

# Image-to-Image Translation with Conditional Adversarial Nets논문을 번역, 설명추가 및 내용정리

-by puostyoon(<https://github.com/puostyoon>)

프로젝트 링크: <https://phillipi.github.io/pix2pix/>

Original code: <https://github.com/phillipi/pix2pix>

Recommended version code: <https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix>

이 문서는 “Image-to-Image Trnaslation with Conditional Adversarial Nets”논문을 한국어로 번역하고, 논문에 상세히 설명되지 않는 내용들에 대한 설명을 추가하여 정리한 문서입니다. 이 문서의 작성 시점에서 필자는 “밑바닥부터 시작하는 딥러닝”, “GAN 첫걸음” 두 권의 책을 읽어 머신러닝과 GAN, 그리고 조건부GAN에 대한 사전지식이 있는 상태에서 작성하였습니다. 그러므로 머신러닝과 GAN에 대한 기본적인 사전지식이 없다면 읽기 힘들 수 있습니다. 다음문단부터 바로 내용번역이 시작됩니다.

## 1. abstract 이전 (figure 1)

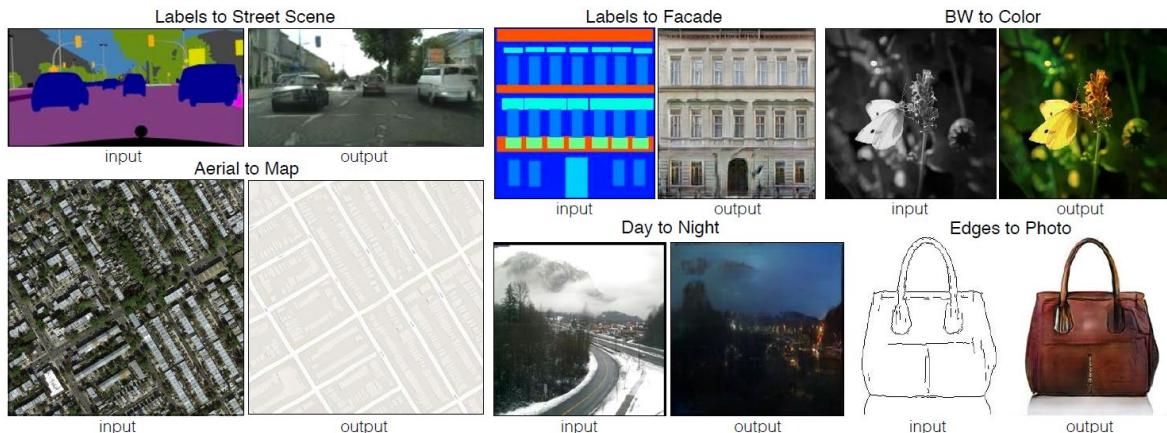


image processing 처리는 application-specific하다. Conditional adversarial net은 general-purpose solution이다. 위 사진의 모든 경우에서 same architecture, objective(목적함수)를 썼으며, 단지 다른 data를 이용하여 학습했을 뿐이다.

## 2. abstract

Conditional adversarial network를 image-to-image translation에 대한 general-purpose solution으로 사용한다. 이 network는 input-image와 output-image 사이의 mapping과, 이 mapping을 train하기 위한 loss function을 학습한다. 그리하여 원래 매우 다양한 loss function을 필요 하는 문제들에 대해 동일한 일반적인 해결책을 제공한다. 논문에서 이러한 방법론이 특히 label map으로부터 사진을 합성하고, edge maps으로부터 물체들을 reconstruct하고, 사진들을 채색하는 데 효과적임

을 보인다. 우리가 이 논문과 함께 pix2pix software를 공개했기 때문에 예술가를 비롯한 수많은 사용자들이 그들 자신의 실험결과를 올려주었다. 그래서 이 기술의 폭넓은 활용가능성과, parameter에 대한 수정 없이도 기술을 사용할 수 있음을 보여주었다. 그래서 Loss function과 mapping function을 일일이 수정하지 않고도 좋은 결과를 얻을 수 있음을 보였다.

