Image-to-Image Translation with Conditional Adversarial Networks

Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros

Berkeley AI Research (BAIR) Laboratory, UC Berkeley

{isola,junyanz,tinghuiz,efros}@eecs.berkeley.edu

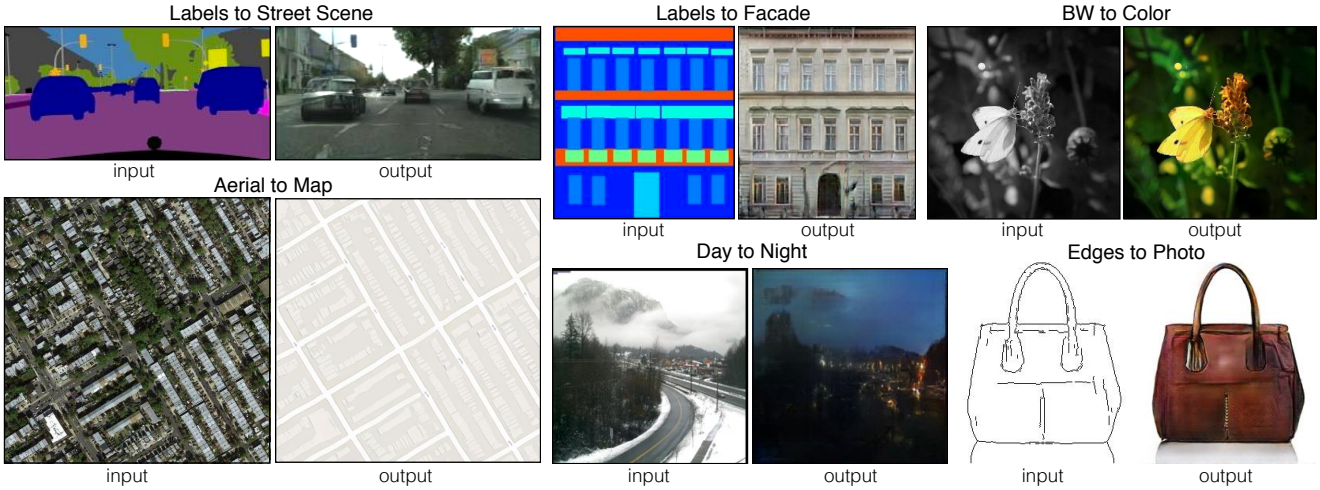


Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.

Abstract

We investigate conditional adversarial networks as a general-purpose solution to image-to-image translation problems. These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping. This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. We demonstrate that this approach is effective at synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images, among other tasks. Indeed, since the release of the `pix2pix` *software associated with this paper, a large number of internet users (many of them artists) have posted their own experiments with our system, further demonstrating its wide applicability and ease of adoption without the need for parameter tweaking. As a community, we no longer hand-engineer our mapping functions, and this work suggests we can achieve reasonable results without hand-engineering our loss functions either.*

1. Introduction

Many problems in image processing, computer graphics, and computer vision can be posed as "translating" an input image into a corresponding output image. Just as a concept may be expressed in either English or French, a scene may be rendered as an RGB image, a gradient field, an edge map, a semantic label map, etc. In analogy to automatic language translation, we define automatic *image-to-image translation* as the task of translating one possible representation of a scene into another, given sufficient training data (see Figure 1). Traditionally, each of these tasks has been tackled with separate, special-purpose machinery (e.g., [16, 25, 20, 9, 11, 53, 33, 39, 18, 58, 62]), despite the fact that the setting is always the same: predict pixels from pixels. Our goal in this paper is to develop a common framework for all these problems.

The community has already taken significant steps in this direction, with convolutional neural nets (CNNs) becoming the common workhorse behind a wide variety of image prediction problems. CNNs learn to minimize a loss function – an objective that scores the quality of results – and although the learning process is automatic, a lot of manual effort still

goes into designing effective losses. In other words, we still have to tell the CNN what we wish it to minimize. But, just like King Midas, we must be careful what we wish for! If we take a naive approach and ask the CNN to minimize the Euclidean distance between predicted and ground truth pixels, it will tend to produce blurry results [43, 62]. This is because Euclidean distance is minimized by averaging all plausible outputs, which causes blurring. Coming up with loss functions that force the CNN to do what we really want – e.g., output sharp, realistic images – is an open problem and generally requires expert knowledge.

It would be highly desirable if we could instead specify only a high-level goal, like "make the output indistinguishable from reality", and then automatically learn a loss function appropriate for satisfying this goal. Fortunately, this is exactly what is done by the recently proposed Generative Adversarial Networks (GANs) [24, 13, 44, 52, 63]. GANs learn a loss that tries to classify if the output image is real or fake, while simultaneously training a generative model to minimize this loss. Blurry images will not be tolerated since they look obviously fake. Because GANs learn a loss that adapts to the data, they can be applied to a multitude of tasks that traditionally would require very different kinds of loss functions.

In this paper, we explore GANs in the conditional setting. Just as GANs learn a generative model of data, conditional GANs (cGANs) learn a conditional generative model [24]. This makes cGANs suitable for image-to-image translation tasks, where we condition on an input image and generate a corresponding output image.

GANs have been vigorously studied in the last two years and many of the techniques we explore in this paper have been previously proposed. Nonetheless, earlier papers have focused on specific applications, and it has remained unclear how effective image-conditional GANs can be as a general-purpose solution for image-to-image translation. Our primary contribution is to demonstrate that on a wide variety of problems, conditional GANs produce reasonable results. Our second contribution is to present a simple framework sufficient to achieve good results, and to analyze the effects of several important architectural choices. Code is available at `https://github.com/phillipi/pix2pix`.

2. Related work

Structured losses for image modeling Image-to-image translation problems are often formulated as per-pixel classification or regression (e.g., [39, 58, 28, 35, 62]). These formulations treat the output space as "unstructured" in the sense that each output pixel is considered conditionally independent from all others given the input image. Conditional GANs instead learn a *structured loss*. Structured losses penalize the joint configuration of the output. A large body of literature has considered losses of this kind, with methods including conditional random fields [10], the SSIM metric [56], feature matching [15], nonparametric losses [37], the convolutional pseudo-prior [57], and losses based on matching covariance statistics [30]. The conditional GAN is different in that the loss is learned, and can, in theory, penalize any possible structure that differs between output and target.

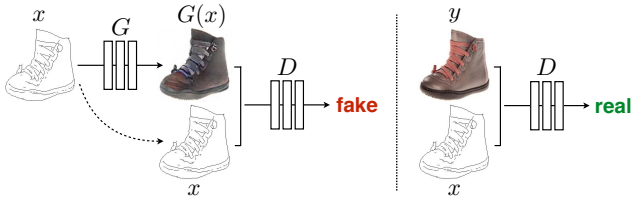


Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

Conditional GANs We are not the first to apply GANs in the conditional setting. Prior and concurrent works have conditioned GANs on discrete labels [41, 23, 13], text [46], and, indeed, images. The image-conditional models have tackled image prediction from a normal map [55], future frame prediction [40], product photo generation [59], and image generation from sparse annotations [31, 48] (c.f. [47] for an autoregressive approach to the same problem). Several other papers have also used GANs for image-to-image mappings, but only applied the GAN unconditionally, relying on other terms (such as L2 regression) to force the output to be conditioned on the input. These papers have achieved impressive results on inpainting [43], future state prediction [64], image manipulation guided by user constraints [65], style transfer [38], and superresolution [36]. Each of the methods was tailored for a specific application. Our framework differs in that nothing is application-specific. This makes our setup considerably simpler than most others.

