiments

In the experiments, we emphasized that there were no correspondence annotations in the two domains in the training sets. The CoGAN learned to generate pairs of corresponding images in a purely unsupervised fashion. We were unaware of existing approaches with the same capability and hence did not compare the CoGAN with prior works. Instead, we compared it to a conditional generative model to demonstrate its advantage in learning the joint distribution. Recognizing that popular performance metrics for evaluating generative models all subject to issues [5], we adopted a pair image generation performance metric for comparison. Due to the page limit, many details including the network architectures and additional experiment results are given in the supplementary materials. We will make our implementation publicly available.

Digit Pair Generation: We used the MNIST training set to train CoGANs for the following two pair generation tasks: Task $\mathbb{A}$, generating a digit and its edge image, and Task $\mathbb{B}$, generating a digit and its negative image. In Task $\mathbb{A}$, the 1st domain consisted of the original handwritten digit images, while the 2nd domain consisted of their edge images. We used an edge detector to compute training edge images for the 2nd domain. In Task $\mathbb{B}$, the two domains were the digit images and their negatives.

The CoGANs were realized using deep convolutional networks. The two generative models had an identical structure; both had 5 layers and were fully convolutional. The stride lengths of the convolutional layers were fractional. The models also employed the batch normalization processing [6] and the parameterized rectified linear unit processing [7]. We shared the parameters for all the

[1]This is also why [29] requires samples from the joint distribution for learning the joint distribution.

layers except for the last convolutional layers that were responsible for generating outputs. For the discriminative models, we used a variant of the LeNet [8]. The inputs to the discriminative models were batches containing output images from the generative models and images from the two training subsets (each pixel value is linearly scaled to $[0\ 1]$).

We divided the training set into two equal-size *non-overlapping* subsets. One was used to train GAN1 and the other was used to train GAN2. Due to lack of corresponding images in the two subsets, the CoGAN learning was performed unsupervisedly. We used the ADAM algorithm [9] for training and set the learning rate to 0.0002, the 1st momentum parameter to 0.5, and the 2nd momentum parameter to 0.999 as suggested in [10]. The mini-batch size was 128. We trained the CoGAN for 25000 iterations. These hyperparameters were fixed for all the visualization experiments.

The CoGAN learning results are shown in Figure 2. We found that although the CoGAN was trained without corresponding images, it learned to render corresponding ones for both Task $\mathbb{A}$ and $\mathbb{B}$. This was due to the weight-sharing constraint imposed to the layers that were responsible for decoding high-level semantics. Exploiting the correspondence between the two domains allowed the GAN1 and GAN2 to utilize more capacity in the networks to better fit the training data. Without the weight-sharing constraint, the two GANs just generated two unrelated images in the two domains.

Weight Sharing: We varied the numbers of weight-sharing layers in the generative and discriminative models to create different CoGANs for analyzing the weight-sharing effect for both Task $\mathbb{A}$ and $\mathbb{B}$. Due to lack of proper validation methods, we did a grid search on the training iteration hyperparameter and reported the best performance achieved by each network. For quantifying the performance, we transformed the image generated by the GAN1 to the 2nd domain using the same method employed for generating the training images in the 2nd domain. We then compared the transformed image with the image generated by the GAN2. A perfect pair image generation should result in two identical images. Hence, we used the ratios of agreed pixels between 10K pairs of images generated by each network (10K randomly sampled $\mathbf{z}$) as the performance metric for the network. We trained each network 5 times with different network initialization weights and reported the average pixel agreement ratios over the 5 trials for each network. The results are shown in Figure 3. We observed that the performance was positively correlated with the number of weight-sharing layers in the generative models. With more sharing layers in the generative models, the rendered pairs of images resembled true pairs drawn from the joint distribution more. We also noted that the performance was uncorrelated to the number of weight-sharing layers in the discriminative models. However, we still preferred discriminator weight-sharing because this reduces the total number of network parameters.

Comparison with Conditional Generative Models: We compared the CoGAN with the conditional generative adversarial nets [11, 12] for the pair digit generation tasks. We designed a conditional GAN with the generative and discriminative models identical to those in the CoGAN. The only difference was the conditional GAN took an additional binary variable as input, which controlled the domain of the output image. When the binary variable was 0, it generated an image resembling images in the 1st domain; otherwise, it generated an image resembling images in the 2nd domain. Similarly, no pairs of corresponding images were given during the conditional GAN learning. We applied the conditional GAN to the two pair digit generation tasks and hoped to empirically answer whether a conditional model can be used to learn to render corresponding images in an unsupervised fashion. The pixel agreement ratio was used as the performance metric. The experiment results showed that for Task $\mathbb{A}$, the CoGAN achieved an average ratio of **0.952**, outperforming 0.909 achieved by the conditional GAN. For Task $\mathbb{B}$, the CoGAN achieved a score of **0.967**, which was much better than 0.778 achieved by the conditional GAN. The conditional GAN just generated two different digits with the same randon noise input but different binary variable values. These results showed that the conditional model failed to learn a joint distribution from samples drawn from the marginal distributions. We note that for the case that the supports of the two domains are different such as the color and depth image domains, the conditional model cannot even be applied.

Pair Face Generation: We applied the CoGAN to the task of generation of faces with different attributes. We trained a couple of CoGANs, each for generating a face with an attribute and a corresponding face without the attribute. We used the CelebFaces Attributes dataset [13] for the experiments. The dataset covered large pose variations and background clutters. Each face image had several attributes, including blond hair, smiling, and eyeglasses. The face images with an attribute constituted the 1st domain; and those without the attribute constituted the 2nd domain. No correspondence information between the two domains was given. We resized the images to a

Figure 4: Pair face image generation with different attributes. From top to bottom, the figure shows pair face generation results for the blond-hair, smiling, and eyeglasses attributes. For each pair, the 1st row contains faces with the attribute, while the 2nd row contains faces without the attribute. The CoGAN learned to generate pari of faces with and without an attribute in an unsupervised fashion.

resolution of 132×132 and randomly sampled 128×128 regions for training. The training procedure and hyperparameters were the same as those used in the digit experiments. The generative and discriminative models were both 7 layer deep convolutional neural networks.

The experiment results are visualized in Figure 4. We randomly sampled two points in the 100-dimensional input noise space and visualized the rendered faces as traveling from one pint to the other point. We found the CoGAN generated pairs of corresponding faces, resembling those from the same person with and without an attribute. As traveling in the space, the faces gradually change from one person to another. Such deformations were consistent for both domains, which verified the CoGAN framework. Note that it is often difficult to create a dataset with correspondences for some attribute such as blond hair since the subjects have to have their hair colored. It is more ideal to have an unsupervised approach. We also noted that the number of faces with an attribute was often several times smaller than that without the attribute in the CelebFaces dataset. However, despite the unbalance, the CoGAN still learned to synthesize pairs of corresponding images in the two domains.

RGB and Depth Image Generation: We used the RGBD dataset [14] and NYU dataset [15] for the experiments. The RGBD dataset contains registered color and depth images of 300 objects captured by the Kinect sensor from different view points. We partitioned the dataset into two equal-size

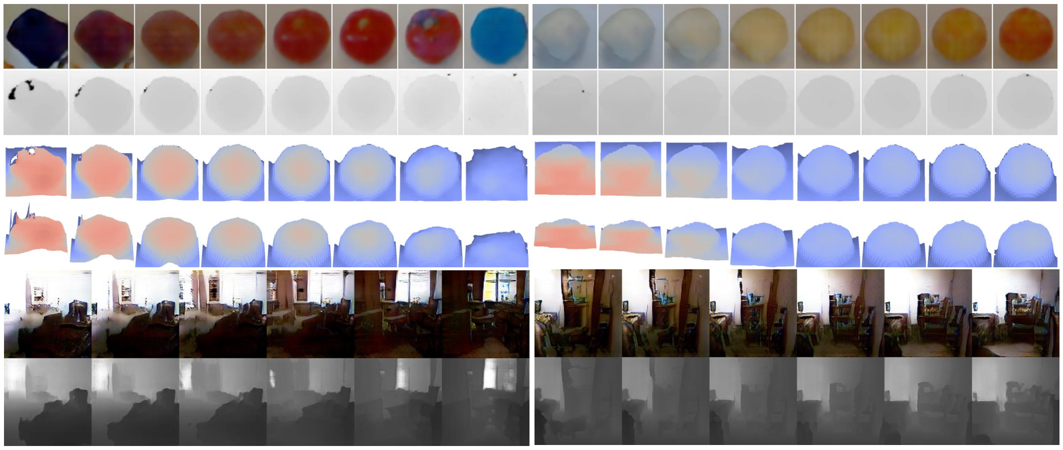

Figure 5: Generation of RGB and depth images. The top figure shows the CoGAN results for the RGBD dataset: the 1st row contains the color images, the 2nd row contains the depth images, and the 3rd and 4th rows visualized the depth profile under different view points. The bottom figure shows the CoGAN results for the NYU dataset.

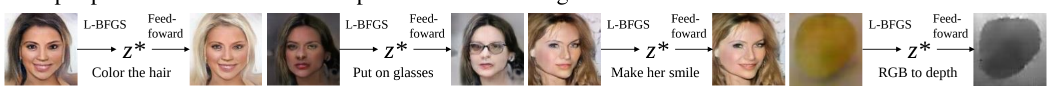

Figure 6: Cross-domain image transformation results.

Table 1: Achieved classification accuracies of the methods for the unsupervised domain adaptation tasks.

Task \ Method	[16]	[17]	[18]	[19]	CoGAN
MNIST→USPS	0.408	0.467	0.478	0.607	**0.912** $\pm$0.008
USPS→MNIST	0.274	0.355	0.631	0.673	**0.891** $\pm$0.008
Average	0.341	0.411	0.554	0.640	**0.902**

non-overlapping subsets. The color images in the 1st subset were used for training the GAN1, while the depth images in the 2nd subset were used for training the GAN2. There were no corresponding depth and color images in the two subsets. The images in the RGBD dataset have different resolutions. We resized them to a fixed resolution of 64×64. The NYU dataset contains color and depth images captured from indoor scenes using the Kinect sensor. We used the 1449 processed depth images for the depth domain. The training images for the color domain were from all the color images in the raw dataset except for those registered with the processed depth images. We resized both the depth and color images to a resolution of 176×132 and randomly cropped 128×128 patches for training. We used the same training hyperparameters in the digit experiments.

