Realistic and Varied Image Translation using conditional GANs

Andrew Jiang
Department of Computer Science
University of California, Irvine
andrewj3@uci.edu

Rajasekhar Reddy Mekala
Department of Computer Science
University of California, Irvine
rmekala@uci.edu

Agniraj Baikani
Department of Computer Science
University of California, Irvine
abaikani@uci.edu

Abstract

In this project, we re-implement Isola et al.'s work on "Image-to-Image Translation with Conditional Adversarial Networks" which aims to learn mappings from class of input images to class of output images. The previous works on image-to-image translation typically focus on deterministic mappings, and use $L2$ reconstruction loss functions that can induce blurriness. These approaches are problematic since the realism of the results depend heavily on proper loss function design and there is a lack of variability in image generation. In this approach, conditional generative adversarial networks are used to learn the image mappings using the $L1$ reconstruction loss. Training of generative adversarial networks are generally unstable and may lead to mode collapse which we hope to overcome through the addition of tailored loss functions on the GAN's input random variable. We will demonstrate that this approach is general and can be utilized effectively for many different datasets, such as at synthesizing photos from label maps, reconstructing facades from manually annotated images among other tasks. The ideal consequence of the project would be to gain a deeper understanding of training robust GAN networks, and provide methods that will improve both visual quality and realism in image translation. Code is available at "https://github.com/rajasekharmekala/pix2pix-experiments"

1 Introduction

GANs, also known as generative adversarial networks, are popular machine learning frameworks frequently used in tasks to generate new data that resembles the data seen in training. GANs in its most basic form consists of 2 different sets of architectures, a generator model and a discriminator model. Given a training data set, a generator attempts to generate new data that are distributionally equivalent to what is observed from the data set. Meanwhile, the discriminator's task is to accurately identify between real data and generated data from the generator. These two models are trained in conjunction. Typically, training is considered converged when the discriminator has a 50 percent accuracy on generated data which in theory means that the generator is creating data indistinguishable from the data set.

In our project, we implement and build upon the CGAN work by Isola et. al. [1]. CGANs are a type of GAN network that generates data based on a pre-existing structure. Typically, GANs take in a random variable z to generate data. While the generated data may be impossible to be discerned as real or fake, the properties of this generated data is solely determined by z. In CGANs, the input to the generator is z and data x. The goal of the generator is to produce $\hat{y}$ that closely resembles y,

which is the ground truth label of x. This is achieved by adding a loss that minimizes the different between y and $\hat{y}$ to the generator. This choice of loss functions along with loss scaling play a large role in the visual outcomes of image-to-image translation using GANs, and it is one area we will explore in this project. We will utilize datasets of images from class x and their transformed labels y. Specifically, we apply our model for reverse image segmentation and on satellite to street maps transformation.

2 Methodology

Conditional GANs used in this approach, learn to generate samples similar to the objective distribution based on prior observations (images in this case) and a random noise vector G: x, z $\rightarrow$ y, unlike the traditional GANs which only use the noise vector to generate samples similar to the training distribution G: z $\rightarrow$ y

2.1 Objective

The objective function of a cGAN can be formally expressed as

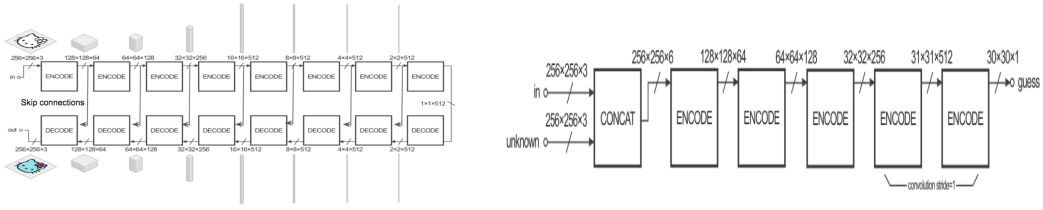
$$L_{cGAN}(G, D) = \mathop{\mathbb{E}}_{x,y}[logD(x, y)] + \mathop{\mathbb{E}}_{x,z}[log(1 - D(x, G(x, z)))]$$

Some approaches[2] have shown that using L1 and L2 regularization along with cGAN loss was effective in ensuring that the generated image is similar to the ground truth output labels. We experimented with both L1 and L2 losses and found that using L1 loss generates sharper images. Hence we included an L1 loss in our objective function.

$$G^* = arg \min_G \max_D L_{cGAN}(G, D) + \lambda L_{L1}(G)$$

2.2 GAN Architecture

The generator G is a modified UNet[3],[4] and the discriminator is a patchGAN inspired from [5][5], where the generator is trained to produce images that cannot be distinguished from "real" label images by a discriminator D, which is trained along with the generator to detect the generator's "fakes".



