Cross-View Image Synthesis using Conditional GANs

Krishna Regmi and Ali Borji
Center for Research in Computer Vision, University of Central Florida

krishna.regmi7@gmail.com, aborji@crcv.ucf.edu

Abstract

Learning to generate natural scenes has always been a challenging task in computer vision. It is even more painstaking when the generation is conditioned on images with drastically different views. This is mainly because understanding, corresponding, and transforming appearance and semantic information across the views is not trivial. In this paper, we attempt to solve the novel problem of cross-view image synthesis, aerial to street-view and vice versa, using conditional generative adversarial networks (cGAN). Two new architectures called Crossview Fork (X-Fork) and Crossview Sequential (X-Seq) are proposed to generate scenes with resolutions of 64 × 64 and 256 × 256 pixels. X-Fork architecture has a single discriminator and a single generator. The generator hallucinates both the image and its semantic segmentation in the target view. X-Seq architecture utilizes two cGANs. The first one generates the target image which is subsequently fed to the second cGAN for generating its corresponding semantic segmentation map. The feedback from the second cGAN helps the first cGAN generate sharper images. Both of our proposed architectures learn to generate natural images as well as their semantic segmentation maps. The proposed methods show that they are able to capture and maintain the true semantics of objects in source and target views better than the traditional image-to-image translation method which considers only the visual appearance of the scene. Extensive qualitative and quantitative evaluations support the effectiveness of our frameworks, compared to two state of the art methods, for natural scene generation across drastically different views.

1. Introduction

In this work, we address the problem of synthesizing ground-level images from overhead imagery and vice versa using conditional Generative Adversarial Networks [20]. Primarily, such models try to generate new images from conditioning variables as input. Preliminary works in GANs utilize unsupervised learning to generate samples from latent representations or from a random noise vector [9].

View synthesis is a long-standing problem in computer vision. This task is more challenging when views are drastically different, fields of views have little or no overlap, and objects are occluded. Furthermore, two objects that are similar in one view may look quite different in another (i.e., the view-invariance problem). For example, the aerial view of a building (i.e., the roof) tells very little about the color and design of the building seen from the street-view. The generation process is generally easier when the image contains a single object in a uniform background. In contrast, when the scene contains multiple objects, generating other view becomes much more challenging. This is due to the increase in underlying parameters that contribute to the variations (e.g., occlusions, shadows, etc.). An example scenario, addressed here, is generating street-view (a.k.a ground level) image of a location from its aerial (a.k.a overhead) imagery. Figure 1 illustrates some corresponding images in two views.

Isola et al. [12] put forward a general-purpose framework to solve multiple image translation tasks. Their work translates images of objects or scenes which are represented by RGB images, gradient fields, edge maps, aerial images, sketches, etcetera between these representations. Thus, their method operates on representations in a single view. Formulating our problem as an image translation task between the views, we use their method as a baseline and extend it for cross-view image generation.

Inspired by recent works of [12, 35], we formulate the cross-view image synthesis problem as an image-to-image translation problem and solve it using the conditional generative adversarial network. Previous works in view synthesis [8, 38, 31] have generated images with single objects in
them or natural scenes with very little variation in viewing angles between the input and target images. The network learns to copy large parts of image content from the input. Images in each view of our work contain high degree of details and clutter (e.g., trees, cars, roads, buildings, etc.) along with variations between the corresponding image pairs. Thus, the network needs to learn how the corresponding images in each view need to contain all details and place them in correct positions with proper orientations and inclinations. Perhaps the closest work to ours is the one by Zhai et al. [35]. They generate ground-level panorama from aerial imagery by predicting ground image features from aerial image features and use them to synthesize images.