Figure 5 showed the learning results. We found the rendered color and depth images resembled corresponding RGB and depth image pairs despite of no registered images existed in the two domains in the training set. The CoGAN recovered the appearance–depth correspondence unsupervisedly.

5 Applications

In addition to rendering novel pairs of corresponding images for movie and game production, the CoGAN finds applications in the unsupervised domain adaptation and image transformation tasks.

Unsupervised Domain Adaptation (UDA): It concerns adapting a classifier trained in one domain to classify samples in a new domain where there is no labeled sample in the new domain for re-training the classifier. Early works have explored ideas from subspace learning [16, 17] to deep discriminative network learning [18, 19]. Here, we show that the CoGAN can be extended to dealing with the UDA problem. We studied adapting the digit classifier from the MNIST dataset to the USPS dataset. Due to domain shift, a classifier trained using one dataset achieves poor performance in the other. We followed the experiment protocol in [16, 19], which randomly samples 2000 images from the MNIST dataset, denoted as D_1, and 1800 images from the USPS dataset, denoted as D_2, to define an UDA problem. The USPS digits have a different resolution. We resized them to have the same resolution as the MNIST digits. We employed the CoGAN used for the digit generation task. For digit classification, we attached a softmax layer to the last hidden layer of the discriminative models. We trained the CoGAN by jointly solving the digit classification problem in the MNIST domain which used the images and labels in D_1 and the CoGAN learning problem which used the images in both D_1 and D_2. This produced two classifiers: $c_1(\mathbf{x}_1) \equiv c(f_1^{(3)}(f_1^{(2)}(f_1^{(1)}(\mathbf{x}_1))))$ for MNIST and $c_2(\mathbf{x}_2) \equiv c(f_2^{(3)}(f_2^{(2)}(f_2^{(1)}(\mathbf{x}_2))))$ for USPS. No label information in D_2 was used. Note that $f_1^{(2)} \equiv f_2^{(2)}$ and $f_1^{(3)} \equiv f_2^{(3)}$ due to weight sharing and c denotes the softmax layer. We then applied c_2 to classify digits in the USPS dataset. The classifier adaptation from USPS to MNIST can be achieved in the same way. The learning hyperparameters was determined via a validation set. We reported the average accuracy over 5 trails with different randomly selected D_1 and D_2.

Table 1 reports the performance of the proposed CoGAN approach with comparison to the state-of-the-art methods for the UDA task. The results for the other methods were duplicated from [19]. From the table, we observed that the proposed method significantly outperformed the state-of-the-art methods. It improved the accuracy from 0.64 to 0.90, which translates to a **72%** error reduction rate.

Cross-domain Image Transformation: Let $\mathbf{x}_1$ be an image in the 1st domain. The cross-domain transformation task aims at finding the corresponding image in the 2nd domain, $\mathbf{x}_2$, such that the joint probability density, $p(\mathbf{x}_1, \mathbf{x}_2)$, is maximized. Let $\mathcal{L}$ be a loss function measuring difference between two images. Given g_1 and g_2, the transformation can be achieved by first solving $\mathbf{z}^* = arg\min_{\mathbf{z}} \mathcal{L}(g_1(\mathbf{z}), \mathbf{x}_1)$. After finding $\mathbf{z}^*$, one can apply g_2 to obtain the transformed image, $\mathbf{x}_2 = g_2(\mathbf{z}^*)$. In Figure 6, we show several CoGAN cross-domain transformation results, computed by using the Euclidean loss function and the L-BFGS optimization algorithm. We found the transformation was successful when the input image was covered by g_1 (or the input image can be generated by g_1) but generated a blurry image when it was not covered. To improve the coverage, we hypothesize that more training images and a better objective function are required, which are left as future work.

6 Related Work

Neural image generation has recently received an increasing attention. Several approaches, including generative adversarial nets[3], variational autoencoders (VAE)[20], attention models[21], moment matching[22], stochastic back-propagation[23], and diffusion processes[24], have recently shown that a deep network can learn to generate realistic images. It can learn to model an image distribution from samples drawn from the distribution. Our work was built on the earlier works, particularly [3]. But we studied a different problem—the unsupervised pair image generation problem. We were interested in whether the joint distribution of images in different domains can be learned from samples drawn separately from the marginal distributions of the individual domains (correspondence information unavailable). Our work is different to the Attribute2Image work[25], which is based on a conditional VAE model [26]. The conditional model can be used to generate images of different styles, but they are unsuitable for generating images in two different domains such as color and depth image domains.

Following [3], several works improved the image generation quality of GAN, including a Laplacian pyramid implementation[27], a deeper architecture[10], and conditional models[11, 12]. Our work extended the GAN to dealing with the pair image generation problem.

Our work is related to the multi-modal learning works, including the joint embedding space learning work[28] and the multi-modal Boltzmann machine works[1, 29]. These approaches can be used for generating corresponding samples in different domains only when correspondence annotations are available during training. The same limitation is also applied to dictionary learning-based approaches [2, 30]. Our work is also related to the cross-domain image generation works[31, 32, 33], which studied transforming an image in one style to the corresponding images in another style. Our work focus on learning the joint distribution in an unsupervised fashion, while [31, 32, 33] focus on learning a transformation function directly in a supervised fashion.

7 Conclusion

We presented the CoGAN framework for learning to generate corresponding images in two different domains. We showed that via enforcing a simple weight-sharing constraint to the layers that are responsible for decoding abstract semantics, the CoGAN learned to generate pairs of corresponding images in an unsupervised fashion. This proved that the joint distribution of images in two different domains can be learned via samples drawn from the marginal distributions. In addition to convincing image generation results on faces and RGBD images, we also showed promising results of the CoGAN framework for the image transformation and unsupervised domain adaptation tasks.

References

[1] Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, 2012.

[2] Shenlong Wang, Lei Zhang, Yan Liang, and Quan Pan. Semi-coupled dictionary learning with applications to image super-resolution and photo-sketch synthesis. In *CVPR*, 2012.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[5] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. In *ICLR*, 2016.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[10] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2016.

[11] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014.

[12] Jon Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N*, 2014.

[13] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[14] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*, 2011.

[15] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[16] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip Yu. Transfer feature learning with joint distribution adaptation. In *ICCV*, 2013.

[17] Basura Fernando, Tatiana Tommasi, and Tinne Tuytelaars. Joint cross-domain classification and subspace learning for unsupervised adaptation. *Pattern Recognition Letters*, 65:60–66, 2015.

[18] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv:1412.3474*, 2014.

[19] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *arXiv:1603.06432*, 2016.

[20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[21] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *ICML*, 2015.

[22] Yujia Li, Kevin Swersky, and Richard Zemel. Generative moment matching networks. *ICML*, 2015.

[23] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *ICML*, 2014.

[24] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

[25] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. *arXiv:1512.00570*, 2015.

[26] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

[27] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.

[28] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv:1411.2539*, 2014.

[29] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *ICML*, 2011.

[30] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE TIP*, 2010.

[31] Junho Yim, Heechul Jung, ByungIn Yoo, Changkyu Choi, Dusik Park, and Junmo Kim. Rotating your face using multi-task deep neural network. In *CVPR*, 2015.

[32] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *NIPS*, 2015.

[33] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015.

Coupled Generative Adversarial Nets Learning Algorithm

We present the learning algorithm for the coupled generative adversarial nets in Algorithm 1. The algorithm is an extension of the learning algorithm for the generative adversarial nets (GAN) to the case of training two GANs with weight sharing constraints. The convergence property follows the results shown in [3].

Algorithm 1 Mini-batch stochastic gradient descent for training coupled generative adversarial nets.

Initialize the network parameters $\boldsymbol{\theta}_{f_1^{(i)}}$'s $\boldsymbol{\theta}_{f_2^{(i)}}$'s $\boldsymbol{\theta}_{g_1^{(i)}}$'s and $\boldsymbol{\theta}_{g_2^{(i)}}$'s with the shared network connection weights set to the same values.

for $t = 0, 1, 2, ...,$ maximum number of iterations **do**

Draw N samples from p_Z, $\{\mathbf{z}^1, \mathbf{z}^2, ..., \mathbf{z}^N\}$

Draw N samples from p_{X_1}, $\{\mathbf{x}_1^1, \mathbf{x}_1^2, ..., \mathbf{x}_1^N\}$

Draw N samples from p_{X_2}, $\{\mathbf{x}_2^1, \mathbf{x}_2^2, ..., \mathbf{x}_2^N\}$

Compute the gradients of the parameters of the discriminative model, f_1^t, $\Delta\boldsymbol{\theta}_{f_1^{(i)}}$;

$$\nabla_{\boldsymbol{\theta}_{f_1^{(i)}}} \frac{1}{N} \sum_{j=1}^{N} -\log f_1^t(\mathbf{x}_1^j) - \log\Big(1 - f_1^t\big(g_1^t(\mathbf{z}^j)\big)\Big)$$

Compute the gradients of the parameters of the discriminative model, f_2^t, $\Delta\boldsymbol{\theta}_{f_2^{(i)}}$;

$$\nabla_{\boldsymbol{\theta}_{f_2^{(i)}}} \frac{1}{N} \sum_{j=1}^{N} -\log f_2^t(\mathbf{x}_2^j) - \log\Big(1 - f_2^t\big(g_2^t(\mathbf{z}^j)\big)\Big)$$

Average the gradients of the shared parameters of the discriminative models.

Compute f_1^{t+1} and f_2^{t+1} according to the gradients.