Our method also differs from the prior works in several architectural choices for the generator and discriminator. Unlike past work, for our generator we use a "U-Net"-based architecture [50], and for our discriminator we use a convolutional "PatchGAN" classifier, which only penalizes structure at the scale of image patches. A similar PatchGAN architecture was previously proposed in [38] to capture local style statistics. Here we show that this approach is effective on a wider range of problems, and we investigate the effect of changing the patch size.

3. Method

GANs are generative models that learn a mapping from random noise vector z to output image y, $G : z \rightarrow y$ [24]. In

contrast, conditional GANs learn a mapping from observed image x and random noise vector z, to y, $G : \{x, z\} \rightarrow y$. The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes". This training procedure is diagrammed in Figure 2.

3.1. Objective

The objective of a conditional GAN can be expressed as

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))], \quad (1)$$

where G tries to minimize this objective against an adversarial D that tries to maximize it, i.e. $G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D)$.

To test the importance of conditioning the discriminator, we also compare to an unconditional variant in which the discriminator does not observe x:

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{y}[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x, z))]. \quad (2)$$

Previous approaches have found it beneficial to mix the GAN objective with a more traditional loss, such as L2 distance [43]. The discriminator's job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be near the ground truth output in an L2 sense. We also explore this option, using L1 distance rather than L2 as L1 encourages less blurring:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]. \quad (3)$$

Our final objective is

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G). \quad (4)$$

Without z, the net could still learn a mapping from x to y, but would produce deterministic outputs, and therefore fail to match any distribution other than a delta function. Past conditional GANs have acknowledged this and provided Gaussian noise z as an input to the generator, in addition to x (e.g., [55]). In initial experiments, we did not find this strategy effective – the generator simply learned to ignore the noise – which is consistent with Mathieu et al. [40]. Instead, for our final models, we provide noise only in the form of dropout, applied on several layers of our generator at both training and test time. Despite the dropout noise, we observe only minor stochasticity in the output of our nets. Designing conditional GANs that produce highly stochastic output, and thereby capture the full entropy of the conditional distributions they model, is an important question left open by the present work.

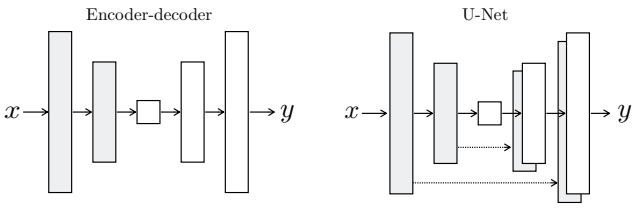


Figure 3: Two choices for the architecture of the generator. The "U-Net" [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

3.2. Network architectures

We adapt our generator and discriminator architectures from those in [44]. Both generator and discriminator use modules of the form convolution-BatchNorm-ReLu [29]. Details of the architecture are provided in the supplemental materials online, with key features discussed below.

3.2.1 Generator with skips

A defining feature of image-to-image translation problems is that they map a high resolution input grid to a high resolution output grid. In addition, for the problems we consider, the input and output differ in surface appearance, but both are renderings of the same underlying structure. Therefore, structure in the input is roughly aligned with structure in the output. We design the generator architecture around these considerations.

Many previous solutions [43, 55, 30, 64, 59] to problems in this area have used an encoder-decoder network [26]. In such a network, the input is passed through a series of layers that progressively downsample, until a bottleneck layer, at which point the process is reversed. Such a network requires that all information flow pass through all the layers, including the bottleneck. For many image translation problems, there is a great deal of low-level information shared between the input and output, and it would be desirable to shuttle this information directly across the net. For example, in the case of image colorization, the input and output share the location of prominent edges.

To give the generator a means to circumvent the bottleneck for information like this, we add skip connections, following the general shape of a "U-Net" [50]. Specifically, we add skip connections between each layer i and layer $n - i$, where n is the total number of layers. Each skip connection simply concatenates all channels at layer i with those at layer $n - i$.

3.2.2 Markovian discriminator (PatchGAN)

It is well known that the L2 loss – and L1, see Figure 4 – produces blurry results on image generation problems [34]. Although these losses fail to encourage high-

frequency crispness, in many cases they nonetheless accurately capture the low frequencies. For problems where this is the case, we do not need an entirely new framework to enforce correctness at the low frequencies. L1 will already do.

This motivates restricting the GAN discriminator to only model high-frequency structure, relying on an L1 term to force low-frequency correctness (Eqn. 4). In order to model high-frequencies, it is sufficient to restrict our attention to the structure in local image patches. Therefore, we design a discriminator architecture – which we term a *Patch*GAN – that only penalizes structure at the scale of patches. This discriminator tries to classify if each $N \times N$ patch in an image is real or fake. We run this discriminator convolutionally across the image, averaging all responses to provide the ultimate output of D.

In Section 4.4, we demonstrate that N can be much smaller than the full size of the image and still produce high quality results. This is advantageous because a smaller PatchGAN has fewer parameters, runs faster, and can be applied to arbitrarily large images.

Such a discriminator effectively models the image as a Markov random field, assuming independence between pixels separated by more than a patch diameter. This connection was previously explored in [38], and is also the common assumption in models of texture [17, 21] and style [16, 25, 22, 37]. Therefore, our PatchGAN can be understood as a form of texture/style loss.

3.3. Optimization and inference

To optimize our networks, we follow the standard approach from [24]: we alternate between one gradient descent step on D, then one step on G. As suggested in the original GAN paper, rather than training G to minimize $\log(1 - D(x, G(x, z))$, we instead train to maximize $\log D(x, G(x, z))$ [24]. In addition, we divide the objective by 2 while optimizing D, which slows down the rate at which D learns relative to G. We use minibatch SGD and apply the Adam solver [32], with a learning rate of 0.0002, and momentum parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$.

At inference time, we run the generator net in exactly the same manner as during the training phase. This differs from the usual protocol in that we apply dropout at test time, and we apply batch normalization [29] using the statistics of the test batch, rather than aggregated statistics of the training batch. This approach to batch normalization, when the batch size is set to 1, has been termed "instance normalization" and has been demonstrated to be effective at image generation tasks [54]. In our experiments, we use batch sizes between 1 and 10 depending on the experiment.

4. Experiments

To explore the generality of conditional GANs, we test the method on a variety of tasks and datasets, including both graphics tasks, like photo generation, and vision tasks, like semantic segmentation:

- *Semantic labels↔photo*, trained on the Cityscapes dataset [12].
- *Architectural labels→photo*, trained on CMP Facades [45].
- *Map↔aerial photo*, trained on data scraped from Google Maps.
- *BW→color photos*, trained on [51].
- *Edges→photo*, trained on data from [65] and [60]; binary edges generated using the HED edge detector [58] plus postprocessing.
- *Sketch→photo*: tests edges→photo models on human-drawn sketches from [19].
- *Day→night*, trained on [33].
- *Thermal→color photos*, trained on data from [27].
- *Photo with missing pixels→inpainted photo*, trained on Paris StreetView from [14].

Details of training on each of these datasets are provided in the supplemental materials online. In all cases, the input and output are simply 1-3 channel images. Qualitative results are shown in Figures 8, 9, 11, 10, 13, 14, 15, 16, 17, 18, 19, 20. Several failure cases are highlighted in Figure 21. More comprehensive results are available at `https://phillipi.github.io/pix2pix/`.

Data requirements and speed We note that decent results can often be obtained even on small datasets. Our facade training set consists of just 400 images (see results in Figure 14), and the day to night training set consists of only 91 unique webcams (see results in Figure 15). On datasets of this size, training can be very fast: for example, the results shown in Figure 14 took less than two hours of training on a single Pascal Titan X GPU. At test time, all models run in well under a second on this GPU.

