# Improving Unpaired Object Translation for Unaligned Domains

**Michael Snower**
Department of Computer Science
Brown University
Advisor: James Tompkin
`michael_snower@brown.edu`

## Abstract

Generative Adversarial Networks have shown promise in unpaired image translation. However, translating unpaired objects from unaligned domains is an unsolved problem. Existing methods are restricted to domain pairs which require only minor shape change, and they typically mode collapse on more challenging domain pairs such as giraffe to sheep, or racket to bat.

First, with a simple toy dataset, we show that existing image translation models are inherently unequipped to handle significant shape change because they fail to disentangle the foreground and background. We present a novel method and network architecture to address this issue. The effectiveness of our network is confirmed by a user study, where we are preferred $2\times$ over the SOTA.

However, even our improved baseline suffers from instability on these diverse domains. Thus, we explore various multi-scale architectures. Additionally, we improve the efficiency of our networks because previous methods are highly resource intensive and require long training times.

We then design a transformer-based network that is not limited by receptive field, as convolutions are. This enables it to make significant global changes to the object shape. We separate the shape and color translation problems so that we can utilize transformer-based models. Effectively, we translate shapes as if they were sentences. We find this enhances the quality and stability of the shape translations and explore ways our transformer representations can be combined with color networks.

## Acknowledgements

First, I would like to thank James. Your advice and support is what made this work possible.

Thank you to Oliver Wang and Kwang In Kim for the advice they have given me. Also, thank you to Youssef Mejjatti, Aaron Gokaslan, and Zejiang Shen, who contributed to this work.

I would also like to acknowledge Professor Doeppner, JJ, Professor Hazeltine, and Brendan McNally. I have learned so much from you.

## 1 Introduction

Data driven generative models such as generative adversarial networks (GANs) [10] enable exciting new possibilities for image editing. We consider the unpaired image-to-image translation problem of mapping object appearance between domains that are unaligned.

CycleGAN [47] enabled many image-to-image translation problems to be solved because it alleviated the need for paired training data; however, it is not well suited to significantly change the shape of objects. For example, when translating a horse to a zebra, the generator is allowed to make modifications to any part of the image to satisfy the discriminator. This may result in a rider covered in zebra texture. Various GAN architectures have made progress in better restricting changes to preserving the background while translating the foreground. For instance, by learning attention masks to focus translation onto foreground regions [41, 27, 28, 43], or by using masks as additional input to the network and modifying the objective function to penalize changes to the background [22, 26]. Of these, the state-of-the-art InstaGAN [26] approach adds a background preservation weight to constrain image modifications to the foreground regions. However, the network still struggles to translate the foreground and translates the background to compensate. We propose an approach which splits the foreground and background generation process:

1. Generate a foreground from the target distribution,
2. Use a separate generator, trained on the source distribution, to inpaint missing areas of the background,
3. Combine the outputs of steps 1–2 to create the translated image.

This approach is simple. Yet, it is preferred in a user study over InstaGAN on a variety of domains on the challenging COCO [24] dataset. Moreover, it succeeds in an a basic toy dataset on which *all previous image translation models fail*. Since our approach generates foregrounds and backgrounds separately, our backgrounds do not suffer during the foreground translation process. This also improves foregrounds since we constrain the mini-max game between our foreground generator-discriminator: as the generator can no longer rely on changing the background to fool the discriminator, it must dedicate more parameters to translating the actual foreground object class.

Though our solution improves over previous networks, we note that it still mode collapses on some challenging classes or at high resolutions. Other methods have found multi-scale training schemes [16, 17] or networks [33] to improve the stability and quality of generated images. We design a network where information is added at multiple scales. The motivation is similar to that of Feature Pyramid Networks (FPN) [23] where features from multiple resolutions in the feature extraction backbone of an object detector are fused. This allows our model to make both global changes to the image (at the low resolution branches) and fine-grained texture changes (at the high resolution branch). Both of these types of changes are important: global changes enable our network to model signficant shape change between domains; fine-grained changes enable our network to change color, such as a giraffe's hair to a sheep's wool. Also, our network is more efficient than previous methods, which require extensive training time and GPU resources.

To further improve training stability, and improve the quality of our global shape changes, we design a transformer-based network [37] to translate the shape of objects. We translate the color with a CNN in a separate step. To create a representation suitable for transformers, we represent shapes as a sequence of keypoints. We then train our transformer based network in a fashion similar to Lample et al. [20], which translates unpaired sentences from one language to another. Thus, our sequences of keypoints are analogous to the sentences translated in [20].

In summary, our contributions are as follows:

1. A split foreground-background generation process which sets a new SOTA for image translation with significant shape change.
2. A toy dataset benchmark to challenge image translation models.
3. A multi-scale architecture that is more stable and efficient than previous approaches.
4. A transformer-based network to more effectively make global shape changes between unaligned domains.

## 2 Related Work

Zhu et al. [47] (CycleGAN) and Kim et al. [19] (DiscoGAN) both demonstrated the ability to map between two image domains without paired training data. To constrain the problem, Zhu et al. proposed a “cyclic loss" to restrict the domain mappings which can be learned by the generator. In this setting, two pairs of GANs are trained at once, with one generator mapping from domain $S$ to $T$ and the other mapping from domain $T$ to $S$.

Recent work such as DistanceGAN [32] and GcGAN [7] have attempted to remove the need of cycles for unpaired image to image translation. DistanceGAN does so by trying to preserve the pairwise distances between images within a mini-batch. GcGAN extends this work by augmenting this with geometric constraints. While these methods have some promise, they have difficulty handling one-sided mappings when significant and non-global transformations occur between domains [9].

Some improvements have been made to these types of cyclic models to help them translate between two domains of differing texture or shapes [14, 9]. However, these methods consider the entire image holistically, and struggle to deal with both small objects and domains which have different distributions of background texture.

Several recent papers [41, 27, 28, 43] have proposed augmenting these networks with unsupervised attention mechanisms to help separate foreground and background regions. For instance, Yang et al. [44] (LR-GAN) decompose the image with a two-component spatial-transformer generator: one component generates the background, and the other is a recurrent neural network (RNN) which repeatedly ‘pastes’ created foregrounds and their masks into the image. [18] uses an interpolated layer and instance normalization to adapt to different levels of shape change within domains. While promising, all these techniques struggle with object transformation use cases requiring a significant shape change.

Two recent works have shown that notably higher quality results can be achieved when masks are available. Liang et al. [22] (ContrastGAN) and Mo et al. [26] (InstaGAN) use binary mask segmentations to simplify the complex task of localizing objects and segmenting them from the background. However, the effect of background modification is still visible in these methods. We propose a method that reduces the computational complexity of Mo. et. al. Moreover, it is more robust than previous methods and the generator is not allowed to affect the source background appearance, which allows for more flexible and reliable generation. We also build a multi-scale version of our method.

[40] improves shape change by first extracting keypoints from the source domain with [29] in an unsupervised manner inspired by [15]. The keypoints are translated to the target domain with a MLP as done in [1]. However, this is not suitable for directly translating keypoint locations in Cartesian space, as it operates in the keypoint feature space. Our keypoint-based method translates keypoints in Cartesian space using transformers [37].

Transformers, original introduced for language modeling, have shown promise for solving image and video problems. They have been used to model temporal dependencies in video [34, 35, 8]. [25, 4] use a BERT-inspired [6] architecture to interpret multimodal vision and language inputs. [31] directly generate color images with transformers. We use transformers to generate shapes defined by a sequence of keypoints and leverage a separate network to translate the color.

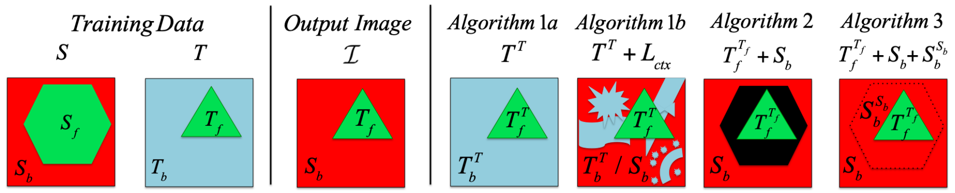

Figure 1: A schematic illustration of our problem and the limtation of existing approaches. Superscript indicates a GAN-generated region and specifies the GAN's training region.