Compute the gradients of the parameters of the generative model, g_1^t, $\Delta\boldsymbol{\theta}_{g_1^{(i)}}$;

$$\nabla_{\boldsymbol{\theta}_{g_1^{(i)}}} \frac{1}{N} \sum_{j=1}^{N} -\log\Big(1 - f_1^{t+1}\big(g_1^t(\mathbf{z}^j)\big)\Big)$$

Compute the gradients of the network parameters of the generative model, g_2, $\Delta\boldsymbol{\theta}_{g_2^{(i)}}$;

$$\nabla_{\boldsymbol{\theta}_{g_2^{(i)}}} \frac{1}{N} \sum_{j=1}^{N} -\log\Big(1 - f_2^{t+1}\big(g_2^t(\mathbf{z}^j)\big)\Big)$$

Average the gradients of the shared parameters of the generative models.

Compute g_1^{t+1} and g_2^{t+1} according to the gradients.

end for

8 Training Datasets

In Figure 7, Figure 8, Figure 9, and Figure 10, we show several example images of the training images used for the pair image generation tasks in the experiment section. Table 2, Table 3, Table 4, and Table 5 contain the statistics of the training datasets for the experiments.

Figure 7: Training images for the digit experiments. Left (Task $\mathbb{A}$): The images in the first row are from the original MNIST digit domain, while those in the second row are from the edge image domain. Right (Task $\mathbb{B}$): The images in the first row are from the original MNIST digit domain, while those in the second row are from the negative image domain.

Figure 8: Training images from the CelebA dataset [13].

Figure 9: Training images from the RGBD dataset [14].

Figure 10: Training images from the RGBD dataset [15].

Table 2: Numbers of training images in Domain 1 and Domain 2 in the MNIST experiments.

	Task $\mathbb{A}$ Pair generation of digits and corresponding edge images	Task $\mathbb{B}$ Pair generation of digits and corresponding negative images
# of images in Domain 1	30,000	30,000
# of images in Domain 2	30,000	30,000

Table 3: Numbers of training images of different attributes in the pair face generation experiments.

Attribute	Smiling	Blond hair	Glasses
# of images with the attribute	97,669	29,983	13,193
# of images without the attribute	104,930	172,616	189,406

Table 4: Numbers of RGB and depth training images in the RGBD experiments.

# of RGB images	125,000
# of depth images	125,000

Table 5: Numbers of RGB and depth training images in the NYU experiments.

# of RGB images	514,192
# of depth images	1,449

9 Networks

We present the details of the network architectures used in the experiments. In the CoGAN design, the generative models are based on the fractional length convolutional (FCONV) layers, while the discriminative models are based on the standard convolutional (CONV) layers with the exceptions that the last two layers are based on the fully-connected (FC) layers. The batch normalization (BN) layers [6] are applied after each convolutional layer, which are followed by the parameterized rectified linear unit (PReLU) processing [7]. The sigmoid units and the hyperbolic tangent units are applied to the output layers of the generative models for generating images with desired pixel range values.

Table 6 shows the CoGAN architecture for the pair digit generation experiments. Table 7 shows the CoGAN architecture for the pair face generation experiments [13]. Table 8 shows the CoGAN architecture for the generation of RGB and depth images experiments trained with the RGBD object dataset [14]. Table 9 shows the CoGAN architecture for the generation of RGB and depth images experiments trained with the NYU indoor scene dataset [15]. The triplets followed the FCONV or CONV denote the parameters of the convolutional layers. In each triplet, the first number denotes the number of neurons, the second number denotes the kernel size of each filter in the layer, and the third number denotes the stride length of each filter in the layer.

Table 6: The CoGAN architecture for the pair generation of digits.

Generative models			
Layer	Domain 1	Domain 2	Shared?
1	FCONV-(N1024,K4x4,S1), BN, PReLU	FCONV-(N1024,K4x4,S1), BN, PReLU	Yes
2	FCONV-(N512,K3x3,S2), BN, PReLU	FCONV-(N512,K3x3,S2), BN, PReLU	Yes
3	FCONV-(N256,K3x3,S2), BN, PReLU	FCONV-(N256,K3x3,S2), BN, PReLU	Yes
4	FCONV-(N128,K3x3,S2), BN, PReLU	FCONV-(N128,K3x3,S2), BN, PReLU	Yes
5	FCONV-(N1,K6x6,S1), Sigmoid	FCONV-(N1,K6x6,S1), Sigmoid	No
Discriminative models			
Layer	Domain 1	Domain 2	Shared?
1	CONV-(N20,K5x5,S1), POOL-(MAX,2)	CONV-(N20,K5x5,S1), POOL-(MAX,2)	No
2	CONV-(N50,K5x5,S1), POOL-(MAX,2)	CONV-(N50,K5x5,S1), POOL-(MAX,2)	Yes
3	FC-(N500), PReLU	FC-(N500), PReLU	Yes
4	FC-(N1), Sigmoid	FC-(N1), Sigmoid	Yes

Table 7: The CoGAN architecture for the pair generation of faces.

Generative models			
Layer	Domain 1	Domain 2	Shared?
1	FCONV-(N1024,K4x4,S1), BN, PReLU	FCONV-(N1024,K4x4,S1), BN, PReLU	Yes
2	FCONV-(N512,K4x4,S2), BN, PReLU	FCONV-(N512,K4x4,S2), BN, PReLU	Yes
3	FCONV-(N256,K4x4,S2), BN, PReLU	FCONV-(N256,K4x4,S2), BN, PReLU	Yes
4	FCONV-(N128,K4x4,S2), BN, PReLU	FCONV-(N128,K4x4,S2), BN, PReLU	Yes
5	FCONV-(N64,K4x4,S2), BN, PReLU	FCONV-(N64,K4x4,S2), BN, PReLU	Yes
6	FCONV-(N32,K4x4,S2), BN, PReLU	FCONV-(N32,K4x4,S2), BN, PReLU	No
7	FCONV-(N3,K3x3,S1), TanH	FCONV-(N3,K3x3,S1), TanH	No
Discriminative models			
Layer	Domain 1	Domain 2	Shared?
1	CONV-(N32,K5x5,S2), BN, PReLU	CONV-(N32,K5x5,S2), BN, PReLU	No
2	CONV-(N64,K5x5,S2), BN, PReLU	CONV-(N64,K5x5,S2), BN, PReLU	No
3	CONV-(N128,K5x5,S2), BN, PReLU	CONV-(N128,K5x5,S2), BN, PReLU	Yes
4	CONV-(N256,K3x3,S2), BN, PReLU	CONV-(N256,K3x3,S2), BN, PReLU	Yes
5	CONV-(N512,K3x3,S2), BN, PReLU	CONV-(N512,K3x3,S2), BN, PReLU	Yes
6	CONV-(N1024,K3x3,S2), BN, PReLU	CONV-(N1024,K3x3,S2), BN, PReLU	Yes
7	FC-(N2048), BN, PReLU	FC-(N2048), BN, PReLU	Yes
8	FC-(N1), Sigmoid	FC-(N1), Sigmoid	Yes

Table 8: The CoGAN architecture for the pair generation of RGB and depth images trained with the RGBD object dataset.

Generative models			
Layer	Domain 1	Domain 2	Shared?
1	FCONV-(N1024,K4x4,S1), BN, PReLU	FCONV-(N1024,K4x4,S1), BN, PReLU	Yes
2	FCONV-(N512,K4x4,S2), BN, PReLU	FCONV-(N512,K4x4,S2), BN, PReLU	Yes
3	FCONV-(N256,K4x4,S2), BN, PReLU	FCONV-(N256,K4x4,S2), BN, PReLU	Yes
4	FCONV-(N128,K4x4,S2), BN, PReLU	FCONV-(N128,K4x4,S2), BN, PReLU	Yes
5	FCONV-(N64,K4x4,S2), BN, PReLU	FCONV-(N64,K4x4,S2), BN, PReLU	Yes
6	FCONV-(N32,K3x3,S1), BN, PReLU	FCONV-(N32,K3x3,S1), BN, PReLU	No
7	FCONV-(N3,K3x3,S1), TanH	FCONV-(N1,K3x3,S1), Sigmoid	No
Discriminative models			
Layer	Domain 1	Domain 2	Shared?
1	CONV-(N32,K5x5,S2), BN, PReLU	CONV-(N32,K5x5,S2), BN, PReLU	No
2	CONV-(N64,K5x5,S2), BN, PReLU	CONV-(N64,K5x5,S2), BN, PReLU	No
3	CONV-(N128,K5x5,S2), BN, PReLU	CONV-(N128,K5x5,S2), BN, PReLU	Yes
4	CONV-(N256,K3x3,S2), BN, PReLU	CONV-(N256,K3x3,S2), BN, PReLU	Yes
5	CONV-(N512,K3x3,S2), BN, PReLU	CONV-(N512,K3x3,S2), BN, PReLU	Yes
6	CONV-(N1024,K3x3,S2), BN, PReLU	CONV-(N1024,K3x3,S2), BN, PReLU	Yes
7	FC-(N2048), BN, PReLU	FC-(N2048), BN, PReLU	Yes
8	FC-(N1), Sigmoid	FC-(N1), Sigmoid	Yes

Table 9: The CoGAN architecture for the pair generation of RGB and depth images trained with the NYU indoor scene dataset.