4.1. Evaluation metrics

Evaluating the quality of synthesized images is an open and difficult problem [52]. Traditional metrics such as per-pixel mean-squared error do not assess joint statistics of the result, and therefore do not measure the very structure that structured losses aim to capture.

To more holistically evaluate the visual quality of our results, we employ two tactics. First, we run "real vs. fake" perceptual studies on Amazon Mechanical Turk (AMT). For graphics problems like colorization and photo generation, plausibility to a human observer is often the ultimate goal. Therefore, we test our map generation, aerial photo generation, and image colorization using this approach.

Figure 4: Different losses induce different quality of results. Each column shows results trained under a different loss. Please see `https://phillipi.github.io/pix2pix/` for additional examples.

Second, we measure whether or not our synthesized cityscapes are realistic enough that off-the-shelf recognition system can recognize the objects in them. This metric is similar to the "inception score" from [52], the object detection evaluation in [55], and the "semantic interpretability" measures in [62] and [42].

AMT perceptual studies For our AMT experiments, we followed the protocol from [62]: Turkers were presented with a series of trials that pitted a "real" image against a "fake" image generated by our algorithm. On each trial, each image appeared for 1 second, after which the images disappeared and Turkers were given unlimited time to respond as to which was fake. The first 10 images of each session were practice and Turkers were given feedback. No feedback was provided on the 40 trials of the main experiment. Each session tested just one algorithm at a time, and Turkers were not allowed to complete more than one session. $\sim$ 50 Turkers evaluated each algorithm. Unlike [62], we did not include vigilance trials. For our colorization experiments, the real and fake images were generated from the same grayscale input. For map↔aerial photo, the real and fake images were not generated from the same input, in order to make the task more difficult and avoid floor-level results. For map↔aerial photo, we trained on 256×256 resolution images, but exploited fully-convolutional translation (described above) to test on 512×512 images, which were then downsampled and presented to Turkers at 256×256 resolution. For colorization, we trained and tested on 256×256 resolution images and presented the results to Turkers at this same resolution.

"FCN-score" While quantitative evaluation of generative models is known to be challenging, recent works [52, 55, 62, 42] have tried using pre-trained semantic classifiers to measure the discriminability of the generated stimuli as a pseudo-metric. The intuition is that if the generated images are realistic, classifiers trained on real images will be able to classify the synthesized image correctly as well. To this end, we adopt the popular FCN-8s [39] architecture for semantic segmentation, and train it on the cityscapes dataset. We then score synthesized photos by the classification accuracy against the labels these photos were synthesized from.

4.2. Analysis of the objective function

Which components of the objective in Eqn. 4 are important? We run ablation studies to isolate the effect of the L1 term, the GAN term, and to compare using a discriminator conditioned on the input (cGAN, Eqn. 1) against using an unconditional discriminator (GAN, Eqn. 2).

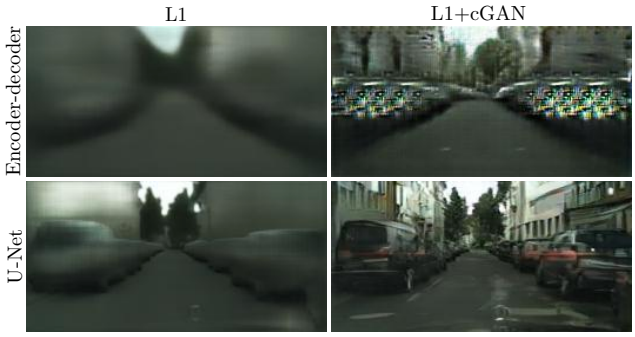


Figure 5: Adding skip connections to an encoder-decoder to create a "U-Net" results in much higher quality results.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
L1	0.42	0.15	0.11
GAN	0.22	0.05	0.01
cGAN	0.57	0.22	0.16
L1+GAN	0.64	0.20	0.15
L1+cGAN	**0.66**	**0.23**	**0.17**
Ground truth	0.80	0.26	0.21

How about use GIOU?

Table 1: FCN-scores for different losses, evaluated on Cityscapes labels↔photos.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
Encoder-decoder (L1)	0.35	0.12	0.08
Encoder-decoder (L1+cGAN)	0.29	0.09	0.05
U-net (L1)	0.48	0.18	0.13
U-net (L1+cGAN)	**0.55**	**0.20**	**0.14**

Table 2: FCN-scores for different generator architectures (and objectives), evaluated on Cityscapes labels↔photos. (U-net (L1-cGAN) scores differ from those reported in other tables since batch size was 10 for this experiment and 1 for other tables, and random variation between training runs.)

Discriminator receptive field	Per-pixel acc.	Per-class acc.	Class IOU
1×1	0.39	0.15	0.10
16×16	0.65	0.21	**0.17**
70×70	**0.66**	**0.23**	**0.17**
286×286	0.42	0.16	0.11

Table 3: FCN-scores for different receptive field sizes of the discriminator, evaluated on Cityscapes labels→photos. Note that input images are 256 × 256 pixels and larger receptive fields are padded with zeros.

Figure 4 shows the qualitative effects of these variations on two labels→photo problems. L1 alone leads to reasonable but blurry results. The cGAN alone (setting $\lambda = 0$ in Eqn. 4) gives much sharper results but introduces visual artifacts on certain applications. Adding both terms together (with $\lambda = 100$) reduces these artifacts.

We quantify these observations using the FCN-score on the cityscapes labels→photo task (Table 1): the GAN-based objectives achieve higher scores, indicating that the synthesized images include more recognizable structure. We also test the effect of removing conditioning from the discriminator (labeled as GAN). In this case, the loss does not penalize mismatch between the input and output; it only cares that the output look realistic. This variant results in poor performance; examining the results reveals that the generator collapsed into producing nearly the exact same output regardless of input photograph. Clearly, it is important, in this case, that the loss measure the quality of the match between input and output, and indeed cGAN performs much better than GAN. Note, however, that adding an L1 term also encourages that the output respect the input, since the L1 loss penalizes the distance between ground truth outputs, which correctly match the input, and synthesized outputs, which may not. Correspondingly, L1+GAN is also effective at creating realistic renderings that respect the input label maps. Combining all terms, L1+cGAN, performs similarly well.

Colorfulness A striking effect of conditional GANs is that they produce sharp images, hallucinating spatial structure even where it does not exist in the input label map. One might imagine cGANs have a similar effect on "sharpening" in the spectral dimension – i.e. making images more colorful. Just as L1 will incentivize a blur when it is uncertain where exactly to locate an edge, it will also incentivize an average, grayish color when it is uncertain which of several plausible color values a pixel should take on. Specially, L1 will be minimized by choosing the median of the conditional probability density function over possible colors. An adversarial loss, on the other hand, can in principle become aware that grayish outputs are unrealistic, and encourage matching the true color distribution [24]. In Figure 7, we investigate whether our cGANs actually achieve this effect on the Cityscapes dataset. The plots show the marginal distributions over output color values in Lab color space. The ground truth distributions are shown with a dotted line. It is apparent that L1 leads to a narrower distribution than the ground truth, confirming the hypothesis that L1 encourages average, grayish colors. Using a cGAN, on the other hand, pushes the output distribution closer to the ground truth.