In general, some of the challenges pertaining to crossview synthesis task are as follows. First, aerial images cover wider regions of the ground than the street-view images whereas street-view images contain more details about objects (e.g., house, road, trees) than aerial images. So, not only the information in aerial images is too noisy, but also less informative for street-view image synthesis. Similarly, a network needs to estimate a lot regions to synthesize aerial images. Second, transient objects like cars (also people) are not present at the corresponding locations in image pairs since they are taken at different times. Third, houses that are different in street-view look similar from aerial view. This causes synthesized street-view images to contain buildings with similar color or texture, prohibiting diversity in generated buildings. Fourth challenge regards variation among roads in two views due to perspective and occlusions. While the road edges are nearly linear and visible in street-view, they are often occluded by dense vegetation and contorted in aerial view. Fifth, when using model generated segmentation maps as ground truth to improve the quality of generated images, as done here, label noise and model errors introduce some artifacts in the results.

To address the above challenges, we propose the following methods. We start with a simple image-to-image translation network of [12] as a baseline. We then propose two new cGAN architectures that generate images as well as segmentation maps in target view. Addition of semantic segmentation generation to the architectures helps improve the generation of images. The first architecture, called X-Fork, is a slight modification of the baseline, forking at the penultimate block to generate two outputs, target view image and segmentation map. The second architecture, called X-Seq, has a sequence of two baseline networks connected. The target view image generated by the first network is fed to the second network to generate its corresponding segmentation map. Once trained, both architectures are able to generate better images than the baseline that learns to generate the images only. This implies that learning to generate segmentation map along with the image indeed improves the quality of generated image.

2. Related Works
2.1. Relating Aerial and Ground-level Images
Zhai et al. [35] explored to predict the semantic layout of ground image from its corresponding aerial image. They used the predicted layout to synthesize ground-level panorama. Prior works relating the aerial and ground imageries have addressed problems such as cross-view co-localization [18, 33], ground-to-aerial geo-localization [17] and geo-tagging the cross-view images [34].

Cross-view relations have also been studied between egocentric (first person) and exocentric (surveillance or third-person) domains for different purposes. Human re-identification by matching viewers in top-view and egocentric cameras have been tackled by establishing the correspondences between the views in [1]. Soran et al. [29] utilize the information from one egocentric camera and multiple exocentric cameras to solve the action recognition task. Ardeshir et al. [2] learn motion features of actions performed in ego- and exocentric domains to transfer motion information across the two domains.

2.2. Learning View Transformations
Existing works on viewpoint transformation have been conducted to synthesize novel views of the same objects [8, 31, 38]. Zhou et al. [38] proposed models that learn to copy the pixel information from input view and utilize them to preserve the identity and structure of the objects to generate new views. Tatarenko et al. [31] trained an encoder-decoder network to obtain 3D representation models of cars and chairs which they later used to generate different views of an unseen car or chair image. Dosovitskiy et al. [8] learned generative models by training on 3D renderings of cars, chairs and tables and synthesized intermediate views and objects by interpolating between views and models.

2.3. GAN and cGAN
Goodfellow et al. [9] are the pioneers of Generative Adversarial Networks that is very successful at generating sharp and unblurred images, much better compared to existing methods such as Restricted Boltzmann Machines [10, 28] or deep Boltzmann Machines [25].

Conditional GANs are used to synthesize images conditioned on different parameters during both training and testing. Examples include conditioning on labels of MNIST to generate digits by Mirza et al. [20], conditioning on image representations to translate an image between different representations [12], and generating panoramic ground-level scenes from aerial images of the same location [35]. Pathak et al. [23] generated missing parts in images (i.e., inpainting) using networks trained jointly with adversarial and reconstruction losses and produced sharp and coherent images. Reed et al. [24] synthesized images conditioned on detailed textual descriptions of the objects in the scene, and Zhang et al. [36] improved on that by using a two-stage Stacked GAN.
2.4. Cross-Domain Transformations using GANs

Kim et al. [13] utilized the GAN networks to learn the relationship between images in two different domains such that these learned relations can be transferred between the domains. Similar work by Zhu et al. [39] learned mappings between unpaired images using cycle-consistency loss. They assume that accumulating from one domain to the other and back to the first should generate the original image. Both works exploited large unpaired datasets to learn the relation between domains and formulated the mapping task between images in different domains as a generation problem. Zhu et al. [39] compare their generation task with previous works on paired datasets by Isola et al. [12]. They conclude that the results with paired images is the upper-bound for their unpaired examples.