## 3 SplitGAN

### 3.1 Splitting Foreground and Background Translation

We are given two image classes $S$ and $T$. Each comprises color images $s_w \in S$ and $t_w \in T$, with $n_s,\ n_t > 0$ class instances, respectively. We are also given instance masks $s_{m_i}$ and $t_{m_i}$ for each instance in $s$ and $t$. Our goal is to learn a transfer function $f : S_w \rightarrow T_w$ which maps the foreground $s_f$ of $s$ to $t'_f$, where $s_f = (s_{m_1} \cup s_{m_2} \cup \dots \cup s_{m_n})$, while *preserving the source background* $s_b$.

It should be noted that this cannot be accomplished by naïvely segmenting the foreground objects in $s_w$, transferring them to the target domain, and *paste* the transferred objects in the target background in $t_w$: when the translated target foreground is smaller than the source foreground, we must inpaint the background of the output image and therefore, the transfer algorithm should change *both* foreground and background and blend them in visually plausible manner. Figure 1 illustrates this problem and the limitation of existing approaches in this context. First, CycleGAN [25] and attention nets [14] (*Algorithm 1b*) learn only from $T$ and they do not explicitly learn to map the source background $S_b$. When translating to a domain with *smaller* foreground objects, this makes it difficult to generate new background regions which look like $S$. InstaGAN [15] (*Algorithm 1b*) attempts to address this with context loss $L_{ctx}$, but it often generates noticeable background artifacts because its generator learns from all of $T$. These artifacts accumulate when translating multiple instances. In ContrastGAN (*Algorithm 2*) learns from $T_f$ cut out from $T$. Thereafter, the translation $T_f^{T_f}$ is pasted onto $S_b$. This prevents the background artifacts from *Algorithm 1b*. However, if $T_f^{T_f}$ does not completely cover the original foreground $S_f$, then ContrastGAN cannot inpaint these areas of the background.

Our approach is to explicitly address this background inpainting and blending problem by training two generator networks (*Algorithm 3*). The first generator synthesizes $T_f^{T_f}$ to translate the foreground, and the other one generates $S_b^{S_b}$ to inpaint the missing background $(\mathbf{1} + T_f^{T_f} - S_f^{S_f})$.

We adopt a sequential model to translate the foreground. We translate foreground $s_f$ to $t'_f$ by translating each of the $n$ instances, $s_i$ to $t'_i$, with each forward pass of our encoder-decoder foreground generator $G_f$. $G_f$ takes as input the image concatenated with its corresponding instance mask. The empty background $s_b$ defined by the area outside of $s_f$ is inpainted by a separate encoder-decoder generator $G_b$, once for all $n$ instances.

### 3.2 Toy Dataset

To motivate our approach, we introduce a demonstrative toy problem of changing giraffe-textured circles into tiger-textured triangles. These instances are placed on red and blue backgrounds, respectively. The goal is to translate the circles to triangles (and vice-versa) without changing the background (Fig. 2). The dataset consists of 525 training images and 225 testing synthetic images.

CycleGAN [47] is not capable of completing this task: it learns the background distribution from the target images. The backgrounds of the target class are always different from the source class. Thus, CycleGAN does not have the necessary training data to complete the task. InstaGAN [26] has an architecture which uses ground truth instance masks to help translate foreground, and it includes a context loss to penalize background changes. However, in the triangle→circle case, the infilled background region is changed to the wrong color. This is because it only learns from images with

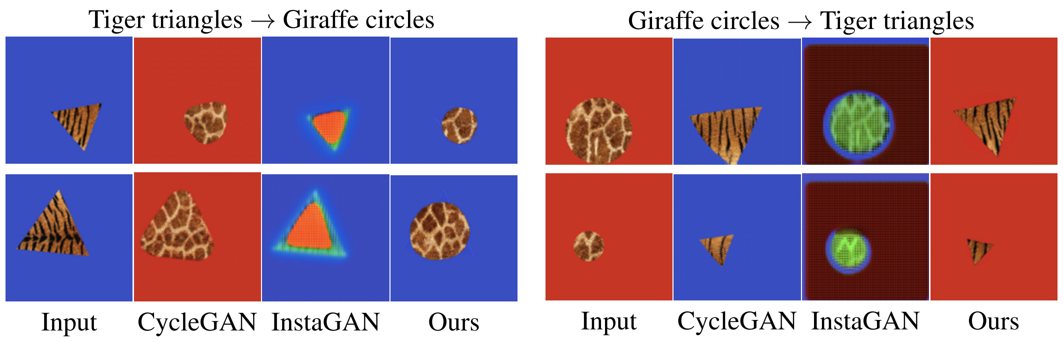
Tiger triangles → Giraffe circles
Giraffe circles → Tiger triangles
Input
CycleGAN
InstaGAN
Ours
Input
CycleGAN
InstaGAN
Ours

Figure 2: *Toy Experiment*. We wish to map the foregrounds between domains with simple textured shapes and solid color backgrounds. The 'tiger triangles' domain has blue backgrounds, while 'giraffe circles' has red backgrounds. CycleGAN modifies the solid color background but not the foreground; InstaGAN is not able to correctly modify the foreground shape or texture and sometimes distorts the background. Our method preserves the background while correctly translating shape and texture.

blue backgrounds. Therefore, any image without a blue background must be classified as fake, which causes the context loss and the GAN loss to behave adversarially.

Our network succeeds at this task. We learn the backgrounds from the source distribution, which lets us inpaint the images with the correct color and translate the foregrounds correctly.

### 3.3 Joint color-mask architecture

Our base generators and discriminators follow a similar architecture as CycleGAN and InstaGAN. For our discriminators, however, we also added a multi-scale discriminator to our foreground GAN as it has proved useful in image translation [13, 38, 39, 46, 45].

We have split the translation process, and so in a final step we must combine the generated foregrounds and backgrounds. Thus, an ideal generator for our application would produce masks which are identical in shape to the foreground. Some prior approaches propose architectures with separate encoder-decoder pairs for the images and the masks [26]: The images are encoded separately from the masks, and during the decoding process, the image and mask decoders each receive various combinations of concatenated input from the encoders. This is paired with a discriminator with a separate encoder for both the images and the masks, but a shared decoder.

Despite the concatenated decoder input, this prior approach produces binary masks which are unsuitable for clean foreground extraction. Instead, we jointly generate both the color image and the mask as concatenations within a single network. Our discriminator similarly takes the image concatenated with its mask, which is now sufficient for our generator to learn shape-consistent images and masks.

### 3.4 A New SOTA

**COCO** We train on several classes from the COCO dataset [24]. We used the Inception Resnet-v2 (Atrous) network [36], from Google's Object Detection API [12], pretrained on COCO to detect our generated instances. We measure the percentage of the detectable instances, the mean confidence score the detector assigns our translations, and also the mean intersection of union (mIoU). The mIoU is a useful metric for assessing whether the network is detecting true instances or false positives. For the ground truth in the mIoU calculation, we create bounding boxes from our generated instance masks. Additionally, we set the minimum confidence threshold to 0.3, to further help alleviate false positives.

Table 1 presents the results. We have a higher mean confidence score than InstaGAN on every translation direction. However, as evident from the data, the standard deviations are large. Thus, for our confidence score, $c_o$ and InstaGAN's confidence score $c_i$, we only bold the confidence scores for which we can say $c_o - c_i > 0$ at a significance level of $\alpha = 0.05$. We evaluate statistical significance with a one-tailed two sample Z-test. In regards to mIoU score, although neither we nor InstaGAN

Table 1: Quantitative bounding box classification evaluation on the translated images (higher is better; $\pm$ scores are one standard deviation). For mIoU, in almost all cases our method outperforms the baseline method, except for Apple$\rightarrow$Donut. For confidences, our method always has a higher mean, but due to the large standard deviations we can not always claim our improvement over InstaGAN is statistically significant. Thus, we only bold values which are better than InstaGAN at significance level $\alpha = 0.05$ in a Z-test. For % instances detected, our method again is always better on these datasets. *Note:* InstaGAN mode collapsed on Frisbee$\rightarrow$Sportsball.