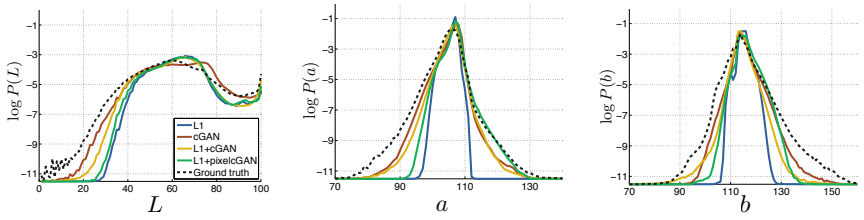
4.3. Analysis of the generator architecture

A U-Net architecture allows low-level information to shortcut across the network. Does this lead to better results? Figure 5 and Table 2 compare the U-Net against an encoder-decoder on cityscape generation. The encoder-decoder is created simply by severing the skip connections in the U-Net. The encoder-decoder is unable to learn to generate realistic images in our experiments. The advantages of the U-Net appear not to be specific to conditional GANs: when both U-Net and encoder-decoder are trained with an L1 loss, the U-Net again achieves the superior results.

4.4. From PixelGANs to PatchGANs to ImageGANs

We test the effect of varying the patch size N of our discriminator receptive fields, from a 1×1 "PixelGAN" to a

Figure 6: Patch size variations. Uncertainty in the output manifests itself differently for different loss functions. Uncertain regions become blurry and desaturated under L1. The 1x1 PixelGAN encourages greater color diversity but has no effect on spatial statistics. The 16x16 PatchGAN creates locally sharp results, but also leads to tiling artifacts beyond the scale it can observe. The 70×70 PatchGAN forces outputs that are sharp, even if incorrect, in both the spatial and spectral (colorfulness) dimensions. The full 286×286 ImageGAN produces results that are visually similar to the 70×70 PatchGAN, but somewhat lower quality according to our FCN-score metric (Table 3). Please see `https://phillipi.github.io/pix2pix/` for additional examples.



(a) (b) (c)

Loss	Histogram intersection against ground truth L	a	b
L1	0.81	0.69	0.70
cGAN	**0.87**	0.74	**0.84**
L1+cGAN	0.86	**0.84**	0.82
PixelGAN	0.83	0.68	0.78

(d)

Figure 7: Color distribution matching property of the cGAN, tested on Cityscapes. (c.f. Figure 1 of the original GAN paper [24]). Note that the histogram intersection scores are dominated by differences in the high probability region, which are imperceptible in the plots, which show log probability and therefore emphasize differences in the low probability regions.

full 286 × 286 "ImageGAN"[1]. Figure 6 shows qualitative results of this analysis and Table 3 quantifies the effects using the FCN-score. Note that elsewhere in this paper, unless specified, all experiments use 70 × 70 PatchGANs, and for this section all experiments use an L1+cGAN loss.

The PixelGAN has no effect on spatial sharpness but does increase the colorfulness of the results (quantified in Figure 7). For example, the bus in Figure 6 is painted gray when the net is trained with an L1 loss, but becomes red with the PixelGAN loss. Color histogram matching is a common problem in image processing [49], and PixelGANs may be a promising lightweight solution.

Using a 16 × 16 PatchGAN is sufficient to promote sharp outputs, and achieves good FCN-scores, but also leads to tiling artifacts. The 70 × 70 PatchGAN alleviates these artifacts and achieves slightly better scores. Scaling beyond this, to the full 286 × 286 ImageGAN, does not appear to improve the visual quality of the results, and in fact gets a considerably lower FCN-score (Table 3). This may be because the ImageGAN has many more parameters and greater depth than the 70 × 70 PatchGAN, and may be harder to train.

Fully-convolutional translation An advantage of the PatchGAN is that a fixed-size patch discriminator can be applied to arbitrarily large images. We may also apply the generator convolutionally, on larger images than those on which it was trained. We test this on the map↔aerial photo task. After training a generator on 256 × 256 images, we test it on 512 × 512 images. The results in Figure 8 demonstrate the effectiveness of this approach.

[1]We achieve this variation in patch size by adjusting the depth of the GAN discriminator. Details of this process, and the discriminator architectures, are provided in the in the supplemental materials online.

Loss	Photo → Map % Turkers labeled *real*	Map → Photo % Turkers labeled *real*
L1	2.8% ± 1.0%	0.8% ± 0.3%
L1+cGAN	6.1% ± 1.3%	**18.9% ± 2.5%**

Table 4: AMT "real vs fake" test on maps↔aerial photos.

Method	% Turkers labeled *real*
L2 regression from [62]	16.3% ± 2.4%
Zhang et al. 2016 [62]	**27.8% ± 2.7%**
Ours	22.5% ± 1.6%

Table 5: AMT "real vs fake" test on colorization.

4.5. Perceptual validation

We validate the perceptual realism of our results on the tasks of map↔aerial photograph and grayscale→color. Results of our AMT experiment for map↔photo are given in Table 4. The aerial photos generated by our method fooled participants on 18.9% of trials, significantly above the L1 baseline, which produces blurry results and nearly never fooled participants. In contrast, in the photo→map direction our method only fooled participants on 6.1% of trials, and this was not significantly different than the performance of the L1 baseline (based on bootstrap test). This may be because minor structural errors are more visible

Figure 8: Example results on Google Maps at 512x512 resolution (model was trained on images at 256 × 256 resolution, and run convolutionally on the larger images at test time). Contrast adjusted for clarity.

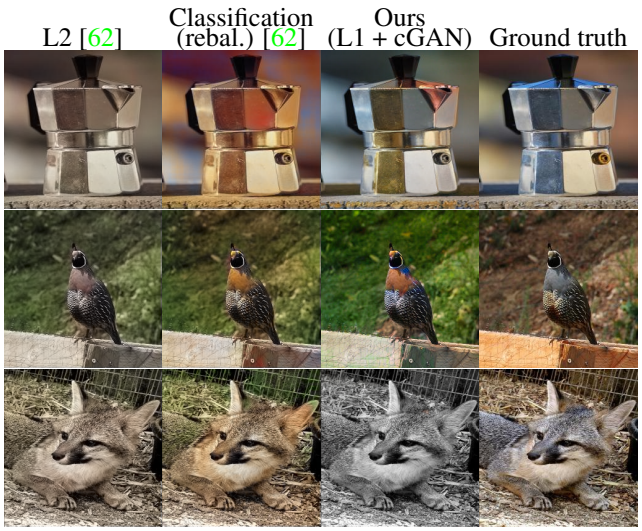


Figure 9: Colorization results of conditional GANs versus the L2 regression from [62] and the full method (classification with rebalancing) from [64]. The cGANs can produce compelling colorizations (first two rows), but have a common failure mode of producing a grayscale or desaturated result (last row).

in maps, which have rigid geometry, than in aerial photographs, which are more chaotic.

We trained colorization on ImageNet [51], and tested on the test split introduced by [62, 35]. Our method, with L1+cGAN loss, fooled participants on 22.5% of trials (Table 5). We also tested the results of [62] and a variant of their method that used an L2 loss (see [62] for details). The conditional GAN scored similarly to the L2 variant of [62] (difference insignificant by bootstrap test), but fell short of [62]'s full method, which fooled participants on 27.8% of trials in our experiment. We note that their method was specifically engineered to do well on colorization.