Generative models			
Layer	Domain 1	Domain 2	Shared?
1	FCONV-(N1024,K4x4,S1), BN, PReLU	FCONV-(N1024,K4x4,S1), BN, PReLU	Yes
2	FCONV-(N512,K4x4,S2), BN, PReLU	FCONV-(N512,K4x4,S2), BN, PReLU	Yes
3	FCONV-(N256,K4x4,S2), BN, PReLU	FCONV-(N256,K4x4,S2), BN, PReLU	Yes
4	FCONV-(N128,K4x4,S2), BN, PReLU	FCONV-(N128,K4x4,S2), BN, PReLU	Yes
5	FCONV-(N64,K4x4,S2), BN, PReLU	FCONV-(N64,K4x4,S2), BN, PReLU	Yes
6	FCONV-(N32,K4x4,S2), BN, PReLU	FCONV-(N32,K4x4,S2), BN, PReLU	No
7	FCONV-(N3,K3x3,S1), TanH	FCONV-(N1,K3x3,S1), Sigmoid	No
Discriminative models			
Layer	Domain 1	Domain 2	Shared?
1	CONV-(N32,K5x5,S2), BN, PReLU	CONV-(N32,K5x5,S2), BN, PReLU	No
2	CONV-(N64,K5x5,S2), BN, PReLU	CONV-(N64,K5x5,S2), BN, PReLU	No
3	CONV-(N128,K5x5,S2), BN, PReLU	CONV-(N128,K5x5,S2), BN, PReLU	Yes
4	CONV-(N256,K3x3,S2), BN, PReLU	CONV-(N256,K3x3,S2), BN, PReLU	Yes
5	CONV-(N512,K3x3,S2), BN, PReLU	CONV-(N512,K3x3,S2), BN, PReLU	Yes
6	CONV-(N1024,K3x3,S2), BN, PReLU	CONV-(N1024,K3x3,S2), BN, PReLU	Yes
7	FC-(N2048), BN, PReLU	FC-(N2048), BN, PReLU	Yes
8	FC-(N1), Sigmoid	FC-(N1), Sigmoid	Yes

10 Visualization

Figure 11 illustrates additional visualization results for the pair digit generation experiments. Figure 12, Figure 13, and Figure 14 present additional visualization results of the pair generation of faces with the blond-hair attribute and without the blond-hair attribute. Figure 15, Figure 16, and Figure 17 present additional visualization results of the pair generation of faces with the smiling attribute and without the smiling attribute. Figure 18, Figure 19, and Figure 20 present additional visualization results of the Pair generation of faces with the eyeglasses attribute and without the eyeglasses attribute. Figure 21, Figure 22, Figure 23, and Figure 24 present additional visualization results of the pair generation of RGB and depth image experiments.

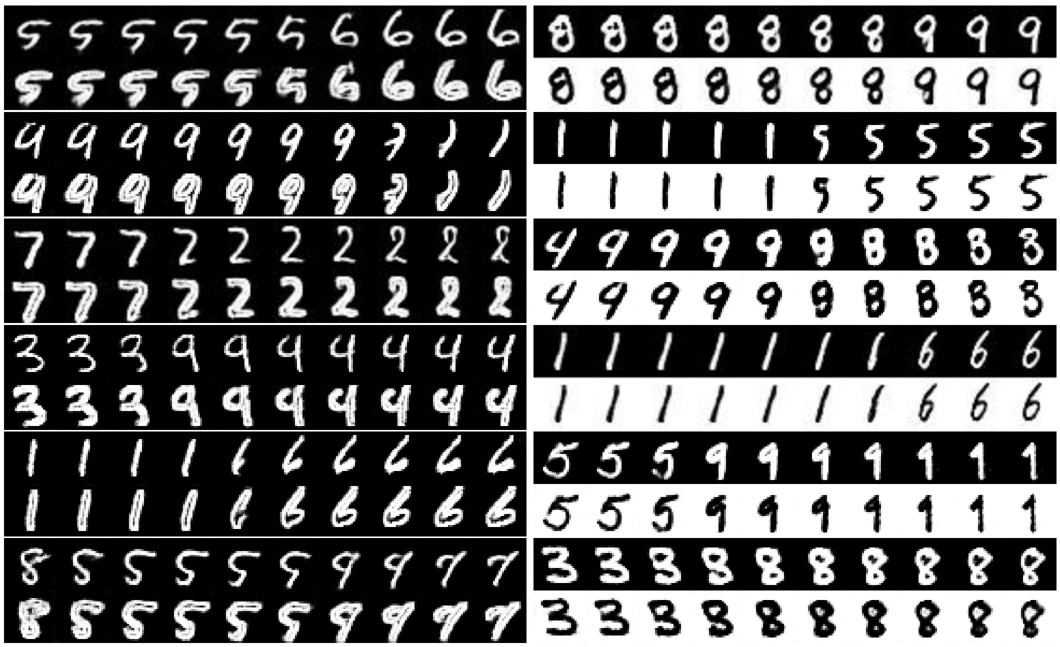

Figure 11: Left: pair generation of digit and corresponding edge images. Right: pair generation of digit and corresponding negative images. We visualized the CoGAN results by rendering pairs of images, using the vectors that corresponded to paths connecting two pints in the input noise space. For each of the sub-figures, the top row was from the GAN1 and the bottom row was from the GAN2. Each of the top and bottom pairs was rendered using the same input noise vector. We observed that for both tasks the CoGAN learned to synthesized corresponding images in the two domains. This was interesting because there were no corresponding images in the training datasets. The correspondences were figured out during training in an unsupervised fashion.

Figure 12: Pair generation of faces with blond hair and without blond hair.

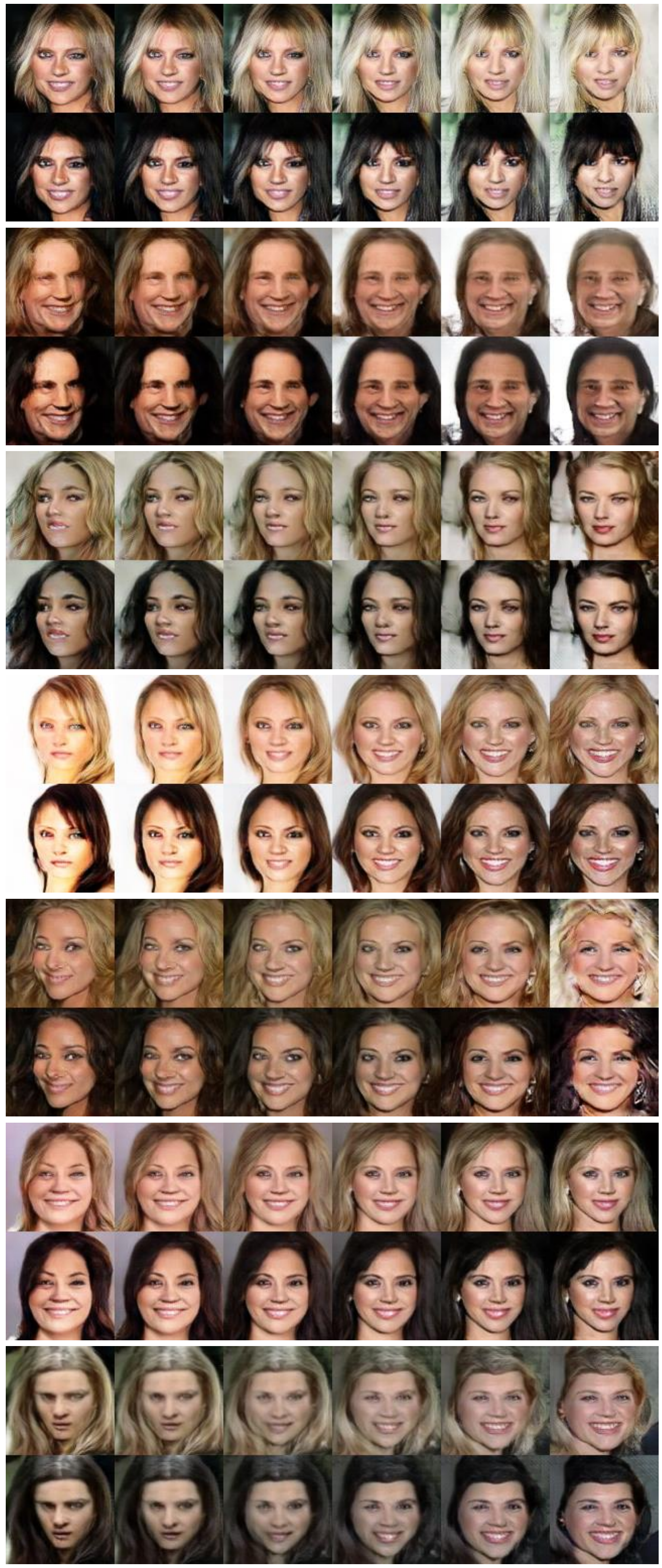

Figure 13: Pair generation of faces with blond hair and without blond hair.

Figure 14: Pair generation of faces with blond hair and without blond hair.

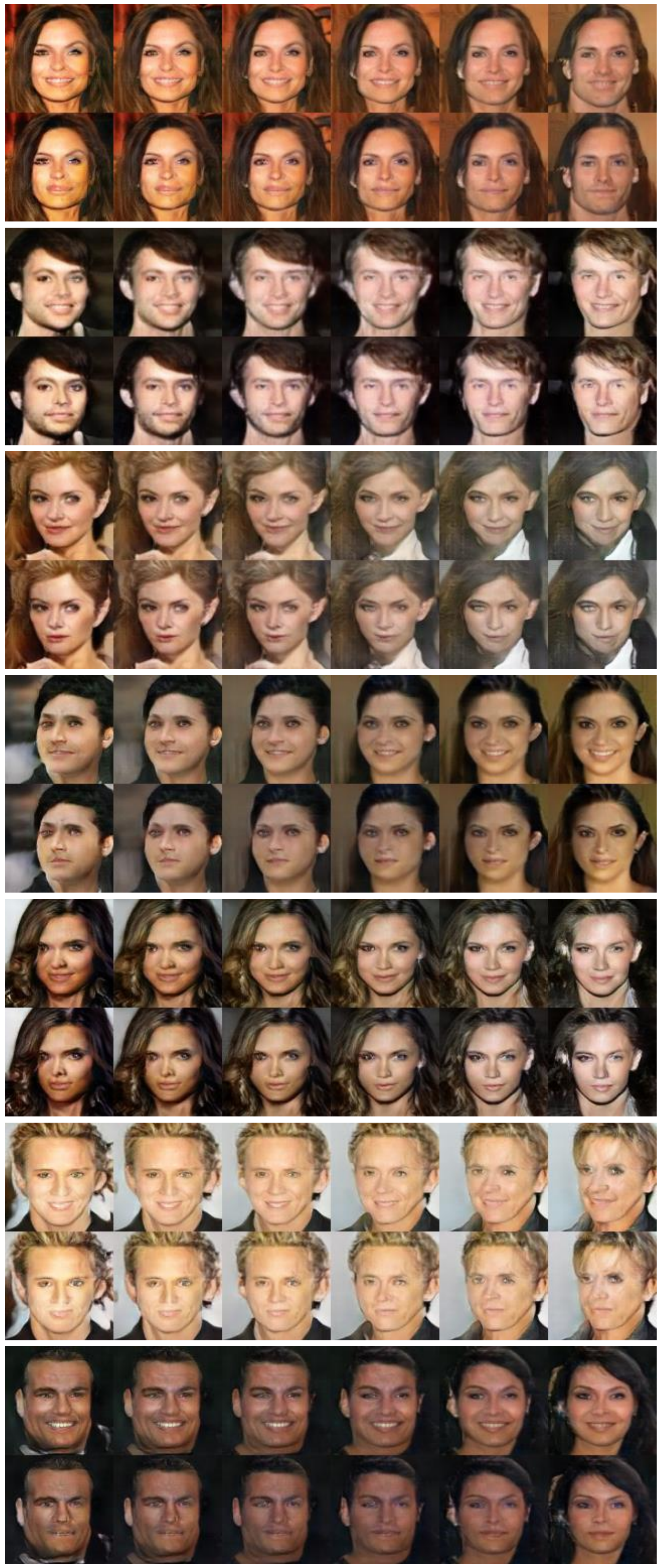

Figure 15: Pair generation of smiling and non-smiling faces.