(a) Generator (b) Discriminator

Figure 1: Architecture of the adversarial network[9]

2.2.1 Unet Generator

The generator has a UNet architecture (shown in Figure 1(a)) is an encoder-decoder network with skip connections added across layers "i" and "n-i" (where n is the total number of layers). The skip connections intend to shuttle the structural features of the input image while decoding to the output image since the output label image has similar low-level features to the input image in translation tasks. Each skip connection concatenates channels of both the layers of the skip connection.

2.2.2 PatchGAN discriminator

The discriminator uses a patchGAN architecture(Figure 1(b)). Each node of the output layer focuses only on a particular patch of the generated image, to capture the high-frequency features and determine if it is fake. As suggested in [5], such a discriminator effectively models the image assuming independence between pixels separated by a distance of patch size.

3 Experiments

We test the above-mentioned architecture on both facades (400 images from CMP Facades [11]) and maps (1096 training images scraped from Google Maps) datasets for both upstream and downstream data generation tasks. For the Image colorization task, we sampled 5,000 images of different object categories for training from the COCO image dataset [12].

3.1 Training

We trained our GAN in a conventional method described in [7], alternating one gradient step on the discriminator and generator. We use the Adam optimization with initial learning rate of 2e-4 and optimization parameters $\beta 1$ = 0.5, $\beta 2$ = 0.999 as mentioned in the original work. We used batch normalization and dropout on the same layers mentioned in the original work and varied the experiments on various parameters like batch size, various distributions of the random vector(z), regularization constants. We used "instance normalization"(batch size of 1) for testing in all our experiments and has been proved to be effective by earlier works[8]. Figure 2 shows the results of generated images after 200 epochs, patchGAN receptive field of 70 X 70, trained with a batch size of 16 and images resized initially from 600 X 600 to 256 X 256, then to 286 X 286 and then cropped to 256 X 256

(a) Input (b) Label (c) Official software (d) Our implementation

Figure 2: Labels and corresponding generated images for facades and maps

3.2 Loss functions

We alter the loss calculation of several parts in our CGAN implementation in order to visualize the impact that they have. In the original pix2pix paper, $L1$ loss is used for the reconstruction loss. The loss is scaled by λ, where $\lambda = 100$ in our implementation. We try both $L1$ and $L2$ for the reconstruction loss.

We consider another loss as well through the addition of a second discriminator. This second discriminator is trained with an optimizer using $\beta 1 = 0.2$ and $\beta 2 = 0.9999$. This discriminator $D2$ is trained on the same data as the original discriminator. In the generator loss, we replace $D(x, G(x, z))$ with $D_f = elementwisemin(D1(x, G(x, z)), D2(x, G(x, z)))$. This is a more sensitive loss, as it takes the worst case performance of the generator across both discriminators. We believe the difference in β values for the discriminators help the discriminator pair generalize better to make the framework more pessimistic. The idea is that with different parameters, one discriminator may be more accurate during the earlier stages of training while the second discriminator may have better late-stage training performance. This forces the generator to learn better data generation throughout the training process.

In Figure 3, we see that $L2$ loss suffers from mode collapse. Overall, $L1$ reconstruction loss yields more accurate results than usage of $L2$. We believe that this is the case due to the loss values being

(a) input (b) L1 reconstruction (c) L2 reconstruction (d) L1 + dual discriminator