### 3. Introduction

Image processing, computer graphics, computer vision의 많은 문제는 input image를 output image로 바꾸는 작업이라 할 수 있다. 하나의 장면은 RGB image, gradient field, edge map, semantic label map 등등 다양한 방식으로 표현될 수 있다. 충분한 데이터를 가지고, 우리는 image-to-image 변환 훈련하는 것을 일종의 언어 번역기 훈련에 비유할 수 있다. 원래 image-to-image 변환 자체는 pixel로부터 다음 pixel을 예측하는 같은 업무지만, 각각의 업무는 각각의 특정한 목적을 위해 설계된 기계를 가지고 써름하였다. 이 논문의 목표는 image-to-image 변환에 대한 일반적인 프레임워크를 만들어내는 것이다.

이미 학계는 다양한 이미지 예측 문제의 배후에서 잘 작동하는 CNNs을 가지고 일반 프레임워크 제작에 의미있는 발걸음을 내딛었다. CNNs는 결과의 품질의 점수를 매기는 loss function을 최소화하려고 한다. 그리고 비록 학습 과정이 자동으로 이루어지지만, 효과적인 loss-function을 디자인하기 위해 여전히 수많은 manual effort가 필요하다. 즉 우리는 여전히 CNN에게 우리가 무엇을 최소화하고 싶은지를 알려줘야 한다. 그러나 단순히, 예측한 픽셀과 실제 픽셀 사이의 Euclidean distance를 줄이려고 하면 blurry result를 만들어낸다. 왜냐하면 Euclidean distance를 최소화하려면 모든 그럴듯한 output을 평균내는 식으로 output을 만들기 때문이다. CNN이 우리가 원하는 아웃풋을 내도록 하는 loss function을 생각해 내는 일은 아직 완전히 해결되지 않았다. 대신, 우리가 "output을 실재와 구분하지 못하도록 만들어라"와 같은 high-level goal을 만들고, 목표에 적합한 loss function을 학습시키는 것이 바람직한 방법이 될 수 있다. 다행히 이 과정이 정확히 최근에 제시된 GANs에서 수행되는 일이다. GANs는 output image가 진짜인지 가짜인지 구분하는 loss를 학습한다. 동시에 이 loss를 최소화하려는 generative model을 학습시킨다. Blurry image는 가짜처럼 보이므로 생성되지 않을 것이다. GANs는 data에 적합한 loss를 학습하므로 GANs는 전통적으로 매우 다양한 loss function을 요구할 수많은 task에 적용될 수 있다.

이 논문에서, 우리는 conditional setting 상태에 있는 GANs를 탐구한다. GANs이 data의 generative model을 학습하는 것처럼, conditional GANs(cGANs)는 conditional generative model을 학습한다. 그래서 cGANs은 우리가 input image를 조건으로 하여 그에 대응하는 output image를 만들어내는 image-to-image task에 적합하다. GANs는 지난 2년간 활기차게 연구되어왔으며, 이 논문에서 탐구하는 많은 techniques의 대부분은 이전에 제안되었던 것들이다. 그럼에도 불구하고 이전의 논문들은 specific application에 초점을 맞쳤고, 그래서 image-conditional GANs이 image-to-image translation에 대한 general-purpose solution으로 얼마나 효과적인지에 대해 불분명하게 남았다. 우리의 주된 contribution은 다양한 문제들에 대해 conditional GANs가 타당한 결과를 만들어내는 것을 보여주는 것이다. 우리의 두 번째 contribution은 좋은 결과를 만들어 내는 데 충

분한, simple framework를 보여주는 것이다. 그리고 몇몇 중요한 architectural choices를 분석하는 것이다. 코드는 <https://github.com/phillipi/pix2pix> 에서 이용할 수 있다.