| Dataset | mIoU | | Detection confidence | | % instances detected | |
|---|---|---|---|---|---|---|
| | InstaGAN | Ours | InstaGAN | Ours | InstaGAN | Ours |
| Sheep$\rightarrow$Giraffe | 0.662 | **0.681** | $0.444 \pm 0.435$ | $\mathbf{0.604 \pm 0.408}$ | 56.5 | **74.1** |
| Giraffe$\rightarrow$Sheep | 0.477 | **0.801** | $0.263 \pm 0.388$ | $0.276 \pm 0.392$ | 33.9 | **36.2** |
| Bat$\rightarrow$Racket | 0.586 | **0.633** | $0.275 \pm 0.421$ | $0.305 \pm 0.420$ | 31.6 | **37.5** |
| Racket$\rightarrow$Bat | 0.497 | **0.621** | $0.087 \pm 0.231$ | $\mathbf{0.133 \pm 0.281}$ | 14.2 | **20.9** |
| Sportsball$\rightarrow$Frisbee | 0.300 | **0.613** | $0.108 \pm 0.285$ | $0.136 \pm 0.310$ | 13.9 | **17.3** |
| Frisbee$\rightarrow$Sportsball | Collapse | **0.574** | $0.026 \pm 0.142$ | $\mathbf{0.352 \pm 0.439}$ | 3.8 | **40.9** |
| Bear$\rightarrow$Elephant | 0.758 | **0.841** | $0.316 \pm 0.411$ | $\mathbf{0.599 \pm 0.431}$ | 39.7 | **69.1** |
| Elephant$\rightarrow$Bear | 0.767 | **0.820** | $0.079 \pm 0.240$ | $\mathbf{0.146 \pm 0.312}$ | 10.5 | **19.7** |
| Apple$\rightarrow$Donut | **0.303** | 0.054 | $0.102 \pm 0.234$ | $\mathbf{0.274 \pm 0.370}$ | 18.5 | **38.5** |
| Donut$\rightarrow$Apple | 0.560 | **0.768** | $0.047 \pm 0.173$ | $\mathbf{0.079 \pm 0.224}$ | 8.0 | **12.2** |

Table 2: User preferences in pairwise comparisons. We report the percentage of the time that people preferred our results over InstaGAN's results on COCO dataset classes. *S* is sheep, *G* is giraffe, *R* is racket, *B* is bat, *E* is elephant, *B* is bear, *Sp* is sportsball, *F* is frisbee, *D* is donut, *A* is apple.

| Mean | S$\rightarrow$G | G$\rightarrow$S | R$\rightarrow$B | S$\rightarrow$R | E$\rightarrow$B | B$\rightarrow$E | Sp$\rightarrow$F | F$\rightarrow$Sp | D$\rightarrow$A | A$\rightarrow$D |
|---|---|---|---|---|---|---|---|---|---|---|
| 66.4 | 57.8 | 64.8 | 82.6 | 70.0 | 48.9 | 70.7 | 61.1 | 95.2 | 60.3 | 52.2 |

were free of false positives, we consistently have a higher mIoU score. The only exception is the Apple$\rightarrow$Donut translation. This suggests our confidence score may be inflated for this translation direction.

Additionally, we commissioned a user study where humans evaluated our generated photos in comparison to InstaGANs. In the study, we showed users a ground truth photo, InstaGAN's translation and our translation. We asked them: "Which option changed the 'source domain' to 'target domain' in a more realistic way? Consider the changes made to the entire image." We randomly selected 15 images for each translation direction across 5 translation pairs for a total of 150 image comparisons. We recruited 18 participants, each of whom evaluated all 150 comparisons. Averaged over all datasets, our method was preferred over InstaGAN 66.4% of the time—about twice as often—with a standard deviation of 8.7% (Table 2). A one distribution t-test deems us to be preferred over InstaGAN at least 50% of the time with probability greater than 99.9%.

**Implementation Details** We use the Adam optimizer. Generator and discriminator learning rates are both set to 0.0002 with $\beta_1 = 0.5$, $\beta_2 = 0.999$. In our loss, we set $\lambda_1 = \lambda_2 = 10$ for all our experiments. We resize all images to 200x200. We trained our models for 400 epochs, with a constant learning rate for the first 200 and a linearly decaying rate for the next 200. We train InstaGAN (and CycleGAN) for 200 epochs, as is suggested in their papers. We note that our 400 epochs provides *the same number* of gradient updates as InstaGAN at 200 epochs: We train on 1 instance per image to their 4 instances per image; however, their mini-batch sequential scheme trains on 2 images at once, providing 2 updates per image. All models but one were trained on multiple NVIDIA Tesla P100s or NVIDIA GTX 1080 Ti's; InstaGAN requires 4$\times$ 12GB VRAM; our model requires 2$\times$ 11GB VRAM; CycleGAN requires 1$\times$ 11GB VRAM. We downloaded the InstaGAN Sheep$\rightarrow$Giraffe model from the author's GitHub webpage to save GPU training hours for datasets with published models.

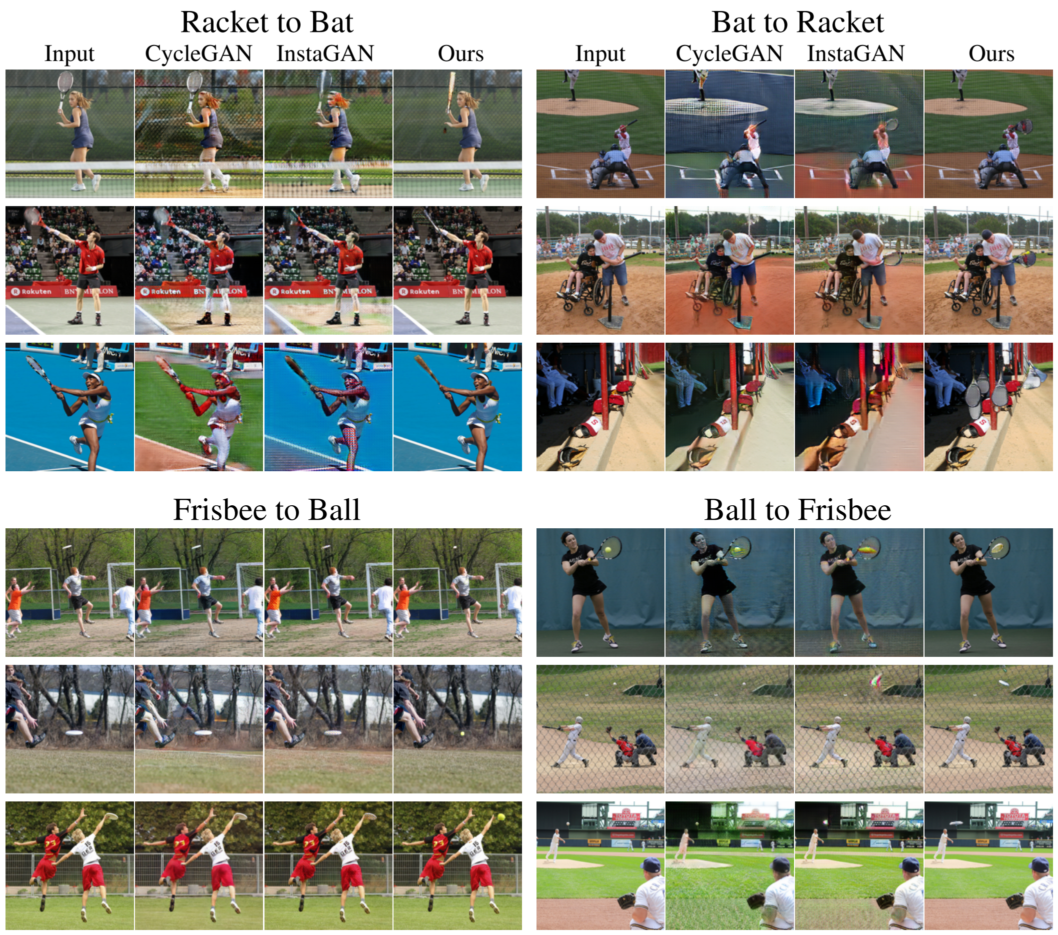

Figure 3: Qualitative comparisons on the Racket→Bat and Ball→Frisbee datasets. The sports dataset is a difficult case in general due to the drastic background differences between grass field sports and clay and hard court sports. Our approach is able to make better use of the object masks in not making unwanted changes to the background and producing comparable-or-better foregrounds.