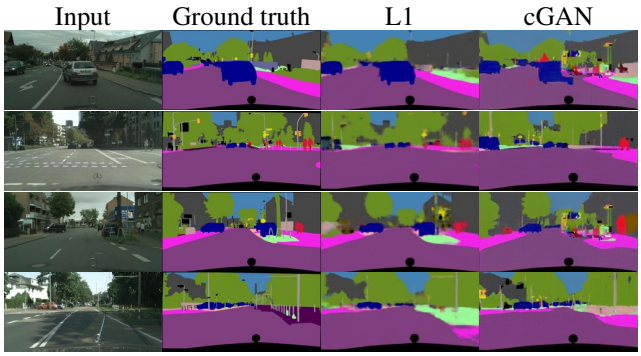


Figure 10: Applying a conditional GAN to semantic segmentation. The cGAN produces sharp images that look at glance like the ground truth, but in fact include many small, hallucinated objects.

4.6. Semantic segmentation

Conditional GANs appear to be effective on problems where the output is highly detailed or photographic, as is common in image processing and graphics tasks. What

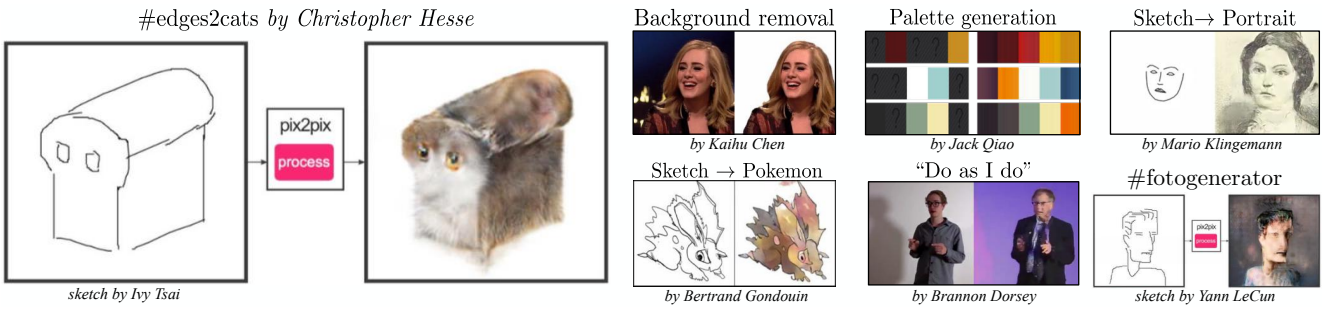


Figure 11: Example applications developed by online community based on our `pix2pix` codebase: *#edges2cats* [3] by Christopher Hesse, *Background removal* [6] by Kaihu Chen, *Palette generation* [5] by Jack Qiao, *Sketch → Portrait* [7] by Mario Klingemann, *Sketch→ Pokemon* [1] by Bertrand Gondouin, *"Do As I Do" pose transfer* [2] by Brannon Dorsey, and *#fotogenerator* by Bosman et al. [4].

Loss	Per-pixel acc.	Per-class acc.	Class IOU
L1	**0.86**	**0.42**	**0.35**
cGAN	0.74	0.28	0.22
L1+cGAN	0.83	0.36	0.29

Table 6: Performance of photo→labels on cityscapes.

about vision problems, like semantic segmentation, where the output is instead less complex than the input?

To begin to test this, we train a cGAN (with/without L1 loss) on cityscape photo→labels. Figure 10 shows qualitative results, and quantitative classification accuracies are reported in Table 6. Interestingly, cGANs, trained *without* the L1 loss, are able to solve this problem at a reasonable degree of accuracy. To our knowledge, this is the first demonstration of GANs successfully generating "labels", which are nearly discrete, rather than "images", with their continuous-valued variation[2]. Although cGANs achieve some success, they are far from the best available method for solving this problem: simply using L1 regression gets better scores than using a cGAN, as shown in Table 6. We argue that for vision problems, the goal (i.e. predicting output close to the ground truth) may be less ambiguous than graphics tasks, and reconstruction losses like L1 are mostly sufficient.

4.7. Community-driven Research

Since the initial release of the paper and our `pix2pix` codebase, the Twitter community, including computer vision and graphics practitioners as well as visual artists, have successfully applied our framework to a variety of novel image-to-image translation tasks, far beyond the scope of the original paper. Figure 11 and Figure 12 show just a few examples from the #pix2pix hashtag, including *Background removal*, *Palette generation*, *Sketch → Portrait*, *Sketch→Pokemon*, *"Do as I Do" pose transfer*, *Learning to see: Gloomy Sunday*, as well as the bizarrely popular #edges2cats and #fotogenerator. Note that these applications are creative projects, were not obtained in controlled, scientific conditions, and may rely on some modifications to the `pix2pix` code we released. Nonetheless, they demonstrate the promise of our approach as a generic commodity tool for image-to-image translation problems.

[2]Note that the label maps we train on are not exactly discrete valued, as they are resized from the original maps using bilinear interpolation and saved as jpeg images, with some compression artifacts.

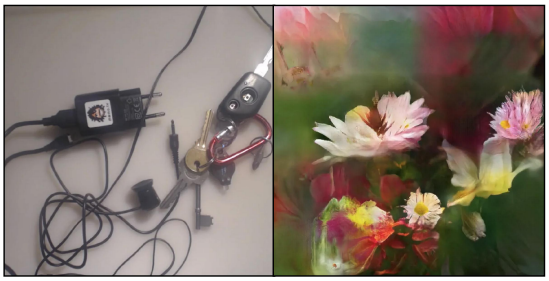


Figure 12: *Learning to see: Gloomy Sunday*: An interactive artistic demo developed by Memo Akten [8] based on our `pix2pix` codebase. Please click the image to play the video in a browser.

5. Conclusion

The results in this paper suggest that conditional adversarial networks are a promising approach for many image-to-image translation tasks, especially those involving highly structured graphical outputs. These networks learn a loss adapted to the task and data at hand, which makes them applicable in a wide variety of settings.

Acknowledgments: We thank Richard Zhang, Deepak Pathak, and Shubham Tulsiani for helpful discussions, Saining Xie for help with the HED edge detector, and the online community for exploring many applications and suggesting improvements. Thanks to Christopher Hesse, Memo Akten, Kaihu Chen, Jack Qiao, Mario Klingemann, Brannon Dorsey, Gerda Bosman, Ivy Tsai, and Yann LeCun for allowing the use of their creations in Figure 11 and Figure 12. This work was supported in part by NSF SMA-1514512, NGA NURI, IARPA via Air Force Research Laboratory, Intel Corp, Berkeley Deep Drive, and hardware donations by Nvidia. J.-Y.Z. is supported by the Facebook Graduate Fellowship. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL or the U.S. Government.

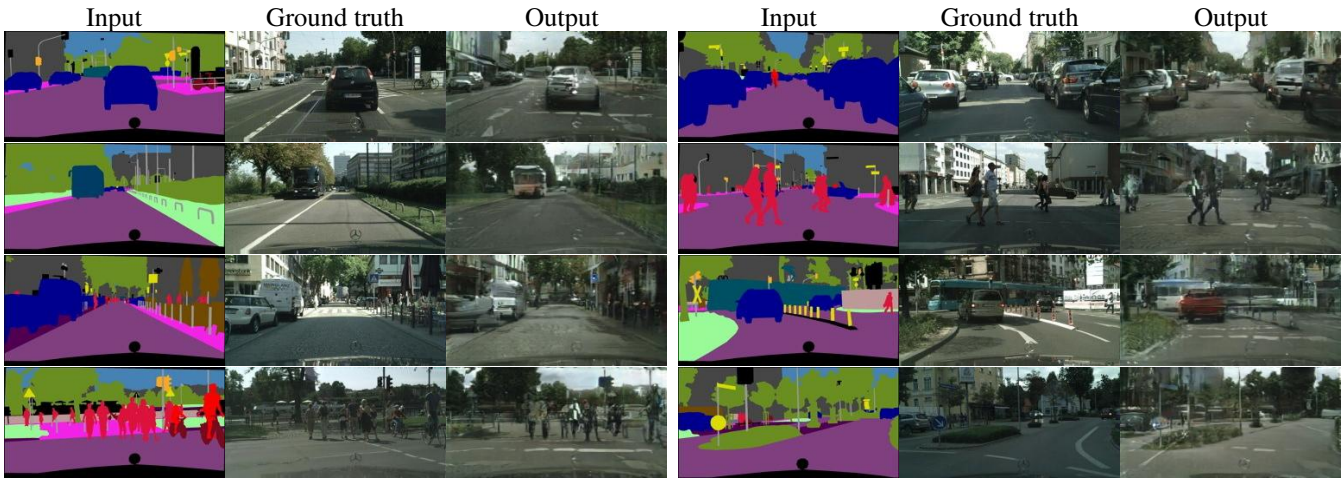


Figure 13: Example results of our method on Cityscapes labels→photo, compared to ground truth.