#### 4. Related work

##### Structured losses for image modeling

Image-to-image translation은 주로 per-pixel classification, regression으로 formulate될 수 있다. 이 formulation은 output space를 “unstructured”라고 간주한다. 왜냐하면 input image가 주어졌을 때, 각각의 output pixel이 다른 모든 것들과 conditionally independent하다고 간주되기 때문이다. 대신에 Conditional GANs는 structured loss를 학습한다. Structured loss는 output의 joint configuration에 대해 패널티를 준다. 수많은 문헌들이 이러한 종류의 losses를, conditional random fields[10], the SSIM metric[56], feature matching[15], nonparametric losses[37], the convolutional pseudo-prior[57], losses based on matching covariance statistics[30]와 같은 방법과 함께 고려하였다. Conditional GANs는 loss가 학습되어 이론적으로는 출력과 target 사이의 차이를 다르게 하는 그 어떤 possible structure도 penalize할 수 있다는 점에서 차별화된다.

##### Conditional GANs

우리가 처음으로 GANs를 conditional setting으로 사용하는 것은 아니다. 이전과 현재의 works는 discrete labels[41, 23, 13], text[46], 그리고 image에 대한 conditioned GANs을 가지고 있다. Image-conditioned model은 image prediction from normal map[55], future frame prediction[40], product photo generation[59], 그리고 image generation from sparse annotations[31, 48](c.f. [47] for an autoregressive approach to the same problem)을 다루었다. 몇몇 다른 논문들은 또한 GANs를 image-to-image mapping에 사용했지만, output이 input에 조건화하도록 강제하기 위해 오직 GAN을 unconditionally, 다른 조건( L2 regression 같은 것)에 의존하여서만 활용하였다. 이 논문들은 inpainting[43], future state prediction[64], image manipulation guided by user constraints[65], style transfer[38], 그리고 superresolution[36]에 대해 인상적인 결과를 성취했다. 각각의 방법들은 specific application을 위해 맞춰졌다. 우리의 framework는 아무것도 application-specific하지 않다는 점에서 다르다. 이로 인해 우리의 설정은 다른 대부분의 것들보다도 훨씬 간단하다.

우리의 방법은 또한 generator와 discriminator의 architectural choice에 있어서도 이전의 작업들과 차별화된다. 이전의 작업과 다르게, 우리의 generator를 위해 우리는 “U-Net”-based architecture<sup>1</sup>[50]를 사용한다. 그리고 discriminator를 위해선 convolutional “PatchGAN” classifier를 사용한다. Convolutional “PatchGAN” classifier는 오직 scale of image patches의 구조에만 패널티를

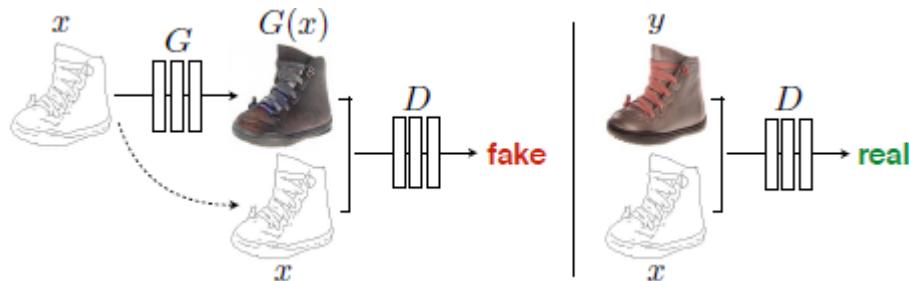
---

<sup>1</sup> U-Net은 입력 이미지의 Context포착을 위한 Contracting Path와 세밀한 Localization을 위한 Expanding path의 대칭적인 형태로 구성되어 있다. 그리고 Skip Architecture 또한 활용한다.