### 3.5 Limitations

Though we outperform previous baselines, there is significant room for improvement for our method. In 1, our detector fails to detect less than half our generated objects for many of the classes. We hypothesize that the limited receptive field of the convolutional layers in our network cannot cope with the significant shape change required to translate between these unaligned domains.

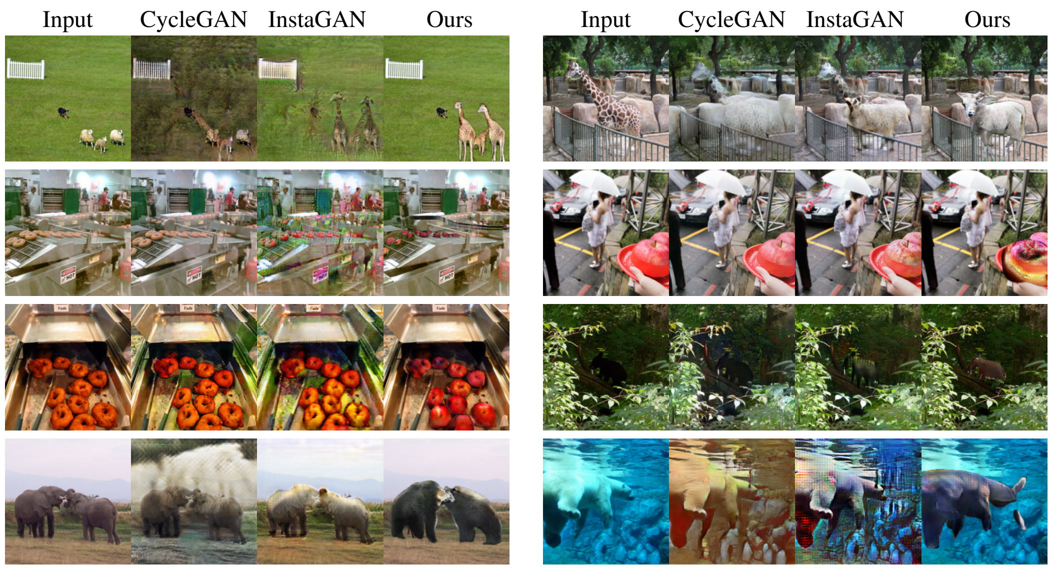

Figure 4: Qualitative comparisons on the Giraffe→Sheep, Donut→Apple, and Elephant→Bear datasets, with their return translations too. Please zoom in. Examples include the challenging translation of a large polar bear in water where the background is atypical within either domain, and many small apple instances. While both our approach and InstaGAN receive input masks, our method is able to make better use of them with unedited backgrounds and comparable or better foregrounds.

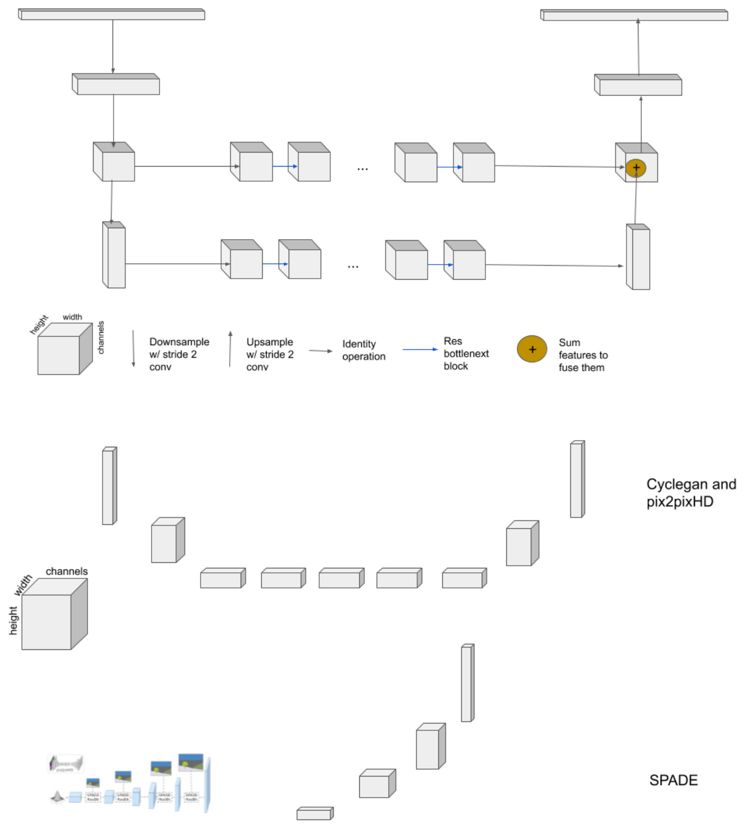

Figure 5: Comparison of our architecture to others. The CycleGAN and Pix2pixHD networks are a special case of ours. SPADE does not use generation branches.

## 4 Feature Translation Pyramid Network

We propose a novel multi-scale architecture that fuses feature maps at multiple spatial resolutions. This improves the stability of our network during training because our network can make global changes to the shape at lower resolution branches while making texture changes at higher resolution branches. Our network is trained end-to-end, meaning it does not require progressive growing [17]. Moreover, each individual generation does not change the spatial resolution of the feature maps. This differs from the approach used in [21], which relies on multiple encoder and decoders (and is also trained progressively). Our network can be considered a single encoder-decoder structure with feature maps at multiple resolutions.

### 4.1 Architecture

As in the SplitGAN section, we seek to translate an object from a source foreground, $S_f$ to a target domain, $T_f$. Our network downsamples the spatial resolution image with a series of strided convolutional layers $n \in N$, and increases the channel resolution with each convolution. The feature map of the $nth$ convolutional layer is translated to the target domain by a series of non-strided convolutions that make up $b_n$. In practice, the $nth$ branch, $b_n$, is comprised of residual blocks, $r_n^i \in R_n$. The output of branch $b_{n+1}$ is fused with the output of $b_n$ by first passing $b_{n+1}$ through a strided transposed convolution layer and then adding it to the output of $b_n$. This is done until we recover the input image's spatial resolution.

The original CycleGAN and Pix2PixHD networks can be considered a special case of our network where $N = 1$. SPADE [30] does not use branches to translate images. Additionally, it does not use strided convolutions to upsample the image. It uses a bilinear interpolation of the feature maps.

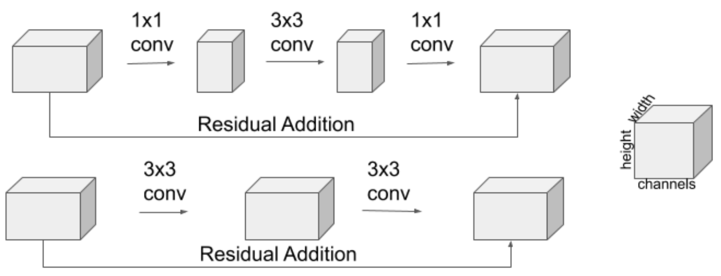

Figure 6: *Top*: The efficient bottlenext residual block our network uses. *Bottom*: The standard residual block design.

## 4.2 Bottlenext Blocks

Our residual blocks, $\{R_i, R_{i+1}...R_n\}$, follow the bottlenext design proposed in [11]. The aforementioned image translation networks use the standard residual block design. The bottlenext design uses a convolutional layer with a $1 \times 1$ kernel to reduce the channel dimension before applying a convolution with a $3 \times 3$ kernel. Following the convolution with the $3 \times 3$ kernel, a $1 \times 1$ kernel convolution is used to recover the original channel dimension. By reducing the channel dimension, with a $1 \times 1$ the network is more efficient. Moreover, we used grouped convolutions to further improve efficiency [42].

## 4.3 Evaluation

**Sheep2Giraffe** We evaluate our network on the sheep to giraffe mapping, using the same COCO dataset from the SplitGAN section. IoU is measured with the Hybrid Task Cascade object detector (implementation from [3]) to assess the quality of generated objects. (This is a different detector than was used in the SplitGAN section so the numbers between the sections are not comparable.) A network with $N = 1$ branches, which is equivalent to the CycleGAN network, has an IoU of 0.62 for giraffes and 0.28 for sheep. Increasing the number of branches to $N = 2$ yields an IoU of 0.64 for giraffes and 0.46 for sheep. Sheep are smaller objects than giraffes, thus the additional higher resolution branch aids with the generation of these smaller objects. However, a network with $N = 3$ mode collapses.