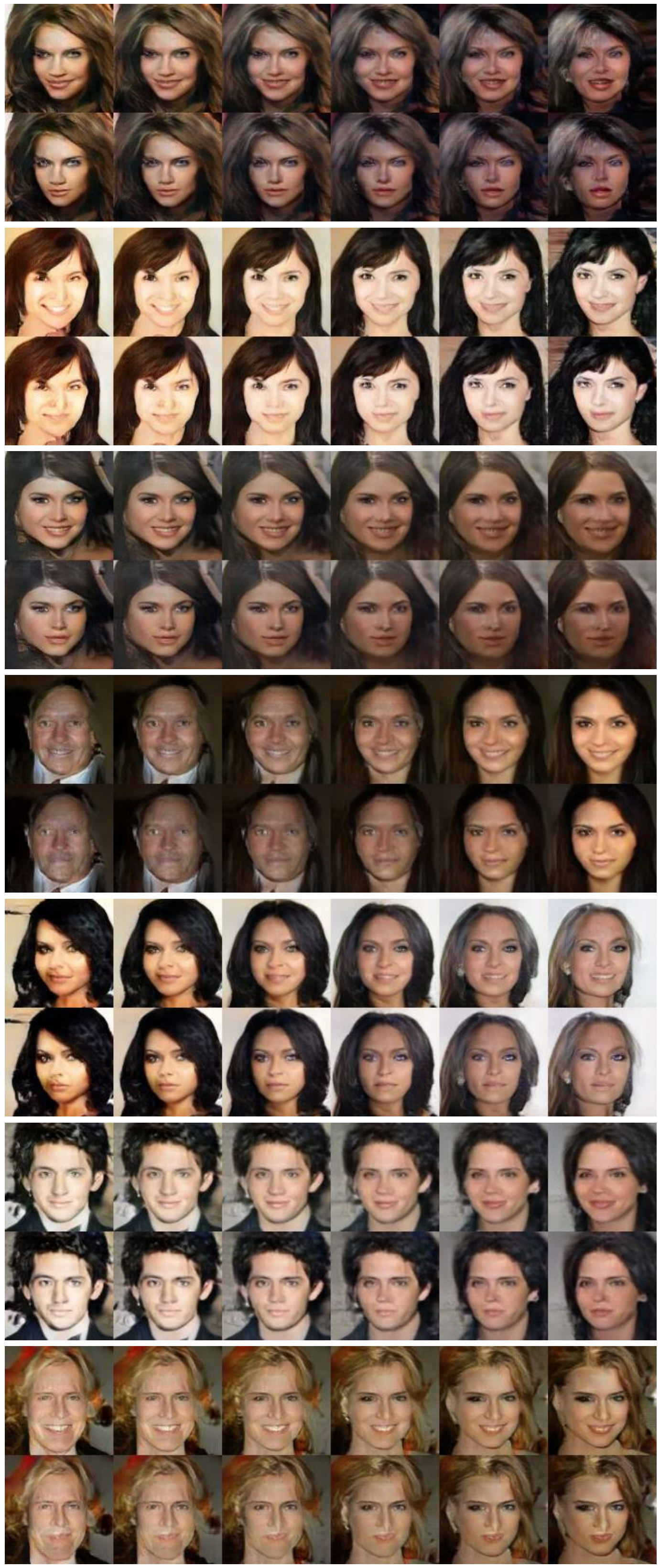

Figure 16: Pair generation of smiling and non-smiling faces.

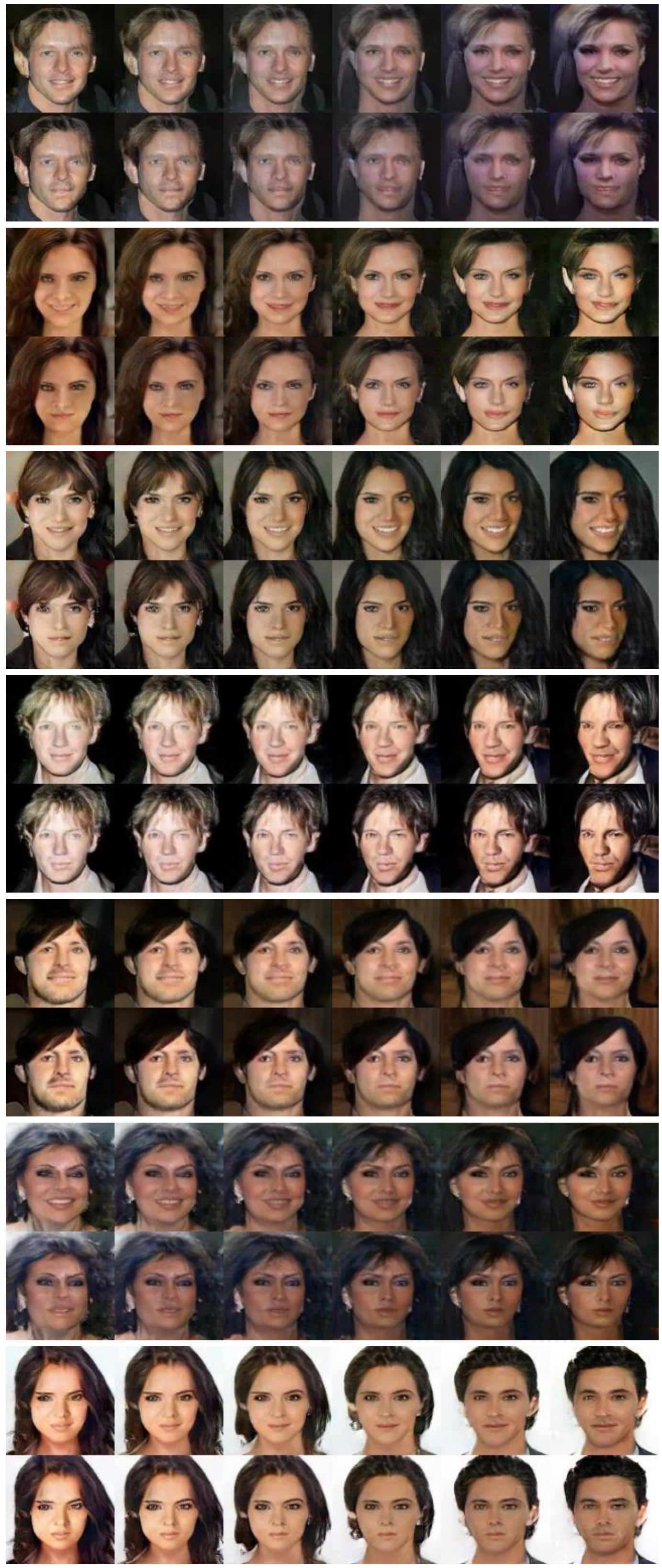

Figure 17: Pair generation of smiling and non-smiling faces.

Figure 18: Pair generation of faces with eyeglasses and without eyeglasses.