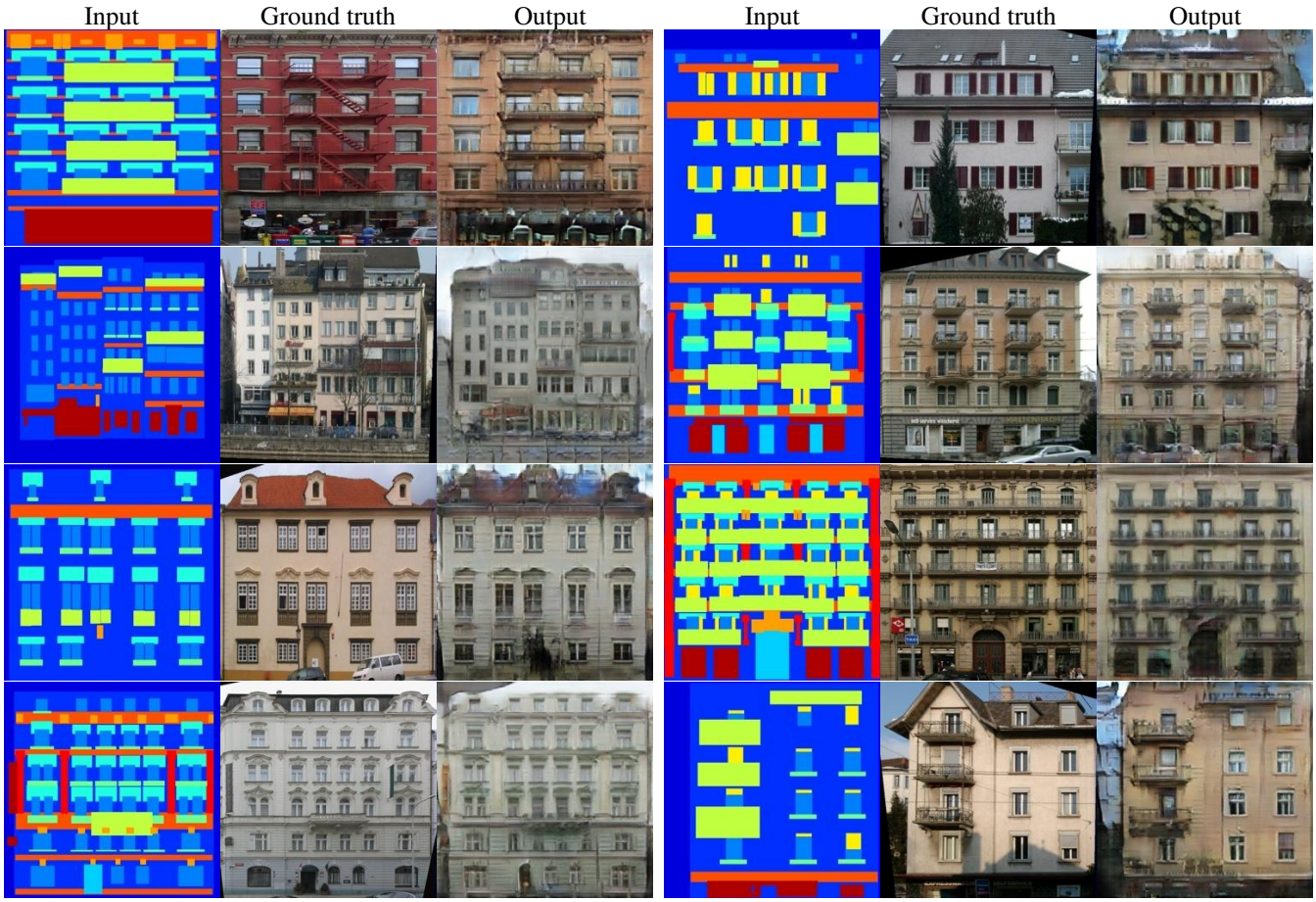


Figure 14: Example results of our method on facades labels→photo, compared to ground truth.

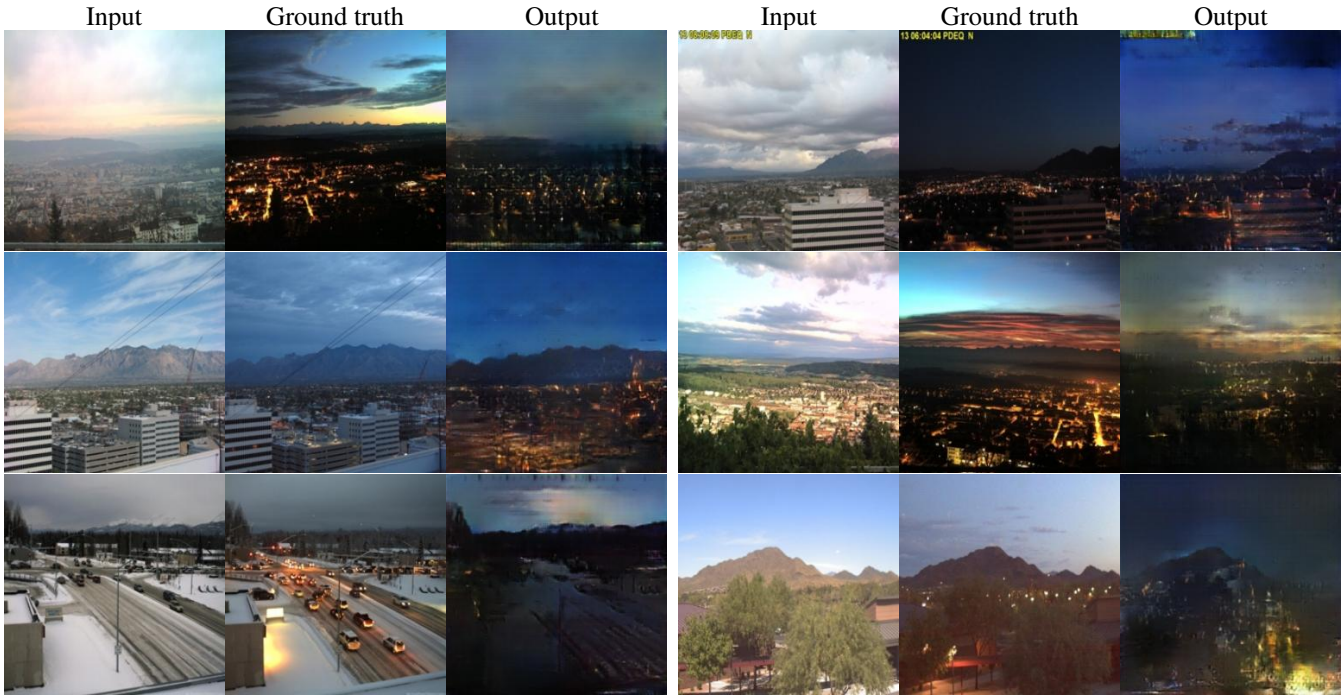


Figure 15: Example results of our method on day→night, compared to ground truth.