3. Background on GANs

Generative Adversarial Network architecture [9] consists of two adversarial networks: a generator and a discriminator that are trained simultaneously based on the min-max game theory. The generator \( G \) is optimized to map a \( d \)-dimensional noise vector (usually \( d=100 \)) to an image (i.e., synthesizing) that is close to the true data distribution. The discriminator \( D \), on the other hand, is optimized to accurately distinguish between the synthesized images coming from the generator and the real images from the true data distribution. The objective function of such a network is

\[
\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}(x)} \log D(x) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D(G(z)))
\]

(1)

where, \( x \) is real data sampled from data distribution \( p_{\text{data}} \) and \( z \) is a \( d \)-dimensional noise vector sampled from a Gaussian distribution \( p_z \).

Conditional GANs synthesize images looking into some auxiliary variable which may be labels [20], text embeddings [36, 24] or images [12, 39, 13]. In conditional GANs, both the discriminator and the generator networks receive the conditioning variable represented by \( c \) in Eqn. (2). The generator uses this additional information during image synthesis while the discriminator makes its decision by looking at the pair of conditioning variable and the image it receives. Real pair input to the discriminator consists of true image from distribution and its corresponding label while the fake pair consists of synthesized image and the label. For conditional GAN, the objective function is

\[
\min_G \max_D \mathbb{E}_{x, c \sim p_{\text{data}}(x, c)} \log D(x, c) + \mathbb{E}_{x', c \sim p_{\text{data}}(x', c)} \log(1 - D(x', c))
\]

(2)

where \( x' = G(z, c) \) is the generated image.

In addition to the GAN loss, previous works (e.g., [12, 39, 23]) have tried to minimize the \( L1 \) or \( L2 \) distances between real and generated image pairs. This step aids the generator to synthesize images very similar to the ground truth. Minimizing \( L1 \) distance generates less blurred images than minimizing the \( L2 \) distance. That is, using the \( L1 \) distance increases image sharpness in generation tasks. Therefore, we use the \( L1 \) distance in our method. The expression to minimize the \( L1 \) distance is

\[
\min_G \mathbb{E}_{x, x' \sim p_{\text{data}}(x', x')} \| x - x' \|_1,
\]

(3)

The objective function for such conditional GAN network is the sum of Eqns. (2) and (3).

Considering the synthesis of the ground level imagery \((I_g)\) from aerial image \((I_a)\), the conditional GAN loss and \( L1 \) loss are represented as in Eqns. (4) and (5), respectively. For ground to aerial synthesis, the roles of \( I_a \) and \( I_g \) are reversed.

\[
\min_G \max_D \mathbb{E}_{I_a, I_g \sim p_{\text{data}}(I_a, I_g)} \log D(I_g, I_a) + \mathbb{E}_{I_a, I_g \sim p_{\text{data}}(I_a, I_g)} \log(1 - D(I_a, I_g)),
\]

(4)

\[
\min_G \mathbb{E}_{I_g \sim p_{\text{data}}(I_g)} \| I_g - I'_g \|_1,
\]

(5)

where, \( I'_g = G(I_a) \). We employ the network of [12] as our baseline architecture. The objective function for the baseline is the sum of conditional GAN loss in Eqn. (4) and \( L1 \) loss in Eqn. (5), as represented in Eqn. (6):

\[
L_{\text{network}} = \mathbb{E}_{I_a, I_g \sim p_{\text{data}}(I_a, I_g)} \log D(I_g, I_a) + \lambda \mathbb{E}_{I_g \sim p_{\text{data}}(I_g)} \| I_g - I'_g \|_1,
\]

(6)

where, \( \lambda \) is the balancing factor between the losses.

4. Proposed cGAN-based Approaches

In this section, we propose two architectures for the task of cross-view image synthesis.