Figure 3: Images generated by alterations to the loss function

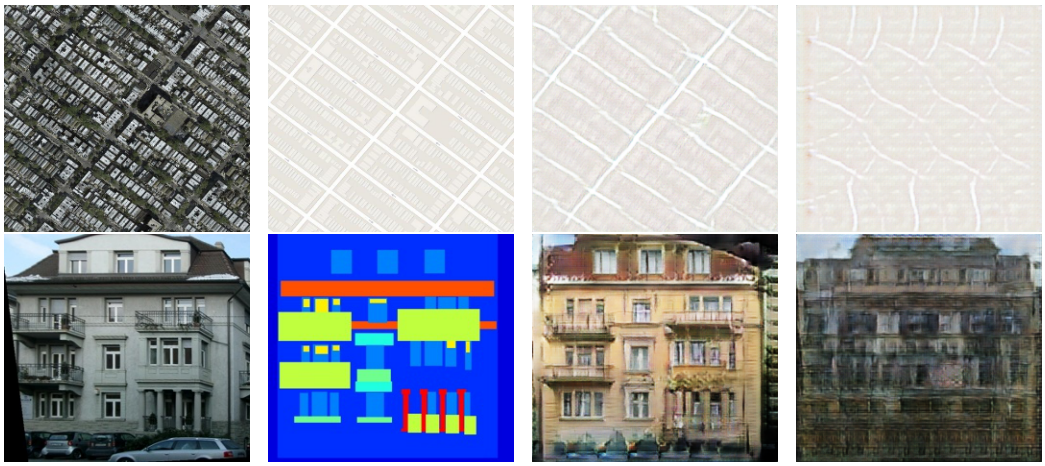
higher when using $L1$. This is because the pixel values are re-scaled to be between -1 and 1. As a result, there is a stronger incentive for the generated image to closely resemble the ground truth, and thus reducing the chance of mode collapse. $L1$ used alongside dual discriminator yielded the best results out of all three losses tested, with less blurriness than just using $L1$. We believe that because the discriminator system is more sensitive in the dual discriminator method, it is more easily able to tell if the image is real or fake through generator artifacts such as blurring. The generator trained using $L1$ alongside dual discriminator also shows better results in terms of river, road, and grass placements.

3.3 Generator architecture analysis

We evaluated different generator architectures impact in generating realistic images in our experiments. Original pix2pix paper implemented U-Net architecture in generator, which is simply an encoder-decoder framework with skip connections between the encoding and decoding layers. We tried both encoder-decoder and U-Net in our generator implementations. We observed that U-Net achieves superior results in generating higher-quality images shown in Figure 4. The encoder-decoder directly can't share information between the input and output across the bottleneck. Unlike regular encoder-decoder networks, U-Net makes use of skip connections to transfer information from the input image to the reconstructed image. These skip connections allow the generator to produce more precise images that resemble the input images.

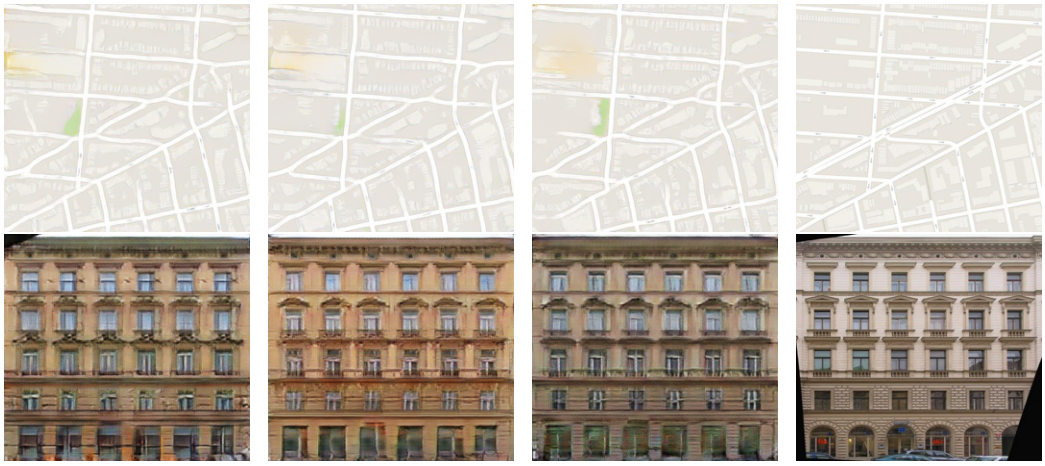
3.4 PatchGAN evaluation

To understand the effect of varying receptive fields of the patchGAN on the quality of generated images, we varied the number of hidden layers used in the discriminator from 1 to 5 resulting in receptive fields from 7 to 286. Figure 4 shows the generated images. We observed that for downstream tasks like images to maps, even smaller receptive fields of 16 X 16 performed well but for the upstream tasks like generating real facades and images from maps, the quality of the images improved with every additional layer, although some images produced artificial artifacts on the clear sky in the images.