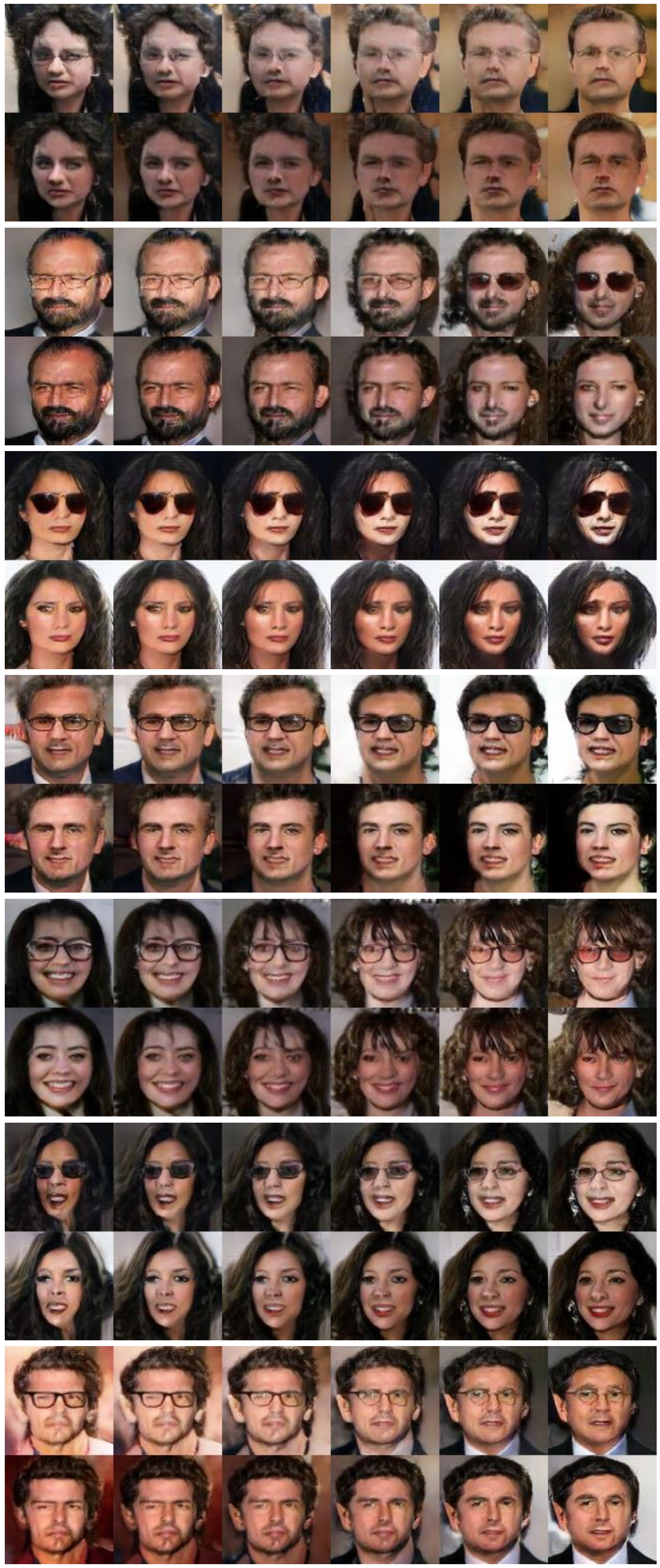

Figure 19: Pair generation of faces with eyeglasses and without eyeglasses.

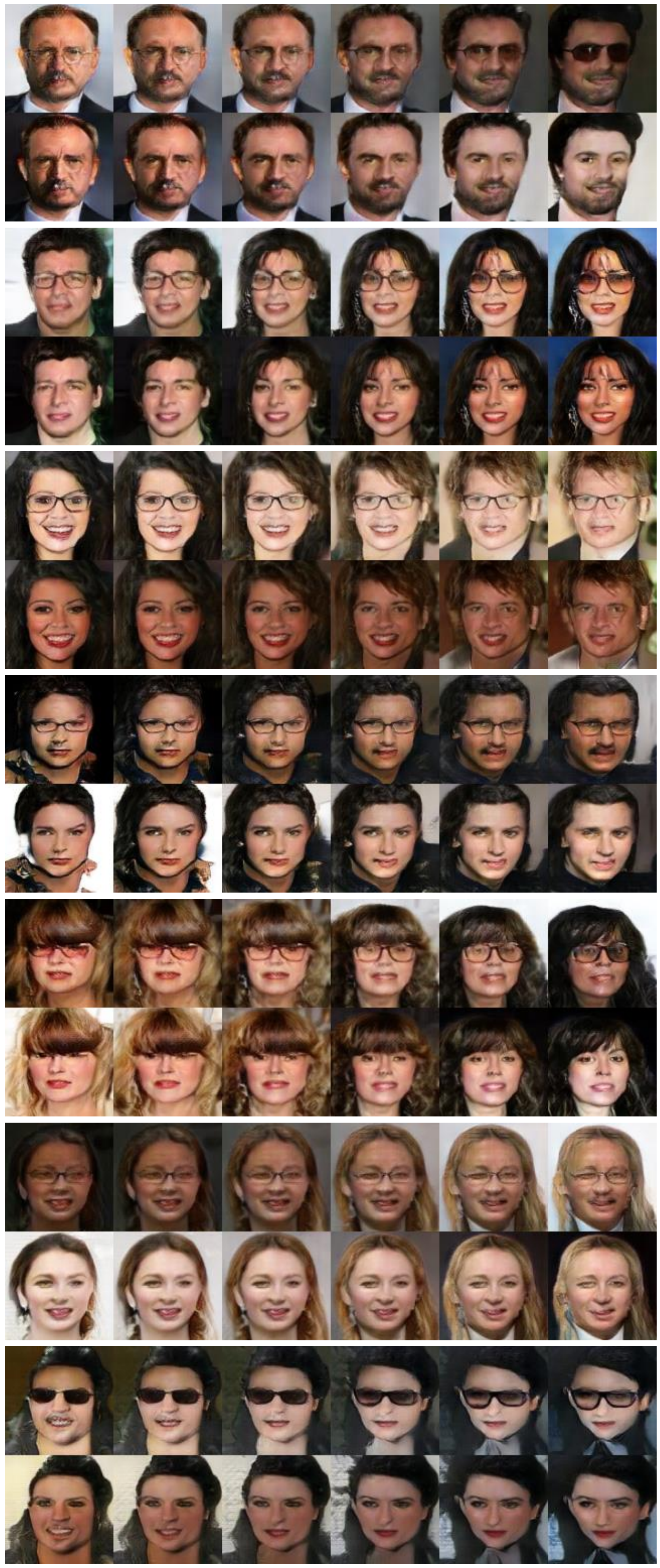

Figure 20: Pair generation of faces with eyeglasses and without eyeglasses.

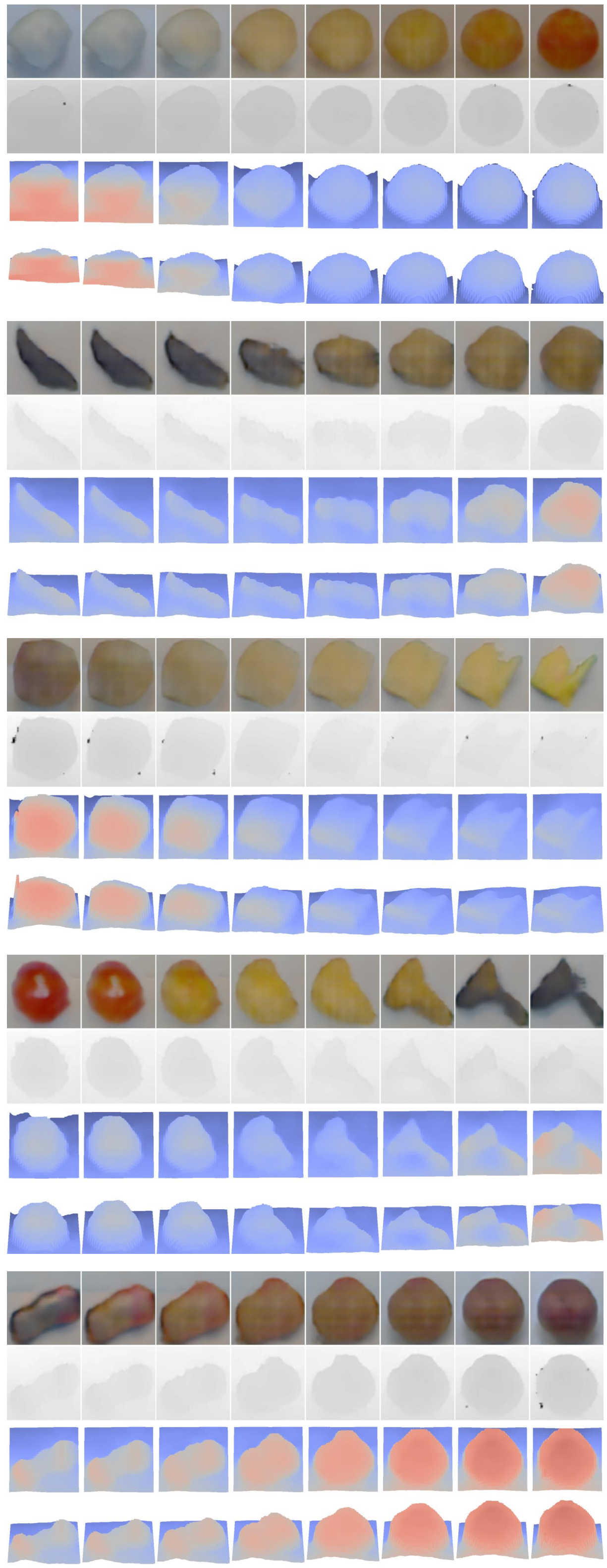

Figure 21: Pair generation of RGB and depth images of objects. The 1st row contains the color images. The 2nd row contains the depth images. The 3rd and 4th rows visualized the point clouds under different view points.