4.1. Crossview Fork (X-Fork)

Our first architecture, known as Crossview Fork, is shown in Figure 2a. The discriminator architecture is taken from the baseline [12] but the generator network is forked to synthesize images as well as segmentation maps. The fork-generator architecture is shown in Figure 3. The first six blocks of decoder share the weights. This is because the image and segmentation map contain a lot of shared features. The number of kernels used in each layer (block) of the generator are shown below the blocks.

Even though the X-Fork architecture generates the cross-view image and its segmentation map, the discriminator receives only the real/fake image pairs but not the segmentation pairs during the training. In other words, the generated segmentation map serves as an auxiliary output. Notice that here we are primarily interested in generating higher quality images rather than the segmentation maps. Thus, the conditional GAN loss for this network is still the same as in Eqn. (4). To use the segmentation information, in addition to the \( L1 \) distance between the generated image and the real image, we also include the \( L1 \) distance between the ground-truth segmentation and the generated segmentation map into the loss.
4.2. Crossview Sequential (X-Seq)

Our second architecture uses a sequence of two cGAN networks as shown in Figure 2b. The first network generates cross-view images similar to the baseline. The second network receives images from the first generator as conditioning input to synthesize the segmentation map in the same view. Thus, the first network is a cross-view cGAN while the second one is an image-to-segmentation cGAN. The whole architecture is trained end-to-end so that both cGANs learn simultaneously. Intuitively, the input-output dependency between the cGANs constrains the generated images and the segmentation maps, and in effect improves the quality of the generated outputs. Training the first network to generate better cross-view images enhances generation of better segmentation maps by the second generator. At the same time, the feedback from the better trained second network forces the first network to improve its generation. Thus, when both networks are trained in tandem, better quality images are generated compared to the baseline.

Replacing $G$ and $D$ in Eqns. (4) and (5) by $G_1$ and $D_1$, respectively, we obtain the equivalent expressions for losses of cross-view cGAN network in this architecture. For the image-to-segmentation cGAN network, the images generated by $G_1$ are considered as conditioning inputs. We now express the cGAN loss for this network as

$$
\min_{G_2} \max_{D_2} L_{GAN}(G_2, D_2) = E_{S_g \sim p_{data}(S_g)} [\log D_2(S_g, I'_g)] + E_{S'_g \sim p_{data}(S_g)} [\log(1 - D_2(S'_g, I'_g))].
$$

where, $I'_g = G_1(I_g)$ and $S'_g = G_2(I'_g)$. The $L1$ loss for the image-to-segmentation network is

$$
\min_{G_2} L_{L1}(G_2) = E_{S_g, S'_g \sim p_{data}(S_g, S'_g)} || S_g - S'_g ||_1.
$$

The overall objective function for the X-Seq network is

$$
L_{X-Seq} = L_{GAN}(G_1, D_1) + \lambda L_{L1}(G_1) + L_{GAN}(G_2, D_2) + \lambda L_{L1}(G_2).
$$

Eqn. (9) is optimized during the training to learn the parameters $G_1$, $D_1$, $G_2$ and $D_2$.

5. Experimental Setting

5.1. Dataset

For the experiments in this work, we use the cross-view image dataset provided by Vo et al. [33]. This dataset consists of more than one million pairs of street-view and overhead view images collected from 11 different cities in the US. We select 76,048 image pairs from Dayton and create a train/test split of 55,000/21,048 pairs. We call it Dayton Dataset. The images in the original dataset have resolution of 354×354. We resize them to 256×256. Some example images are shown in Figure 1.

We also recruit the CVUSA dataset [34] for direct comparison of our work with Zhai et al. [35]. This dataset consists of 35,532/8,884 train/test split of image pairs. Following Zhai et al., the aerial images are center-cropped to 224×224 and then resized to 256×256. We only generate a single camera-angle image rather than the panorama. To do so, we take the first quarter of the ground level images and segmentations from the dataset and resize them to 256×256 in our experiments. Please see Figure 7 for some images from the CVUSA dataset.