(<https://medium.com/@msmapark2/u-net-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-u-net-convolutional-networks-for-biomedical-image-segmentation-456d6901b28a> 참고)

준다. Local style statistics를 capture할 수 있는 비슷한 PatchGAN architecture<sup>2</sup>[38]가 이전에 제시되었다. 우리는 이러한 접근방식이 광범위한 문제에 효과적임을 보이고, patch size의 크기를 변경하는 것의 효과를 탐구할 것이다.

### 5. figure2



Edges->photo를 mapping 하는 cGAN을 학습시키는 과정. Discriminator D는 generator에 의해 합성된 가짜 {edge,photo} tuple과 실제 {edge,photo} tuple들을 구분하도록 학습된다. Generator G는 disrcriminator를 속이는 방식으로 학습한다. Unconditional GAN과 달리, generator와 discriminator 둘 다 input edge map을 들여다본다.

### 6. Method

GANs는 random noise vector  $z$ 로부터 output image  $y$ 로의 mapping을 학습하는 generative model이다. 즉  $G: z \rightarrow y$  이다[24]. 반면, Conditional GANs는 observed image  $x$ 와 random noise vector  $z$ 로부터  $y$ 로의 mapping을 학습한다. 즉  $G: \{x, z\} \rightarrow y$  이다. Generator G는, adversarially trained discriminator D가 보기에 실제와 구분되지 않는 이미지를 만들도록 훈련받는다. 그리고 D또한 G의 가짜 이미지를 최대한 잘 detect하도록 훈련받는다. 이 훈련 과정은 Figure2에 나타나져있다.

### 7. Objectice

Conditional GAN의 objective(목적함수)는

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

이렇게 표현될 수 있다.

$G$ 는 이 objective를 최소화하려 하고  $D$ 는 최대화하려고 한다. 즉

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

---

<sup>2</sup> PatchGAN의 discriminator는 합성곱을 이용하여 픽셀들의 조합인 patch 단위로 사진의 진위여부를 판별한다. (<https://brstar96.github.io/mldlstudy/what-is-patchgan-D/> 참고)

이렇다.

Discriminator를 conditioning하는 것의 중요성을 시험하기 위해, 또한 discriminator가  $x$ 를 들여다보지 않는 unconditional variant에 비교해볼 수 있다:

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

이전의 접근방식들은 GAN목적함수를 L2 distance와 같은 전통적인 loss와 함께 사용하는 것이 효과적이라는 것을 발견했다[43]. Discriminator의 역할은 변하지 않았다. 그러나 Generator는 discriminator를 속일 뿐만 아니라 in an L2 sense, ground truth output과 비슷해야 할 필요도 있다. 또한 L1이 덜 blurring 시킬 수 있기 때문에 L2 distance 대신 L1 distance를 사용하는 선택지 또한 탐구한다.

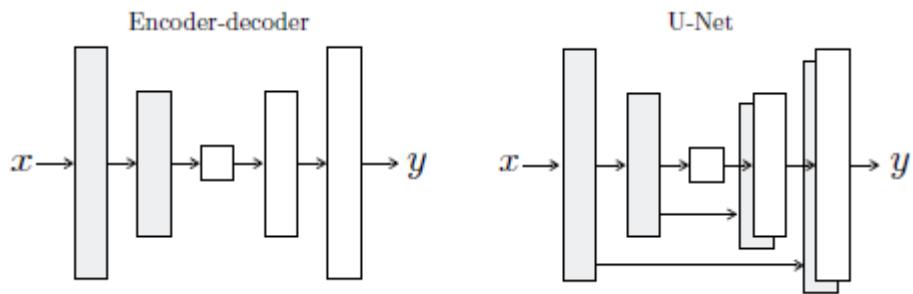
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z} [\|y - G(x, z)\|_1]. \quad (3)$$