Figure 16: Example results of our method on automatically detected edges→handbags, compared to ground truth.

Figure 17: Example results of our method on automatically detected edges→shoes, compared to ground truth.

Figure 18: Additional results of the edges→photo models applied to human-drawn sketches from [19]. Note that the models were trained on automatically detected edges, but generalize to human drawings

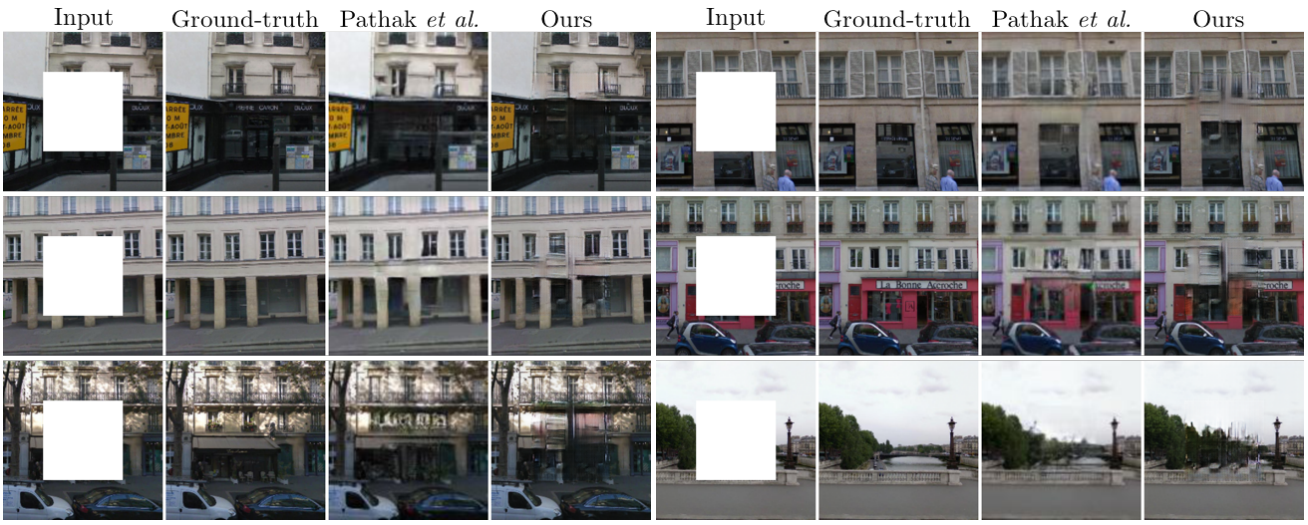


Figure 19: Example results on photo inpainting, compared to [43], on the Paris StreetView dataset [14]. This experiment demonstrates that the U-net architecture can be effective even when the predicted pixels are not geometrically aligned with the information in the input – the information used to fill in the central hole has to be found in the periphery of these photos.

Figure 20: Example results on translating thermal images to RGB photos, on the dataset from [27].

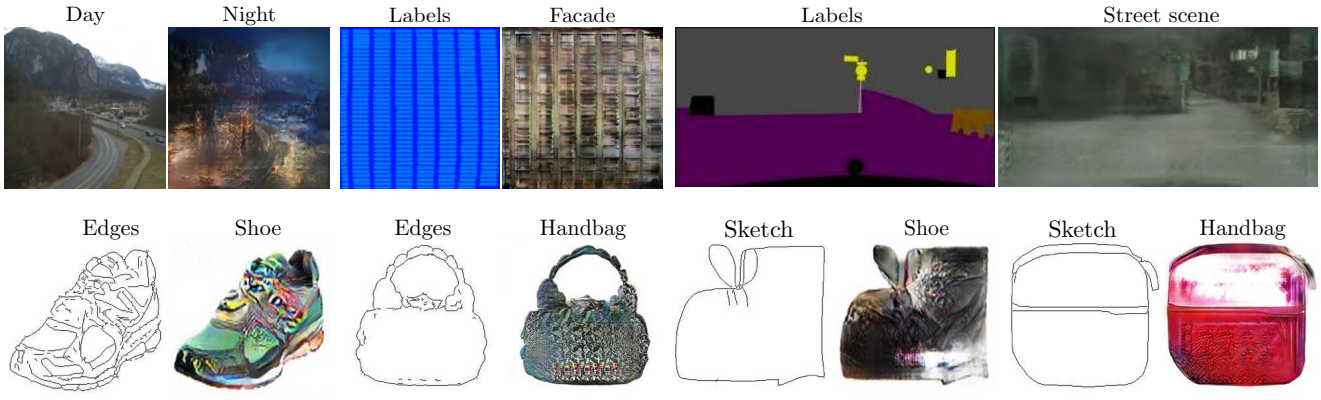


Figure 21: Example failure cases. Each pair of images shows input on the left and output on the right. These examples are selected as some of the worst results on our tasks. Common failures include artifacts in regions where the input image is sparse, and difficulty in handling unusual inputs. Please see `https://phillipi.github.io/pix2pix/` for more comprehensive results.

References

[1] Bertrand gondouin. https://twitter.com/bgondouin/status/818571935529377792. Accessed, 2017-04-21. 9

[2] Brannon dorsey. https://twitter.com/brannondorsey/status/806283494041223168. Accessed, 2017-04-21. 9

[3] Christopher hesse. https://affinelayer.com/pixsrv/. Accessed: 2017-04-21. 9

[4] Gerda bosman, tom kenter, rolf jagerman, and daan gosman. https://dekennisvannu.nl/site/artikel/Help-ons-kunstmatige-intelligentie-testen/9163. Accessed: 2017-08-31. 9

[5] Jack qiao. http://colormind.io/blog/. Accessed: 2017-04-21. 9

[6] Kaihu chen. http://www.terraai.org/imageops/index.html. Accessed, 2017-04-21. 9

[7] Mario klingemann. https://twitter.com/quasimondo/status/826065030944870400. Accessed, 2017-04-21. 9

[8] Memo akten. https://vimeo.com/260612034. Accessed, 2018-11-07. 9

[9] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *CVPR*, 2005. 1

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 2

[11] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu. Sketch2photo: internet image montage. *ACM Transactions on Graphics (TOG)*, 28(5):124, 2009. 1

[12] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 4, 16

[13] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015. 2

[14] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros. What makes paris look like paris? *ACM Transactions on Graphics*, 31(4), 2012. 4, 13, 17

[15] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016. 2

[16] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001. 1, 4

[17] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999. 4

[18] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 1

[19] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? In *SIGGRAPH*, 2012. 4, 12

[20] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics (TOG)*, 25(3):787–794, 2006. 1

[21] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015. 4

[22] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. *CVPR*, 2016. 4

[23] J. Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, (5):2, 2014. 2

[24] 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. 2, 4, 6, 7

[25] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *SIGGRAPH*, 2001. 1, 4

[26] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 3

[27] S. Hwang, J. Park, N. Kim, Y. Choi, and I. So Kweon. Multispectral pedestrian detection: Benchmark dataset and baseline. In *CVPR*, 2015. 4, 13, 16

[28] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (TOG)*, 35(4), 2016. 2

[29] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 3, 4

[30] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2, 3

[31] L. Karacan, Z. Akata, A. Erdem, and E. Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *arXiv preprint arXiv:1612.00215*, 2016. 2