The two networks, X-Fork and X-Seq, learn to generate the target view images and segmentation maps conditioned on source view image. Training procedure requires the images as well as their semantic segmentation maps.
The CVUSA dataset has annotated segmentation maps for ground view images, but for Dayton dataset such information is not available. To compensate, we use one of the leading semantic segmentation methods, known as the RefineNet [16]. This network is pre-trained on outdoor scenes of the Cityscapes dataset [6] and is used to generate the segmentation maps that are utilized as ground truth maps. These semantic maps have pixel labels from 20 classes (e.g., road, sidewalk, building, vegetation, sky, void, etc). Figure 4 shows image pairs from the dataset and their segmentation masks overlaid in both views. As can be seen, the segmentation mask (label) generation process is far from perfect since it is unable to segment parts of buildings, roads, cars, etcetera in images.

5.2. Implementation Details

We use the conditional GAN architecture of [12] as the baseline and call it Pix2pix. The generator is an encoder-decoder network with blocks of Convolution, Batch Normalization [11] and activation layers. Leaky ReLU with a slope of 0.2 is used as the activation function in the encoder, whereas the decoder has ReLU activation except for its final layer where Tanh is used. The first three blocks of the decoder have a Dropout layer in between Batch normalization and activation layer, with dropout rate of 50%. The discriminator is similar to the encoder of the generator. The only difference is that the final layer uses sigmoid non-linearity that gives the probability of its input being real.

The used convolutional kernels are 4×4 with a stride of 2. The upconvolution in the decoder is Torch[5] implementation of SpatialFullConvolution, and upsamples the input by 2. For the encoder and the discriminator, convolutional operation downsamples the images by 2. No pooling operation is used in the networks. The $\lambda$ used in Eqs. (6) and (9) is the balancing factor between the GAN loss and $L_1$ loss. Its value is fixed at 100. Following the idea to smooth the labels by [30] and demonstration of its effectiveness by Salimans et al. [26], we use one-sided label smoothing to stabilize the training process, replacing 1 by 0.9 for real labels. During the training, we utilized different data augmentation methods like random jitter and horizontal flipping of images. The network is trained end-to-end with weights initialized with a random Gaussian distribution with zero mean and 0.02 standard deviation. It is implemented in Torch [5].

6. Results

Our experiments are conducted in $a2g$ (aerial-to-ground) and $g2a$ (ground-to-aerial) directions on Dayton dataset and $a2g$ direction only on CVUSA dataset. We consider image resolutions of $64 \times 64$ and $256 \times 256$ on Dayton dataset while for experiments on CVUSA dataset, $256 \times 256$ resolution images are used.

First, we run experiments on lower resolution images ($64 \times 64$) for proof of concept. Encouraging qualitative and quantitative results in this resolution motivated us to apply our methods to higher resolution ($256 \times 256$) images. The lower resolution experiments are carried out for 100 epochs with batch size of 16, whereas the higher resolution experiments are conducted for 35 epochs with batch size of 4.

We conduct experiments on CVUSA dataset for comparison with Zhai et al.’s work [35]. Following their setup, we train our architectures for 30 epochs, using the Adam optimizer and moment parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

It is not straightforward to evaluate the quality of synthesized images [3]. In fact, evaluation of GAN methods continues to be an open problem [32]. A common evaluation method is to show the generated images to human observers and ask their opinion about the images. Human judgment is based on the response to the question: Is this image (image-pair) real or fake? Alternatively, the images generated by different generative models can be pitted against each other and the observer is asked to select the image that looks more real. But in experiments involving natural scenes, such evaluation methods are more challenging as multiple factors often affect the quality of the generated images. For example, the observer may not be sure whether to base his judgment on better visual quality, higher sharpness at object boundaries, or more semantic information present in the image (e.g., multiple objects in the images, more details on objects, etc). Therefore, instead of behavioral experiments, we illustrate qualitative results in Figures 5, 6 and 7 and conduct an in-depth quantitative evaluation on test images of two datasets.