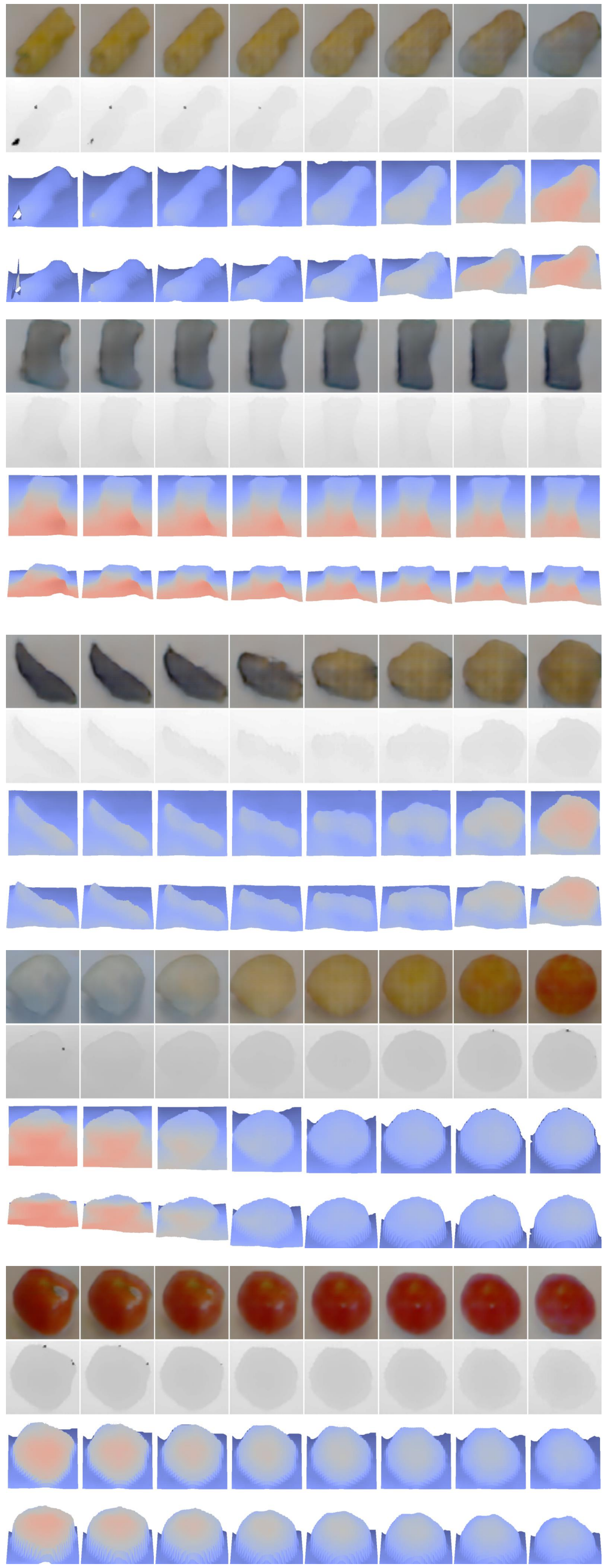

Figure 22: Pair generation of RGB and depth images of objects. The 1st row contains the color images. The 2nd row contains the depth images. The 3rd and 4th rows visualized the point clouds under different view points.

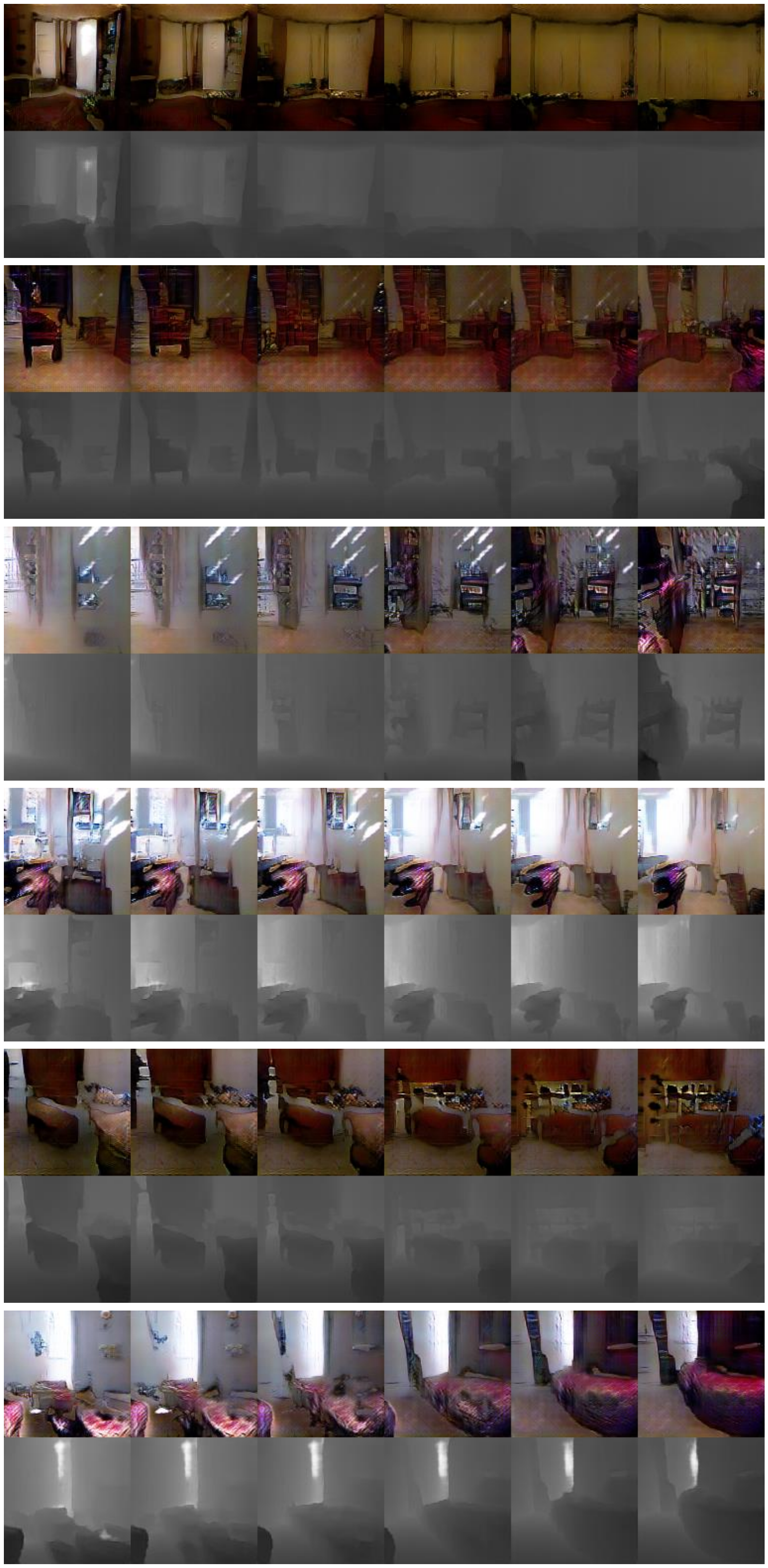

Figure 23: Pair generation of RGB and depth images of indoor scenes.

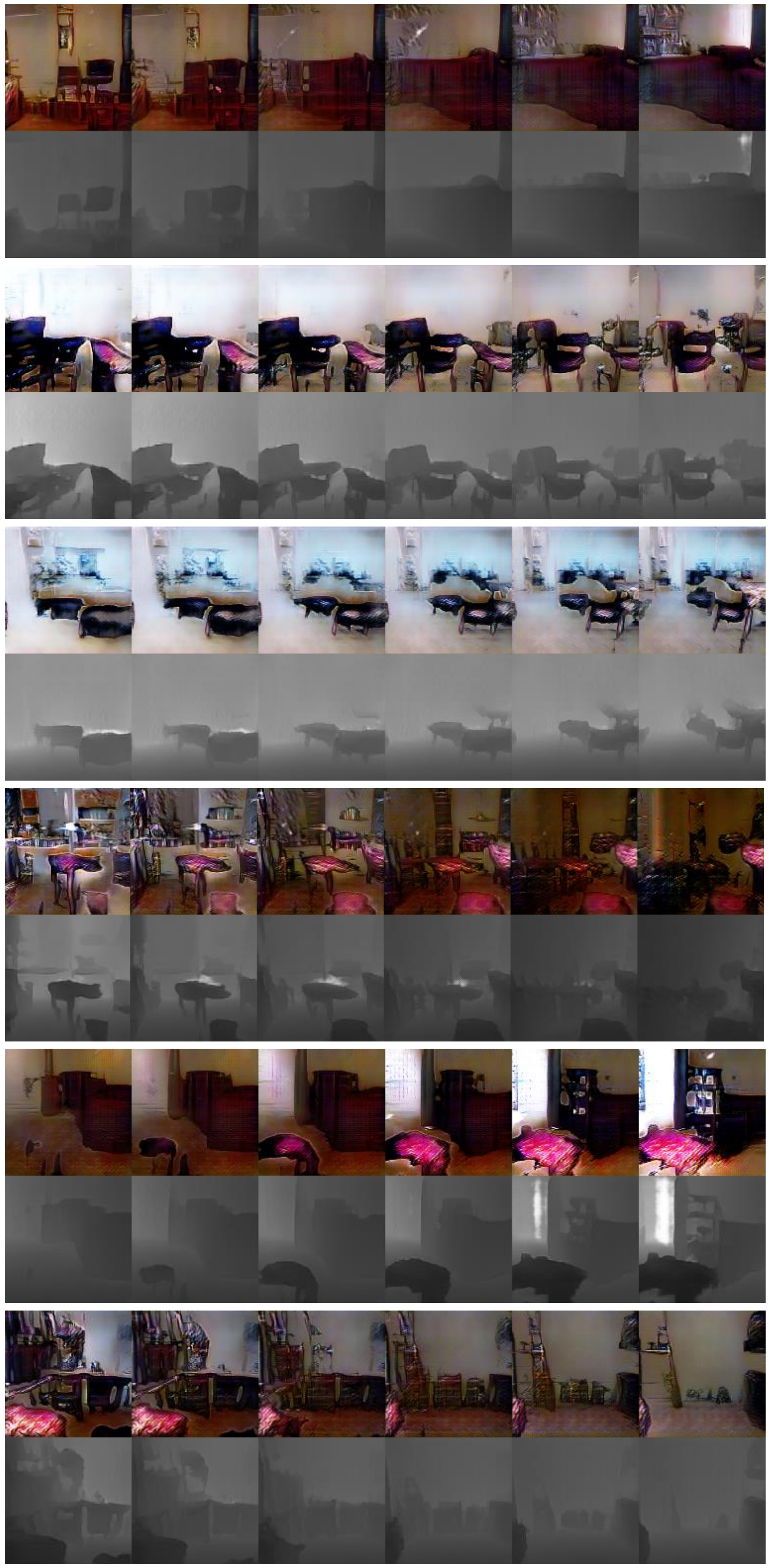

Figure 24: Pair generation of RGB and depth images of indoor scenes.

11 Additional Quantitative Evaluation Discussions

11.1 Weight Sharing

We analyzed the effect of weight sharing in the CoGAN framework. We conducted an experiment where we varied the numbers of weight-sharing layers in the generative and discriminative models to create different CoGAN architectures and trained them with the same hyperparameters. Due to lack of proper validation methods, we did a grid search on the training iteration and reported the best performance achieved by each network configuration for both Task $\mathbb{A}$ and $\mathbb{B}$[2]. For each network architecture, we run 5 trails with different random network initialization weights. We then rendered 10000 pairs of images for each learned network. A pair of images consisted of an image in the first domain (generated by GAN1) and an image in the second domain (generated by GAN2), which were rendered using the same $\mathbf{z}$.