We show qualitative results from our method in Figure 7. We generate these images sequentially as in SplitGAN. However, with our multi-scale network, we translate each object at $384 \times 384$ spatial resolution rather than $200 \times 200$. Though our multi-scale networks improve generation quality, we still suffer many failure cases (Figure 8). We hypothesize that even the multi-scale network does not have a great enough receptive field to make the significant global structural changes between unaligned domain mappings. Thus, we seek a network architecture that does not have this limitation, described in the following section.

**Efficiency** For our multi-scale network, bottlenext blocks reduce parameter count by 8x. We find no change in IoU when returning to using the standard residual blocks. We also test if the benefits of our bottlenext blocks generalize to other architectures. We achieve strong results with Pix2PixHD on the Cityscapes dataset [5]. The network parameters are reduced from 182.9M to 23.5M and GFLOPS are reduced from 749.0 to 314.6. The FID score remains comparable, only increasing from 27 to 31. As shown in Figure 9, the visual quality is comparable.

**Implementation Details** We use the same hyperparameters as in the SplitGAN section. Our models are trained on 2 NVIDIA GTX Titans. We also "clean" the training data by removing instances which are split into multiple connected components and instances which are smaller than 5% of the spatial image resolution or larger than 95% of the spatial image resolution. During preprocessing, we resize all giraffes to be 75% of the generator input image size (in this case $384 \times 384$) and sheep to 37%. They are re-scaled to their appropriate sizes after the generation process. This is so our generator can learn the relative shape changes between sheep and giraffes.

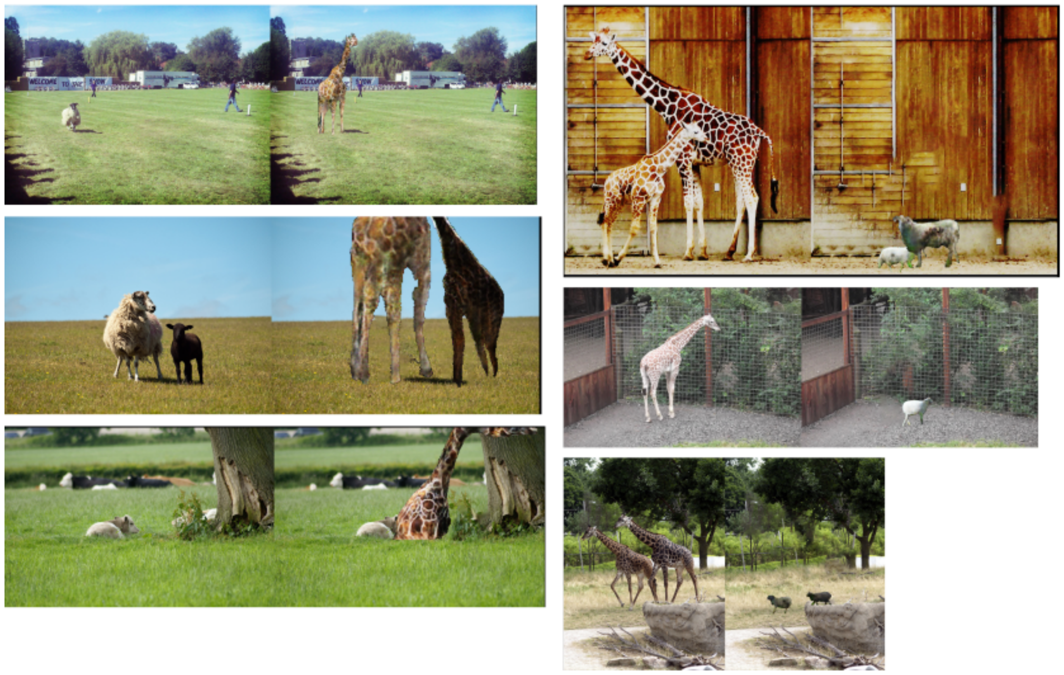

Figure 7: Results of our feature translation pyramid network on the giraffe to sheep and sheep to giraffe mappings.

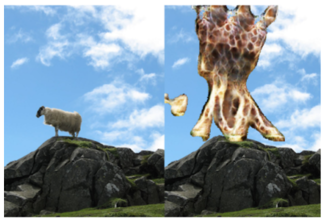

Figure 8: A failure case of our model. We find the network struggles to disentangle the complex shape and color changes required to translate a sheep to a giraffe.

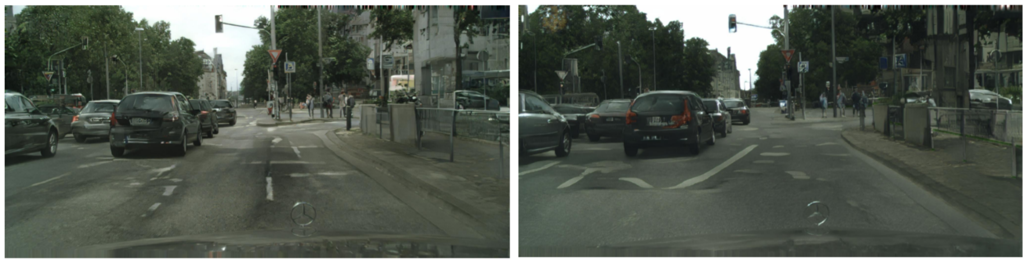

Figure 9: Qualitative of our efficient model (left) compared to the baseline (right).

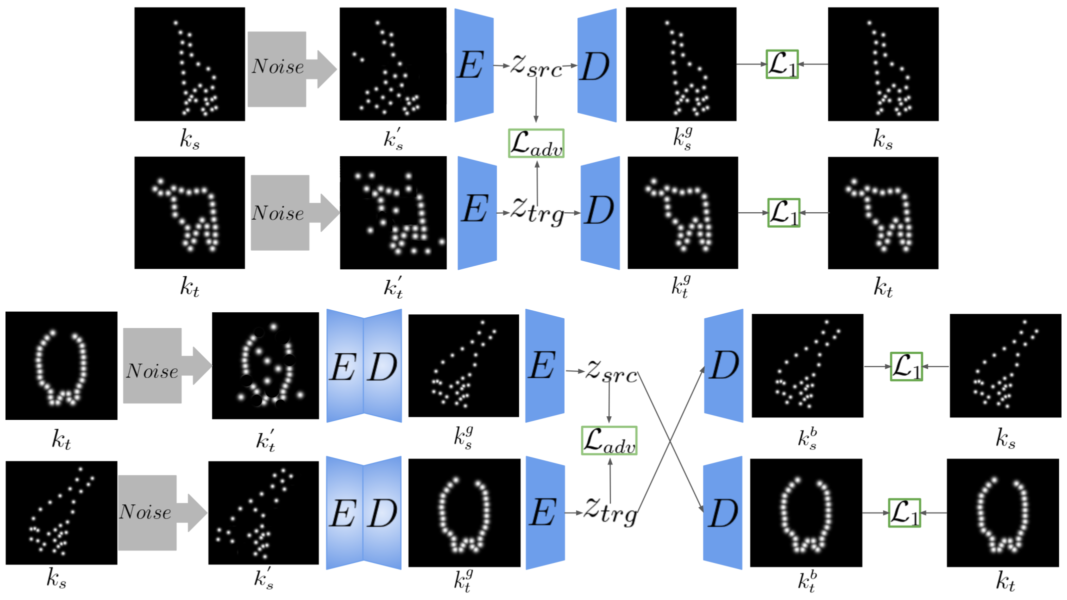

Figure 10: Visualization of our training scheme. Notation corresponds to Section 6.1. Blue weights are trainable and white/blue weights are frozen.

## 5 Translating Object Shapes as Sentences

There are parallels between image and language translation tasks. CycleGAN [47] finds inspiration in language translation models stating: “if we translate, e.g., a sentence from English to French, and then translate it back from French to English, we should arrive back at the original sentence". We take inspiration from the “CycleGAN" of sentence translation [20]. Sentences are translated from a source language to a target language through “back-translation" and “autoencoder" tasks. We decompose object shapes into an outline of $k_i \in \mathcal{K}$ keypoints. We then use a similar training scheme to [20] where a keypoint is a word and the entire shape is a sentence.