6.1. Qualitative Evaluation

For $64 \times 64$ resolution experiments, the networks are modified by removing the last two blocks of CBL from discriminator and encoder, and the first two blocks of UBDR from decoder of the generator. We run experiments on all three methods. Qualitative results are depicted in Figure 5. The results affirm that the networks have learned to transfer the image representations across the views. Generated ground level images clearly show details about road, trees, sky, clouds, and pedestrian lanes. Trees, grass, road, house roofs are well rendered in the synthesized aerial images.

For $256 \times 256$ resolution synthesis, we conduct experiments on all three architectures and illustrate the qualitative results on Dayton and CVUSA datasets in Figures 6 and 7.
respectively. For Dayton dataset, we observe that the images generated in higher resolution contain more details of objects in both views and are less granulated than those in lower resolution. Houses, trees, pedestrian lanes, and roads look more natural. Test results on CVUSA dataset show that images generated by proposed methods are visually better compared to Zhai et al. [35] and Pix2pix [12] methods.

6.2. Quantitative Evaluation

The quantitative results of our experiments on both datasets are presented in Tables 1-4. 64×64 and 256×256 in column headers of the tables refer to resolutions of Dayton dataset. Next, we discuss the quantitative measures used to evaluate our methods.

6.2.1 Inception Score

A common quantitative GAN evaluation measure is the Inception Score [26]. The core idea behind the inception score is to assess how diverse the generated samples are within a class while being meaningfully representative of the class at the same time.

\[ \text{Inception Score, } I = \exp \left( E_x D_{KL}(p(y|x) \| p(y)) \right) \text{,} \tag{10} \]

where, \( x \) is a generated sample and \( y \) is its predicted label.
works. A quantitative analysis of our work, similar to some generative works, shows that our methods generate much better results than existing generative methods on both datasets. X-Fork generates images very similar to real distribution in all experiments except on the higher resolution a2g experiment where X-Seq is slightly better than X-Fork.

### 6.2.2 KL(model ∥ data)

We next compute the KL divergence between the model generated images and the real data distribution for quantitative analysis of our work, similar to some generative works [4, 21]. We again use the same pre-trained Alexnet as in the previous subsection. The lower KL score implies that the generated samples are closer to the real data distribution. The scores are provided in Table 3. As it can be seen, our proposed methods generate much better results than existing generative methods on both datasets. X-Fork generates images very similar to real distribution in all experiments except on the higher resolution a2g experiment where X-Seq is slightly better than X-Fork.

### 6.2.3 SSIM, PSNR and Sharpness Difference

As in some generative works [19, 15, 27, 22], we also employ Structural-Similarity (SSIM), Peak Signal-to-Noise Ratio (PSNR) and Sharpness Difference measures to evaluate our methods.

SSIM measures the similarity between the images based on their luminance, contrast and structural aspects. SSIM value ranges between -1 and +1. A higher value means greater similarity between the images being compared. It is computed as

$$SSIM(I_g, I_g') = \frac{\mu I_g \mu I_g' + c_1 (\sigma I_g^2 + \sigma I_g'^2 + c_2)}{\mu I_g^2 + \mu I_g'^2 + c_1 (\sigma I_g^2 + \sigma I_g'^2 + c_2)}$$  

(11)

PSNR measures the peak signal-to-noise ratio between two images to assess the quality of a transformed (generated) image compared to its original version. The higher the PSNR, the better is the quality of generated image. It is computed as

$$PSNR(I_g, I_g') = 10 \log_{10} \left( \frac{\text{max}_{I_g}}{\text{mse}} \right)$$  

(12)

where, \(mse(I_g, I_g') = \frac{1}{n} \sum_{i=1}^{n} (I_g[i] - I_g'[i])^2\), and \(\text{max}_{I_g} = 255\) (maximum pixel intensity value).