[32] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 4

[33] P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (TOG)*, 33(4):149, 2014. 1, 4, 16

[34] A. B. L. Larsen, S. K. Sønderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016. 3

[35] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. *ECCV*, 2016. 2, 8, 16

[36] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 2

[37] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. *CVPR*, 2016. 2, 4

[38] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. *ECCV*, 2016. 2, 4

[39] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2, 5

[40] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *ICLR*, 2016. 2, 3

[41] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[42] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman. Visually indicated sounds. In *CVPR*, 2016. 5

[43] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2, 3, 13, 17

[44] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 2, 3, 16

[45] R. Š. Radim Tyleček. Spatial pattern templates for recognition of objects with regular structure. In *German Conference on Pattern Recognition*, 2013. 4, 16

[46] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *ICML*, 2016. 2

[47] S. Reed, A. van den Oord, N. Kalchbrenner, V. Bapst, M. Botvinick, and N. de Freitas. Generating interpretable images with controllable structure. In *ICLR Workshop*, 2017. 2

[48] S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. Learning what and where to draw. In *NIPS*, 2016. 2

[49] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21:34–41, 2001. 7

[50] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2, 3

[51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 4, 8, 16

[52] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NIPS*, 2016. 2, 4, 5

[53] Y. Shih, S. Paris, F. Durand, and W. T. Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)*, 32(6):200, 2013. 1

[54] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 4

[55] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016. 2, 3, 5

[56] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 2

[57] S. Xie, X. Huang, and Z. Tu. Top-down learning for structured labeling with convolutional pseudoprior. In *ECCV*, 2015. 2

[58] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 1, 2, 4

[59] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon. Pixel-level domain transfer. *ECCV*, 2016. 2, 3

[60] A. Yu and K. Grauman. Fine-Grained Visual Comparisons with Local Learning. In *CVPR*, 2014. 4

[61] A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *CVPR*, 2014. 16

[62] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *ECCV*, 2016. 1, 2, 5, 7, 8, 16

[63] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. In *ICLR*, 2017. 2

[64] Y. Zhou and T. L. Berg. Learning temporal transformations from time-lapse videos. In *ECCV*, 2016. 2, 3, 8

[65] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 2, 4, 16

6. Appendix

6.1. Network architectures

We adapt our network architectures from those in [44]. Code for the models is available at `https://github.com/phillipi/pix2pix`.

Let `Ck` denote a Convolution-BatchNorm-ReLU layer with k filters. `CDk` denotes a Convolution-BatchNorm-Dropout-ReLU layer with a dropout rate of 50%. All convolutions are 4×4 spatial filters applied with stride 2. Convolutions in the encoder, and in the discriminator, downsample by a factor of 2, whereas in the decoder they upsample by a factor of 2.

6.1.1 Generator architectures

The encoder-decoder architecture consists of:
encoder:
`C64-C128-C256-C512-C512-C512-C512-C512`
decoder:
`CD512-CD512-CD512-C512-C256-C128-C64`

After the last layer in the decoder, a convolution is applied to map to the number of output channels (3 in general, except in colorization, where it is 2), followed by a Tanh function. As an exception to the above notation, BatchNorm is not applied to the first `C64` layer in the encoder. All ReLUs in the encoder are leaky, with slope 0.2, while ReLUs in the decoder are not leaky.

The U-Net architecture is identical except with skip connections between each layer i in the encoder and layer $n-i$ in the decoder, where n is the total number of layers. The skip connections concatenate activations from layer i to layer $n-i$. This changes the number of channels in the decoder:

U-Net decoder:
`CD512-CD1024-CD1024-C1024-C1024-C512-C256-C128`

6.1.2 Discriminator architectures

The 70×70 **discriminator** architecture is:
`C64-C128-C256-C512`

After the last layer, a convolution is applied to map to a 1-dimensional output, followed by a Sigmoid function. As an exception to the above notation, BatchNorm is not applied to the first `C64` layer. All ReLUs are leaky, with slope 0.2.

All other discriminators follow the same basic architecture, with depth varied to modify the receptive field size:

1×1 **discriminator:**
`C64-C128` (note, in this special case, all convolutions are 1×1 spatial filters)
16×16 **discriminator:**
`C64-C128`
286×286 **discriminator:**
`C64-C128-C256-C512-C512-C512`

6.2. Training details

Random jitter was applied by resizing the 256×256 input images to 286×286 and then randomly cropping back to size 256×256.

All networks were trained from scratch. Weights were initialized from a Gaussian distribution with mean 0 and standard deviation 0.02.

Cityscapes labels→photo 2975 training images from the Cityscapes training set [12], trained for 200 epochs, with random jitter and mirroring. We used the Cityscapes validation set for testing. To compare the U-net against an encoder-decoder, we used a batch size of 10, whereas for the objective function experiments we used batch size 1. We find that batch size 1 produces better results for the U-net, but is inappropriate for the encoder-decoder. This is because we apply batchnorm on all layers of our network, and for batch size 1 this operation zeros the activations on the bottleneck layer. The U-net can skip over the bottleneck, but the encoder-decoder cannot, and so the encoder-decoder requires a batch size greater than 1. Note, an alternative strategy is to remove batchnorm from the bottleneck layer. See errata for more details.

Architectural labels→photo 400 training images from [45], trained for 200 epochs, batch size 1, with random jitter and mirroring. Data were split into train and test randomly.

Maps↔aerial photograph 1096 training images scraped from Google Maps, trained for 200 epochs, batch size 1, with random jitter and mirroring. Images were sampled from in and around New York City. Data were then split into train and test about the median latitude of the sampling region (with a buffer region added to ensure that no training pixel appeared in the test set).

BW→color 1.2 million training images (Imagenet training set [51]), trained for $\sim$ 6 epochs, batch size 4, with only mirroring, no random jitter. Tested on subset of Imagenet val set, following protocol of [62] and [35].

Edges→shoes 50k training images from UT Zappos50K dataset [61] trained for 15 epochs, batch size 4. Data were split into train and test randomly.

Edges→Handbag 137K Amazon Handbag images from [65], trained for 15 epochs, batch size 4. Data were split into train and test randomly.

Day→night 17823 training images extracted from 91 webcams, from [33] trained for 17 epochs, batch size 4, with random jitter and mirroring. We use 91 webcams as training, and 10 webcams for test.

Thermal→color photos 36609 training images from set 00–05 of [27], trained for 10 epochs, batch size 4. Images from set 06-11 are used for testing.

Photo with missing pixels→inpainted photo 14900 training images from [14], trained for 25 epochs, batch size 4, and tested on 100 held out images following the split of [43].

6.3. Errata

For all experiments reported in this paper with batch size 1, the activations of the bottleneck layer are zeroed by the batchnorm operation, effectively making the innermost layer skipped. This issue can be fixed by removing batchnorm from this layer, as has been done in the public code. We observe little difference with this change and therefore leave the experiments as is in the paper.

6.4. Change log

arXiv v2 Reran generator architecture comparisons (Section 4.3) with batch size equal to 10 rather than 1, so that bottleneck layer is not zeroed (see Errata). Reran FCN-scores with minor details cleaned up (results saved losslessly as pngs, removed unecessary downsampling). FCN-scores computed using scripts at `https://github.com/phillipi/pix2pix/tree/master/scripts/eval_cityscapes`, commit d7e7b8b. Updated several figures and text. Added additional results on thermal→color photos and inpainting, as well as community contributions.

arXiv v3 Added additional results on community contributions. Fixed minor typos.