### 5.1 Transformer-based Shape Network

We have unpaired samples $s_i \in \mathcal{S}$ from a source domain, which we wish to translate to a target domain, for which we have samples $t_i \in \mathcal{T}$. Let us assume we have a mask for each of these samples. From these masks, we extract keypoints $k_s$ and $k_t$ from the source and target domains, respectively. We train our transformer-based model to encode a sequence of keypoints from either $\mathcal{S}$ or $\mathcal{T}$ to a latent vector $z$. We enforce that, regardless of the domain input, $z$ be aligned for both distributions with an adversarial discriminator tasked with distinguishing which domain the latent vector was encoded from. This vector is then decoded to either the source or target keypoints, depending on the decoder’s binary conditioning. We train our model with the following two tasks:

1. An “autoencoder" task in which random noise is applied to $k$ from either domain, giving us $k^{'}$. $k^{'}$ is passed through our encoder-decoder architecture, giving us $k^g$. The goal is for $k^g$ to be identical to $k$. We train with an $\mathcal{L}_1$ reconstruction loss.
2. A “back-translation" task in which keypoints $k_t$ are passed forward through the frozen weights of our model, which gives us $k_s^g$. We then wish to back-translate $k_s^g$ to the original domain. Thus, after unfreezing our model’s weights, we input $k_s^g$ to our model to obtain the back-translation, $k_t^b$. An $\mathcal{L}_1$ reconstruction loss is the objective function. We do the converse to back-translate from target to source domain.

We visualize our training scheme in Figure 10.

To convert keypoints to a representation for our transformers, we compute the linear projection of their $x, y$ coordinates. We then feed this to an embedding layer. We also embed the id of each

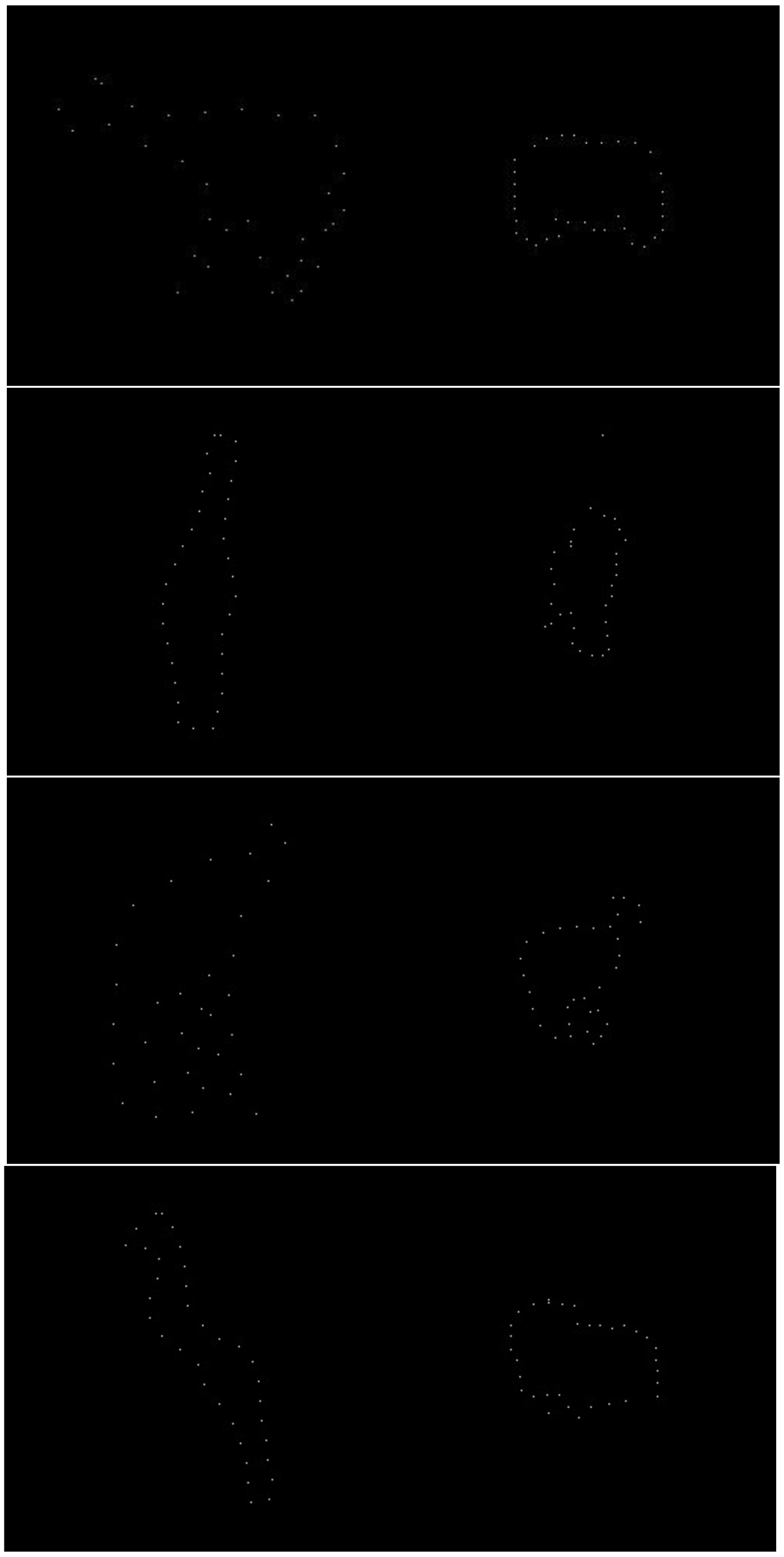
Figure 11: Visualizations of giraffe to sheep shape translations.

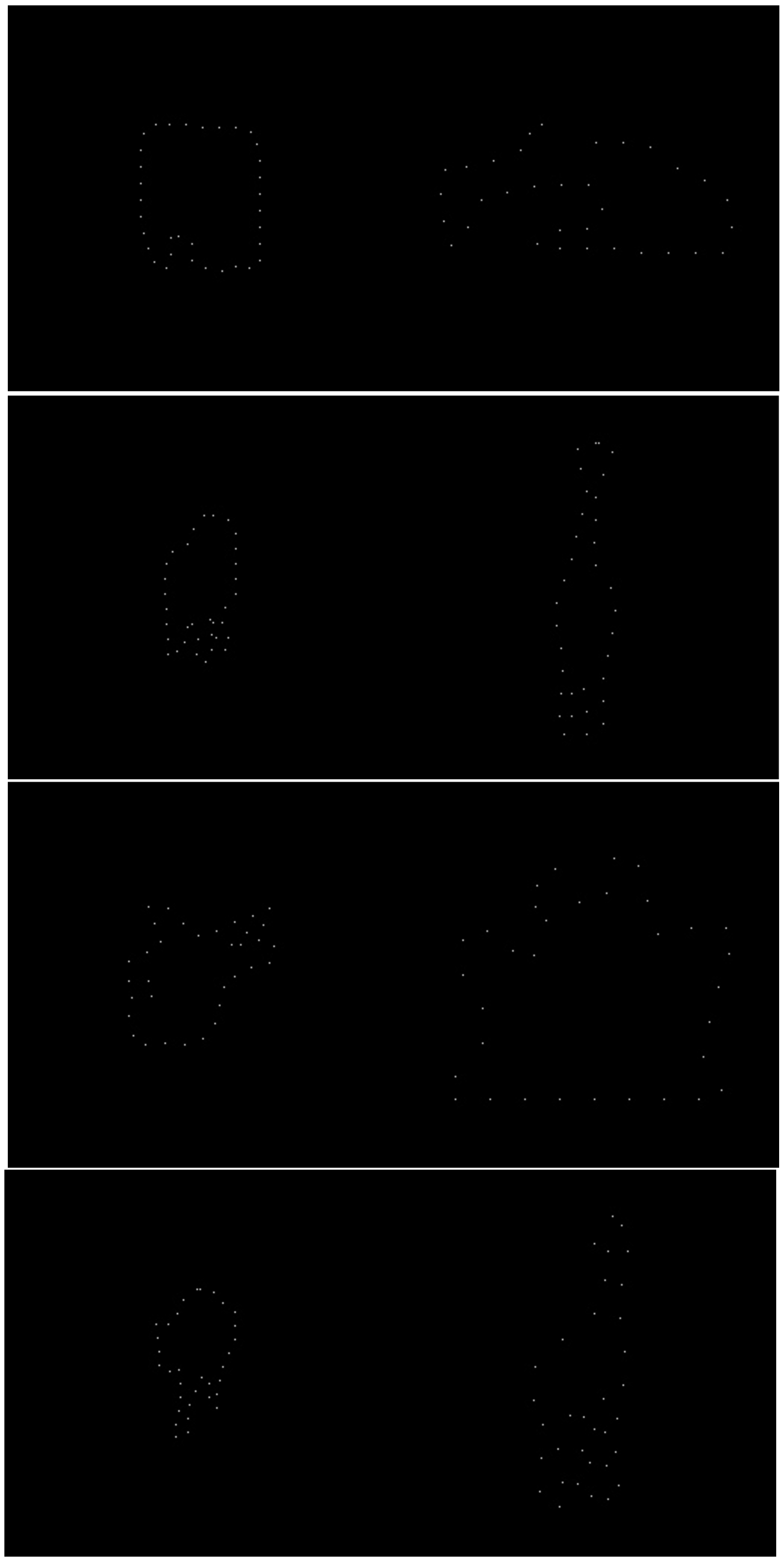

Figure 12: Visualizations of sheep giraffe shape translations.