---

### Table 1: KL divergence scores between conditional and marginal probabilities (Inception Score).

<table>
<thead>
<tr>
<th>Dir.</th>
<th>Methods</th>
<th>64 × 64</th>
<th>256 × 256</th>
<th>CVUSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Top-1</td>
<td>Top-5</td>
<td>Top-1</td>
</tr>
<tr>
<td></td>
<td>all classes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Top-1 class</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Top-5 classes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zhai et al. [35]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>a2g</td>
<td>X-Fork</td>
<td>1.9600</td>
<td>1.5908</td>
<td>2.0348</td>
</tr>
<tr>
<td></td>
<td>X-Seq</td>
<td>1.8503</td>
<td>1.4850</td>
<td>1.9623</td>
</tr>
<tr>
<td>Real Data</td>
<td>2.2096</td>
<td>1.6961</td>
<td>2.3008</td>
<td>–</td>
</tr>
</tbody>
</table>

### Table 2: Accuracies: Top-1 and Top-5.

<table>
<thead>
<tr>
<th>Dir.</th>
<th>Methods</th>
<th>64 × 64</th>
<th>256 × 256</th>
<th>CVUSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Top-1</td>
<td>Top-5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zhai et al. [35]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>a2g</td>
<td>X-Fork</td>
<td>16.63</td>
<td>34.73</td>
<td>46.35</td>
</tr>
<tr>
<td></td>
<td>X-Seq</td>
<td>4.83</td>
<td>5.56</td>
<td>19.55</td>
</tr>
<tr>
<td>Real Data</td>
<td>2.1408</td>
<td>1.4302</td>
<td>1.8606</td>
<td>–</td>
</tr>
</tbody>
</table>

### Table 3: KL Divergence between model and data distributions.

<table>
<thead>
<tr>
<th>Dir.</th>
<th>Method</th>
<th>64 × 64</th>
<th>256 × 256</th>
<th>CVUSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhai et al. [35]</td>
<td>–</td>
<td>–</td>
<td>27.43 ± 1.63</td>
<td>–</td>
</tr>
<tr>
<td>Pix2pix [12]</td>
<td>6.29 ± 0.8</td>
<td>38.26 ± 1.88</td>
<td>59.81 ± 2.12</td>
<td>–</td>
</tr>
<tr>
<td>a2g</td>
<td>X-Fork</td>
<td>3.42 ± 0.72</td>
<td>6.00 ± 1.28</td>
<td>11.71 ± 1.55</td>
</tr>
<tr>
<td></td>
<td>X-Seq</td>
<td>6.22 ± 0.87</td>
<td>5.93 ± 1.32</td>
<td>15.52 ± 1.73</td>
</tr>
<tr>
<td>Pix2pix [12]</td>
<td>6.39 ± 0.90</td>
<td>7.88 ± 1.24</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>g2a</td>
<td>X-Fork</td>
<td>4.45 ± 0.84</td>
<td>6.92 ± 1.15</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>X-Seq</td>
<td>7.20 ± 0.92</td>
<td>7.07 ± 1.19</td>
<td>–</td>
</tr>
</tbody>
</table>
Sharpness difference measures the loss of sharpness during image generation. To compute the sharpness difference between the generated image and the true image, we follow [19] and compute the difference of gradients between the images as

$$\text{SharpDiff.}(I_g, I'_g) = 10 \log_{10}(\frac{\max_{I_g}^2}{\text{grads}}),$$  \hspace{1cm} (13)

where, $\text{grads} = \frac{1}{N} \sum_i \sum_j \left| (\nabla_i I_g + \nabla_j I_g) - (\nabla_i I'_g + \nabla_j I'_g) \right|$

and, $\nabla_i I = |I_{i,j} - I_{i-1,j}|$, $\nabla_j I = |I_{i,j} - I_{i,j-1}|$.

Sharpness difference in Eqn. (13) is inverse of $\text{grads}$. We would like the $\text{grads}$ to be small, so the higher the overall score the better.