For quantifying the performance of each CoGAN architecture, we transformed the images generated by the GAN1 to the second domain by using the same method employed for generating the training images in the second domain. We then compared the transformed images with the images generated by the GAN2. The performance was measured by the average of the ratios of agreed pixels between the transformed image and the corresponding image in the other domain. Specifically, we rounded the transformed digit image to a binary image and we also rounded the rendered image in the second domain to a binary image. We then compared the pixel agreement ratio—the number of corresponding pixels that have the same value in the two images divided by the total image size. The performance of a trail was given by the pixel agreement ratio of the 10000 pairs of images. The performance of a network configuration was given by the average pixel agreement ratio over the 5 trails. We reported the performance results for Task $\mathbb{A}$ in Table 10 and the performance results for Task $\mathbb{B}$ in Table 11.

From the tables, we observed that the pair image generation performance was positively correlated with the number of weight-sharing layers in the generative models. With more shared layers in the generative models, the rendered pairs of images were resembling more to true pairs drawn from the joint distribution. We noted that the pair image generation performance was uncorrelated to the number of weight-sharing layers in the discriminative models. However, we still preferred applying discriminator weight sharing because this reduces the total number of parameters.

Table 10: The table shows the performance of pair generation of digits and corresponding edge images (Task $\mathbb{A}$) with different CoGAN weight-sharing configurations. The results were the average pixel agreement ratios over 10000 images over 5 trials.

Avg. pixel agreement ratio		Weight-sharing layers in the generative models 5	5,4	5,4,3	5,4,3,2
Weight-sharing		0.894 ± 0.020	0.937 ± 0.004	0.943 ± 0.003	0.951 ± 0.004
layers in the	4	0.904 ± 0.018	0.939 ± 0.002	0.943 ± 0.005	0.950 ± 0.003
discriminative	4,3	0.888 ± 0.036	0.934 ± 0.005	0.946 ± 0.003	0.941 ± 0.024
models	4,3,2	0.903 ± 0.009	0.925 ± 0.021	0.944 ± 0.006	0.952 ± 0.002

Table 11: The table shows the performance of pair generation of digits and corresponding negative images (Task $\mathbb{B}$) with different CoGAN weight-sharing configurations. The results were the average pixel agreement ratios over 10000 images over 5 trials.

Avg. pixel agreement ratio		Weight-sharing layers in the generative models 5	5,4	5,4,3	5,4,3,2
Weight-sharing		0.932 ± 0.011	0.946 ± 0.013	0.970 ± 0.002	0.979 ± 0.001
layers in the	4	0.906 ± 0.066	0.953 ± 0.008	0.970 ± 0.003	0.978 ± 0.001
discriminative	4,3	0.908 ± 0.028	0.944 ± 0.012	0.965 ± 0.009	0.976 ± 0.001
models	4,3,2	0.917 ± 0.022	0.934 ± 0.011	0.955 ± 0.010	0.969 ± 0.008

[2]We noted that the performances were not sensitive to the number of training iterations.

11.2 Comparison with the Conditional Generative Adversarial Nets

We compared the CoGAN framework with the conditional generative adversarial nets (GAN) framework for the pair image generation task. We designed a conditional GAN whose generative and discriminative models were almost identical to those used in the CoGAN in the digit experiments. The only difference was that the conditional GAN took an additional binary variable as input, which controlled the domain of the output image. The binary variable acted as a switch. When the value of the binary variable was zero, it generated images resembling images in the first domain. Otherwise, it generated images resembling those in the second domain. The output layer of the discriminative model was a softmax layer with three neurons. If the first neuron was on, it meant the input to the discriminative model was a synthesized image from the generative model. If the second neuron was on, it meant the input was a real image from the first domain. If the third neuron was on, it meant the input was a real image from the second domain. The goal of the generative model was to render images resembling those from the first domain when the binary variable was zero and to render images resembling those from the second domain when the binary variable was one. The details of the conditional GAN network architecture is shown in Table 12.

Table 12: Network architecture of the conditional GAN

Layer	Generative models
input	$\mathbf{z}$ and conditional variable $c \in \{0, 1\}$
1	FCONV-(N1024,K4x4,S1), BN, PReLU
2	FCONV-(N512,K3x3,S2), BN, PReLU
3	FCONV-(N256,K3x3,S2), BN, PReLU
4	FCONV-(N128,K3x3,S2), BN, PReLU
5	FCONV-(N1,K6x6,S1), Sigmoid

Layer	Discriminative models
1	CONV-(N20,K5x5,S1), POOL-(MAX,2)
2	CONV-(N50,K5x5,S1), POOL-(MAX,2)
3	FC-(N500), PReLU
4	FC-(N3), Softmax

Table 13: Pair Generation Performance Comparison. For each task, we reported the average pixel agreement ratio scores and standard deviations over 5 trails, each trained with a different random initialization of the network connection weights.

Experiment	Task $\mathbb{A}$: Digit and Edge Images	Task $\mathbb{B}$: Digit and Negative Images
Conditional GAN	0.909 ± 0.003	0.778 ± 0.021
CoGAN	$\mathbf{0.952} \pm 0.002$	$\mathbf{0.967} \pm 0.008$

Similarly to the CoGAN learning, no correspondence was given during the conditional GAN learning. We applied the conditional GAN to the two digit generation tasks and hoped to answer whether a conditional model can be used to render corresponding images in two different domains without pairs of corresponding images in the training set. We used the same training data and hyperparameters as those used in the CoGAN learning. We trained the CoGAN for 25000 iterations[3] and used the trained network to render 10000 pairs of images in the two domains. Specifically, each pair of images was rendered with the same $\mathbf{z}$ but with different conditional variable values. These images were used to compute the pair image generation performance of the conditional GAN measured by the average of the pixel agreement ratios. For each task, we trained the conditional GAN for 5 times, each with a different random initialization of the network weights. We reported the average scores and the standard deviations.

The performance results are reported in Table 13. It can be seen that the conditional GAN achieved 0.909 for Task $\mathbb{A}$ and 0.778 for Task $\mathbb{B}$, respectively. They were much lower than the scores of 0.952 and 0.967 achieved by the CoGAN. Figure 25 visualized the conditional GAN's pair generation results, which suggested that the conditional GAN had difficulties in learning to render corresponding images in two different domains without pairs of corresponding images in the training set. We then conclude that the conditional model is unable to learn a joint distribution from samples drawn separately from the marginal distributions of the individual domains.

[3] We note that the pair image generation performance of the conditional GAN did not change much after 5000 iterations.

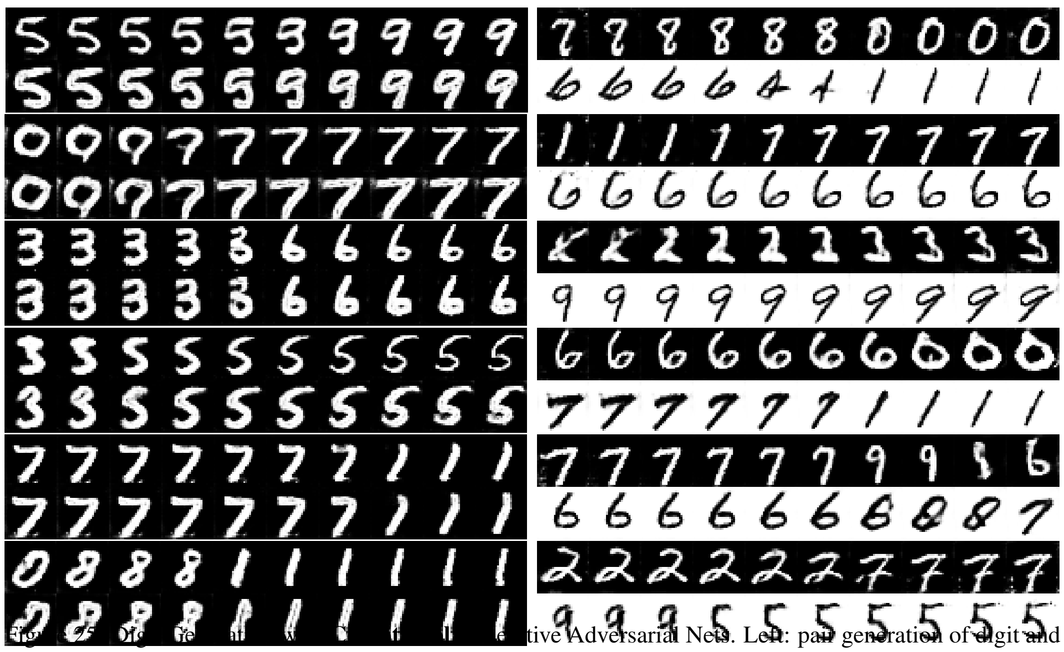

Figure 25: Digit Generation with Conditional Generative Adversarial Nets. Left: pair generation of digit and corresponding edge images. Right: pair generation of digit and corresponding negative images. We visualized the conditional GAN results by rendering pairs of images, using the vectors that corresponded to paths connecting two pints in the input space. For each of the sub-figures, the top row was from the conditional GAN with the conditional variable set to 0, and the bottom row was from the conditional GAN with the conditional variable set to 1. That is each of the top and bottom pairs was rendered using the same input vector except for the conditional variable value. The conditional variable value was used to control the domain of the output images. From the figure, we observed that, although the conditional GAN learned to generate realistic digit images, it failed to learn the correspondence in the two domains. For the edge task, the conditional GAN rendered images of the same digits with a similar font. The edge style was not well-captured. For the negative image generation task, the conditional GAN simply failed to capture any correspondence. The rendered digits with the same input vector but different conditional variable values were not related. This showed that the conditional GAN is not suited for learning to render corresponding images in a unsupervised fashion. On the contrary, the proposed CoGAN framework can learn to generate pairs of corresponding images without pairs of corresponding images in the training dataset as shown in Figure 11.