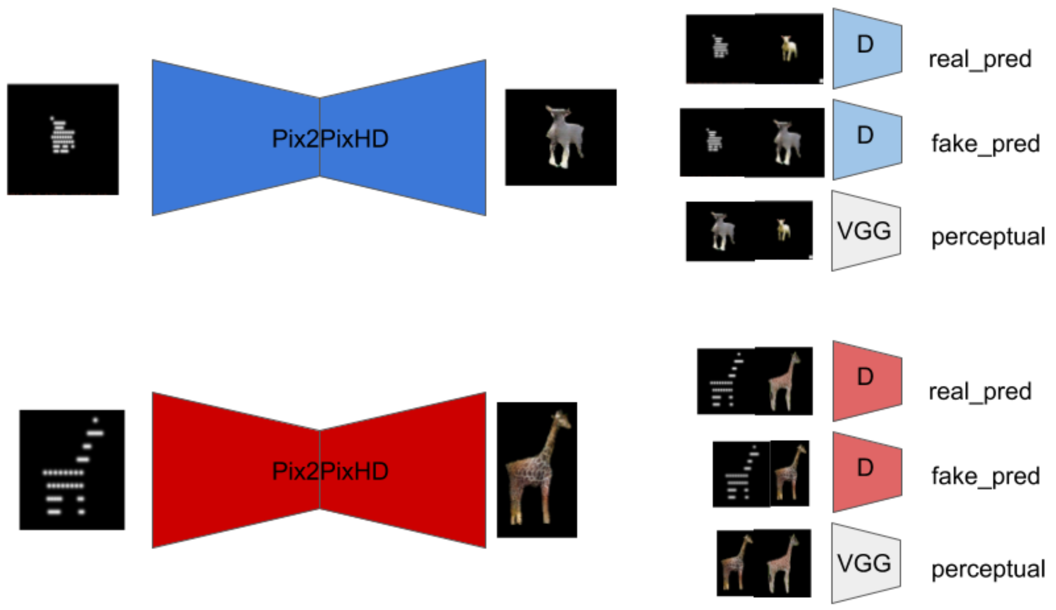

Figure 13: Our color translation model. We use separate encoder-decoder networks for each domain (as indicated by the color codes).

keypoint, where an id of $0$ is used for the keypoint at 12 o'clock. We assign the rest of the ids to keypoints moving in a counterclockwise fashion. The ids are embedded and added to the keypoint position embeddings. The sum of these embeddings is fed to the encoder-decoder model.

**Sheep2Giraffe Keypoint Evaluation** We evaluate our network on the same dataset as in previous sections. 32 keypoints are extracted from each mask. Qualitative results are in Figure 14. We obtain improved shape diversity and detail than with previous methods. Moreover, the network produces quality results on a consistent basis. This suggests potential for our network to overcome the instability of previous methods.

**Implementation Details** All masks are centered. The long edge is scaled to be $75\%$ of the image height and width, which we fix to be $256 \times 256$. The learn rate is $1 \times 10^{-4}$ and we use an inverse square root warmup schedule with the Adam optimizer. Batch size is 12. Models converge on a single NVIDIA Titan GTX GPU within 12 hours.

## 5.2 Exploring Color Translation Methods

Before we combine our shape and color models in an end-to-end training scheme, we evaluate the color task independently from the shape. Inspired by [2], we use Pix2PixHD to color our keypoint representation. A 1 channel mask is fed to the model, which outputs a 3 channel color image. This image is concatenated with the input mask and fed to the discriminator which makes a real / fake prediction. Our training scheme is visualized above.

We find that the outline representation generated by our transformer model does not yield good results when used as input to the color network. Thus, we create more dense representation to feed to our color network. We find these create higher quality color images. Qualitative results are in Figure 14.

We will explore various color translation methods and transformer models to find a keypoint representation that is suitable for use by both the shape and color steps. This will allow our network to train end-to-end and for us take advantage of the stability of separating the shape and color translation steps.

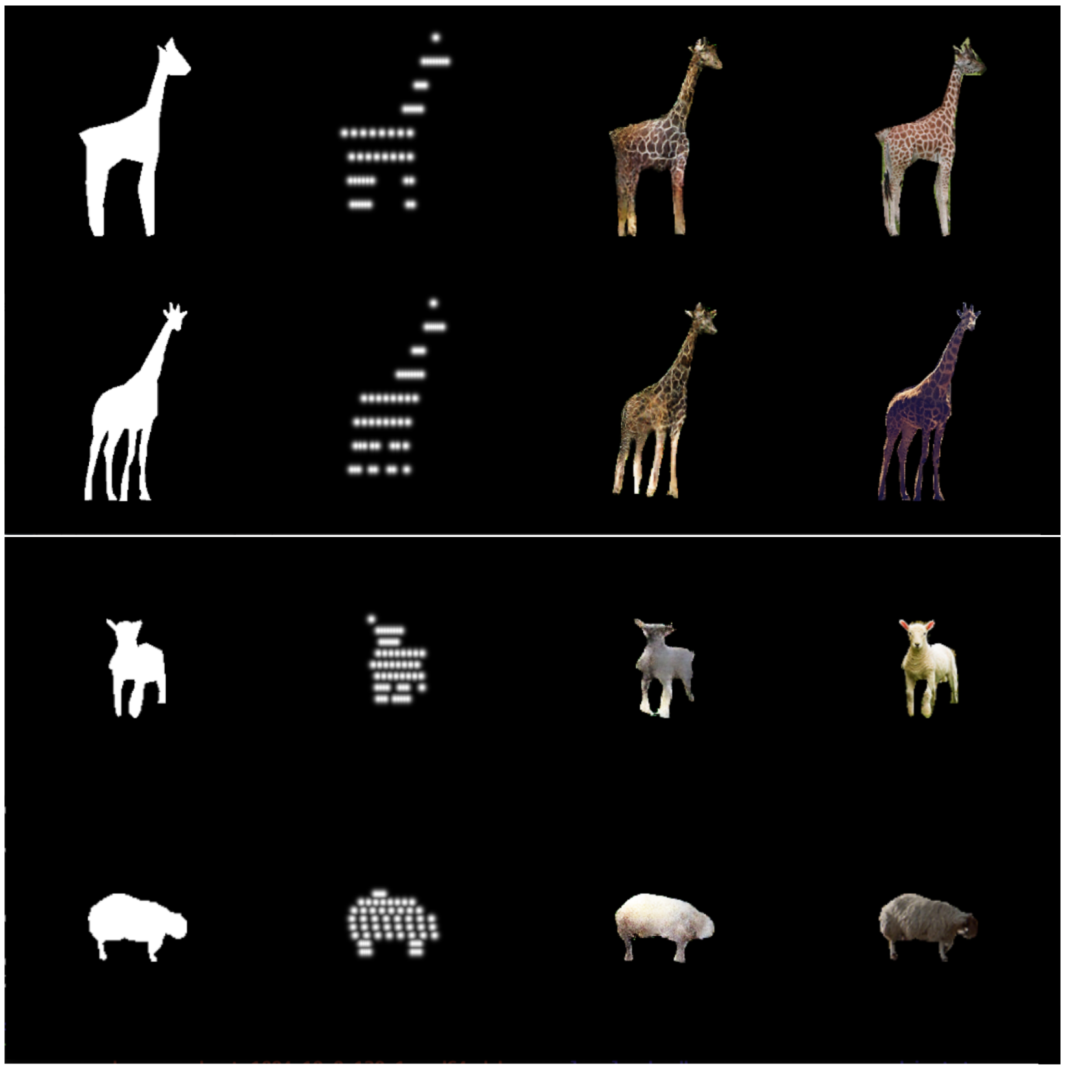

Figure 14: Sheep and giraffe from our color translation model. The 1st column is the mask, the 2nd column is the extracted keypoints, the 3rd column is the generated image, and the 4th column is the ground truth image.