The scores are reported in Table 4. Over Dayton dataset, X-Seq model works the best in $\text{a2g}$ direction while X-Fork outperforms the rest in the $\text{g2a}$ direction. On CVUSA, X-Fork improves over Zhai et al. by 5.03% in SSIM, 8.93% in PSNR, and 12.35% in Sharpness difference.

Because there is no consensus in evaluation of GANs, we had to use several scores. Theis et al. [32] show that these scores often do not agree with each other and this was observed in our evaluations as well. So, it is difficult to infer whether X-Fork or X-Seq is better. We find that the proposed methods are consistently superior to the baselines.

### 6.3. Generated Segmentation Maps

Our methods generate semantic segmentation maps along with the real images in cross-view. The overlay of segmentation maps generated by X-Fork network (pre-trained RefineNet) on Dayton images are presented in the last (second) column of Figure 4. Please see supplementary materials for more qualitative results on semantic segmentation. The overlaid images show that the network is able to learn the semantic representations of object classes. For quantitative analysis, segmentation maps generated by our methods are compared against the segmentation maps obtained by applying RefineNet [16] to the target images. We compute per-class accuracies and mean IOU for the most common classes in our datasets: ‘vegetation’, ‘road’ and ‘building’ in aerial segmentation maps plus the ‘sky’ in ground segmentations. The scores are reported in Table 5. Even though X-Fork does better than X-Seq, we find that both methods achieve good scores for segmentation.

**Table 4:** SSIM, PSNR and Sharpness Difference between real data and samples generated using different methods.

<table>
<thead>
<tr>
<th>Dir.</th>
<th>Methods</th>
<th>$64 \times 64$</th>
<th>$256 \times 256$</th>
<th>CVUSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SSIM</td>
<td>PSNR</td>
<td>Sharp Diff</td>
<td>SSIM</td>
</tr>
<tr>
<td></td>
<td>Zhai et al. [35]</td>
<td>0.4808</td>
<td>19.4919</td>
<td>16.4489</td>
</tr>
<tr>
<td></td>
<td>Pix2pix [12]</td>
<td>0.3682</td>
<td>20.6933</td>
<td>14.7984</td>
</tr>
<tr>
<td></td>
<td>X-Fork</td>
<td>0.5171</td>
<td>20.104</td>
<td>16.6836</td>
</tr>
<tr>
<td></td>
<td>X-Seq</td>
<td>0.2763</td>
<td>20.5978</td>
<td>16.9962</td>
</tr>
</tbody>
</table>

**Table 5:** Evaluation Scores for segmentation maps.

<table>
<thead>
<tr>
<th>Methods</th>
<th>$\text{a2g}$</th>
<th>$\text{g2a}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per-class acc. mIOU</td>
<td>Per-class acc. mIOU</td>
<td></td>
</tr>
<tr>
<td>X-Fork</td>
<td>0.6262</td>
<td>0.4163</td>
</tr>
<tr>
<td>X-Seq</td>
<td>0.4783</td>
<td>0.3187</td>
</tr>
</tbody>
</table>

Figure 8: Along the columns, we show real image pairs, corresponding street-view image synthesized by X-Seq method and three nearest images in the training set retrieved by computing $L1$ distance between generated image and training set images.

### 6.4. kNN

Here, we test whether our proposed architectures have actually learned the representations between images in two views rather than just memorizing the blocks from training images to generate new ones. For this, we pick three images from the training set that are closest to the generated images in terms of $L1$ distance. As shown in Figure 8, the generated images have subtle differences with the training set images implying that our network has indeed learned important semantic representations in input view needed to transform the source image to target view.

### 7. Discussion and Conclusion

We explored image generation using conditional GANs between two drastically different views. Generating semantic segmentations together with images in target view helps the networks learn better images compared to the baselines. Extensive qualitative and quantitative evaluations testify the effectiveness of our methods. Using higher resolution images provided significant improvements in visual quality and added more details to synthesized images. The challenging nature of the problem leaves room for further improvements. Code and data will be shared.

**Acknowledgement.** We would like to thank NVIDIA for the donation of the Titan-X GPUs used in this work.
References