(a) Input (b) Label (c) U-Net (d) encoder-decoder

Figure 4: U-Net and encoder-decoder results



(a) 16 X 16 (b) 70 X 70 (c) 286 X 286 (d) Label image

Figure 5: Images generated by varying PatchGAN receptive field

3.5 Black & White Image Colorization

We conducted experiments for coloring black and white images tasks using our cGAN architecture [10]. To train the model for colorization, we give input as a grayscale image and expect to make it colorful. Here to create a dataset, we use L*a*b color space (L encodes the Lightness, *a and *b channels encode how much green-red and yellow-blue each pixel are respectively) give the L channel as input (which is the grayscale image) and predict the other two channels (*a, *b) and concatenate for validation. We used only 5,000 images of different object categories for training from the COCO image dataset. Our training sample is 1.5% of the total dataset (330K images) learn to colorize and validate. We used default main model parameters, layers count as 3 with the receptive field of 70 X 70 in patchGAN Discriminator, and λ value as 100 for L1 loss. We trained the model for 30 epochs to produce reasonable outputs as shown in Figure 5.

4 Discussion and Conclusions

Since the evaluation of generated images is an open problem, manual evaluation is out of the scope of this work. We decided to compare our generated samples with the official software from the

Figure 6: Color image samples - Input, Generated, and real images in row order respectively

(a) Label image (b) Generated image (c) Label image (d) Generated image

Figure 7: Mode collapse

original work. Figure 2 compares the images generated by our work and the official software for both upstream and downstream tasks.

We observed that training the generated images is highly unstable. The quality of images depended on configurations such as batch size (larger batch sizes resulted in blurred images), initialization of the networks. Also, different noise distributions did not impact the generated images and the models seemed to ignore the noise vector altogether. Initial experiments repetitively resulted in mode collapse and generated only a few different images, as shown in Figure 3. Increasing the regularization of the training process by randomly flipping the input and label images horizontally, rescaling the input image to a slightly larger image, and randomly picking a cropped image of the size of the original input image for training significantly improved the generated samples. We observed a noticeable difference in the generated samples from the official software and our implementation. In Figure 2, Our model segmented the lanes better. But it could not capture the smaller blocks of buildings, unlike the software-generated image. Also, we observed that the generated images for the downstream tasks were much sharper and the training required far fewer epochs as expected than the upstream tasks.

The original work performed PatchGAN analysis on the cityscapes dataset and suggested that a patch size o 70 X 70 should work for upstream tasks as well and there was no need to use larger patch sizes. But we noticed that the images improved even from a patch size of 70 X 70 to 286 X 286 on the maps dataset. We attribute this to the fact that the cityscapes dataset contained images with clearly segmented objects in input images and contained only 4-5 entities. But maps dataset contained much dense information and no clear segmentation. Hence it required larger patch sizes.

5 References

[1] P. Isola, J.Y. Zhu, T. Zhou, A. Efros, Image-to-Image Translation with Conditional Adversarial Networks, Arxiv 2018.

[2] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A.Efros. Context encoders: Feature learning by inpainting. In CVPR, 2016.

[3] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In ICLR, 2016.

[4] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, 2015.

[5] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. ECCV, 2016.

[6] A. A. Efros and T. K. Leung. Texture synthesis by nonparametric sampling. In ICCV, 1999.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014.

[8] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022, 2016.

[9] https://medium.com/@pix2pix.datascience/investigating-the-impact-of-preprocessing-on-image-to-image-translation-a5273d49511e.

[10] Nazeri, K., Ng, E., Ebrahimi, M. (2018). Image Colorization Using Generative Adversarial Networks. Lecture Notes in Computer Science, 85–94.

[11] R. S. Radim Tyle ˇ cek. Spatial pattern templates for recognition of objects with regular structure. In German Conference on Pattern Recognition, 2013.

[12] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft COCO: Common objects ´in context," in ECCV, 2014.