## 6 Conclusion

We have explored various models to translate images between unaligned domains. Our work includes setting a new SOTA by splitting the foreground and background generation process. We further improve over our baseline with multi-scale networks. By using transformer-based network not limited by receptive field, we improve shape generation further. We also make progress in translating these shapes to color images. Additional future work can extend this to the multi-domain setting, common in language translation. Another direction is to add control to the translation process.

## References

[1] Kaidi Cao, Jing Liao, and Lu Yuan. *CariGANs: Unpaired Photo-to-Caricature Translation*. 2018.

[2] Caroline Chan et al. "Everybody Dance Now". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (Oct. 2019). DOI: 10.1109/iccv.2019.00603. URL: http://dx.doi.org/10.1109/ICCV.2019.00603.

[3] Kai Chen et al. "MMDetection: Open MMLab Detection Toolbox and Benchmark". In: *arXiv preprint arXiv:1906.07155* (2019).

[4] Yen-Chun Chen et al. *UNITER: UNiversal Image-TExt Representation Learning*. 2019. arXiv: 1909.11740 [cs.CV].

[5] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016). DOI: 10.1109/cvpr.2016.350. URL: http://dx.doi.org/10.1109/CVPR.2016.350.

[6] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. arXiv: 1810.04805 [cs.CL].

[7] Huan Fu et al. "Geometry-Consistent Adversarial Networks for One-Sided Unsupervised Domain Mapping". In: *NIPS* (2018).

[8] Rohit Girdhar et al. "Video Action Transformer Network". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). DOI: 10.1109/cvpr.2019.00033. URL: http://dx.doi.org/10.1109/CVPR.2019.00033.

[9] Aaron Gokaslan et al. "Improved Shape Deformation in Unsupervised Image to Image Translation". In: *ECCV*. 2018.

[10] I. Goodfellow et al. "Generative adversarial nets". In: *NIPS*. 2014.

[11] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016). DOI: 10.1109/cvpr.2016.90. URL: http://dx.doi.org/10.1109/cvpr.2016.90.

[12] Jonathan Huang et al. "Speed/accuracy trade-offs for modern convolutional object detectors". In: *CVPR*. 2017.

[13] X. Huang et al. "Multimodal Unsupervised Image-to-Image Translation". In: *ECCV*. 2018.

[14] Xun Huang et al. "Multimodal Unsupervised Image-to-image Translation". In: *ECCV*. 2018.

[15] Tomas Jakab et al. *Unsupervised Learning of Object Landmarks through Conditional Image Generation*. 2018. arXiv: 1806.07823 [cs.CV].

[16] Animesh Karnewar and Oliver Wang. *MSG-GAN: Multi-Scale Gradient GAN for Stable Image Synthesis*. 2019. arXiv: 1903.06048 [cs.CV].

[17] Tero Karras et al. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. 2017. arXiv: 1710.10196 [cs.NE].

[18] Junho Kim et al. *U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation*. 2019. arXiv: 1907.10830 [cs.CV].

[19] T. Kim et al. "Learning to discover cross-domain relations with generative adversarial networks". In: *JMLR* (2017).

[20] Guillaume Lample et al. *Unsupervised Machine Translation Using Monolingual Corpora Only*. 2017. arXiv: 1711.00043 [cs.CL].

[21] Minjun Li et al. "Unsupervised Image-to-Image Translation with Stacked Cycle-Consistent Adversarial Networks". In: *Lecture Notes in Computer Science* (2018), pp. 186–201. ISSN: 1611-3349. DOI: 10.1007/978-3-030-01240-3_12. URL: http://dx.doi.org/10.1007/978-3-030-01240-3_12.

[22] Xiaodan Liang, Hao Zhang, and Eric P. Xing. "Generative Semantic Manipulation with Contrasting GAN". In: *ECCV*. 2017.

[23] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). DOI: 10.1109/cvpr.2017.106. URL: http://dx.doi.org/10.1109/CVPR.2017.106.

[24] Tsung-Yi Lin et al. "Microsoft CoCo: Common objects in context". In: *ECCV*. 2014.

[25] Jiasen Lu et al. *ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks*. 2019. arXiv: 1908.02265 [cs.CV].

[26] M. Sangwoo, M. cho, and j. Shin. “Instance-aware Image-to-Image Translation”. In: *ICLR*. 2019.

[27] S. Ma et al. “DA-GAN: Instance-level Image Translation by Deep Attention Generative Adversarial Networks”. In: *CVPR*. 2018.

[28] Youssef A Mejjati et al. “Unsupervised Attention-guided Image to Image Translation”. In: *NIPS*. 2018.

[29] Alejandro Newell, Kaiyu Yang, and Jia Deng. “Stacked Hourglass Networks for Human Pose Estimation”. In: *Lecture Notes in Computer Science* (2016), pp. 483–499. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46484-8_29. URL: http://dx.doi.org/10.1007/978-3-319-46484-8_29.

[30] Taesung Park et al. “Semantic Image Synthesis With Spatially-Adaptive Normalization”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). DOI: 10.1109/cvpr.2019.00244. URL: http://dx.doi.org/10.1109/CVPR.2019.00244.

[31] Niki Parmar et al. *Image Transformer*. 2018. arXiv: 1802.05751 [cs.CV].

[32] S. Benaim and L. Wolf. “One-sided unsupervised domain mapping”. In: *NIPS*. 2017.

[33] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. “SinGAN: Learning a Generative Model From a Single Natural Image”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (Oct. 2019). DOI: 10.1109/iccv.2019.00467. URL: http://dx.doi.org/10.1109/ICCV.2019.00467.

[34] Michael Snower et al. *15 Keypoints Is All You Need*. 2019. arXiv: 1912.02323 [cs.CV].

[35] Chen Sun et al. “VideoBERT: A Joint Model for Video and Language Representation Learning”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (Oct. 2019). DOI: 10.1109/iccv.2019.00756. URL: http://dx.doi.org/10.1109/ICCV.2019.00756.

[36] C. Szegedy et al. “Rethinking the Inception architecture for computer vision”. In: *CVPR*. 2016.

[37] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[38] T.-C. Wang et al. “High-resolution image synthesis and semantic manipulation with conditional GANs”. In: *CVPR*. 2018.

[39] Ting-Chun Wang et al. “Video-to-Video Synthesis”. In: *NIPS*. 2018.

[40] Wayne Wu et al. “TransGaGa: Geometry-Aware Unsupervised Image-To-Image Translation”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). DOI: 10.1109/cvpr.2019.00820. URL: http://dx.doi.org/10.1109/CVPR.2019.00820.

[41] X. Chen et al. “Attention-GAN for Object Transfiguration in Wild Images”. In: *ECCV*. 2018.

[42] Saining Xie et al. “Aggregated Residual Transformations for Deep Neural Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). DOI: 10.1109/cvpr.2017.634. URL: http://dx.doi.org/10.1109/CVPR.2017.634.

[43] Chao Yang et al. “Show, Attend and Translate: Unsupervised Image Translation with Self-Regularization and Attention”. In: *arXiv preprint arXiv:1806.06195* (2018).

[44] Jianwei Yang et al. “LR-GAN: Layered recursive generative adversarial networks for image generation”. In: *ICLR*. 2017.

[45] Jiahui Yu et al. “Free-Form Image Inpainting with Gated Convolution”. In: *arXiv preprint arXiv:1806.03589* (2018).

[46] Jiahui Yu et al. “Generative Image Inpainting with Contextual Attention”. In: *arXiv preprint arXiv:1801.07892* (2018).

[47] J. Zhu et al. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *ICCV*. 2017.