우리의 최종 목적함수는

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

이다.  $z$ 없이도, 신경망은 여전히  $x$ 에서  $y$ 로의 mapping을 학습할 수 있다. 그러나 결정론적인 결과를 만들어내어 delta function외의 다른 어떤 분포에 맞추는 데 실패할 것이다. 즉 random noise  $z$  없이는 input condition  $x$  하나에 따라 하나만의 결과만을 만들어 내기 때문에 deterministic(결과론적인)한 결과를 만들어내게 되어 안좋다. 이전의 cGANs는 이러한 성질 때문에 input으로써  $x$ 와 더불어 Gaussian noise  $z$ 를 generator에 제공했다(e.g., [55]). 초기의 실험들에서, 우리는 이 전략이 효과적임을 find하지 못했다.(이 전략이 효과적이지 않은 것 같다고 말하고 싶은듯함.)-generator는 단순히 noise를 무시하도록 학습했다-그리고 이는 Mathieu등의 연구[40]에서의 결과와 일치한다. 대신에, 우리의 최종 model에서 우리는 noise를 dropout의 형식으로만 제공한다.(dropout이란 각각의 데이터를 학습 시에 신경망의 임의의 노드의 연결을 끊는 학습기법. overfitting예방에도 효과적이고, 하나의 신경망으로 ensemble학습과 같은 효과를 내기도 함.) dropout은 training time과 test time 둘 다에서, 우리의 generator의 몇몇 레이어에 적용된다.(원래는 흔히 training time에서만 dropout을 적용함. 근데 test 시에서, dropout으로 gan의 random noise와 같은 효과를 내려면 test시에도 dropout을 적용해야 함이 마땅하다.) Dropout noise를 사용했지만, 우리는 우리의 신경망의 출력에서 오직 약간의 확률성(stochasticity)를 관찰할 수 있었다. 고도의 확률적 출력물을 내놓는 Conditional GANs을 설계하여 conditional GANs이 model하는 conditional distribution의 불확실성 모두를 capture하는 것은 현재 work에선, 중요한 열린 질문으로 남아있다.

8. Figure3



Generator의 구조에 대한 두 가지 옵션이다. "U-Net"[50]은 encoder와 decoder stacks의 대칭적 계층사이에 skip connection이 있는 encoder-decoder이다.

## 9. Network architectures

우리는 [44]에 나오는(<https://arxiv.org/abs/1511.06434>) discriminator와 generator의 architecture를 선택한다. Generator와 Discriminator 모두 convolution-BatchNorm-ReLu[29] 모듈을 사용한다. Architecture의 세부사항은 온라인의 supplemental materials에 architecture의 핵심 특징들은 아래에서 다룰 것이다.

## 10. Generator with skips

Image-to-image translation 문제들의 정의적 세부특징(defining feature)은 image-to-image translation이 고해상도(high resolution) 인풋 grid(격자, 영역)을 고해상도 아웃풋 grid로 mapping 한다는 것이다. 게다가 우리가 고려하는 문제들에 대해서, input과 output은 겉으로 보기에 다르지만 input과 output 모두가 같은 underlying structure의 renderings(번역)이다. 그래서 input의 structure는 output에서의 structure와 roughly aligned 되어있다. 우리는 이러한 고려사항들을 중심으로 generator architecture를 디자인한다.

이 분야의 문제들에 대한 과거의 많은 해결책들[43, 55, 30, 64, 59]은 encoder-decoder 신경망[26]을 사용했다. 이 encoder-decoder 신경망에서 input은, process를 reverse 하는 bottleneck layer까지 progressively downsample하는 일련의 계층들을 지난다. 이러한 신경망은 모든 정보흐름이 bottleneck layer를 포함한 모든 계층에 전달된다. 많은 image translation problem에서, input과 output사이에 많은 low-level information이 있다. 그리고 이러한 low-level information을 신경망을 직접 가로질러서 전달하는 것이 바람직하다. 예를 들어, image colorization의 경우에, input과 output은 두드러진 edge의 위치를 공유한다.

Generator에 이와 같은 정보들이 bottleneck을 우회할 수 있는 수단을 제공하기 위해 우리는 "U-Net"[50]의 일반적인 형태를 따라하여 skip connections를 추가한다. 구체적으로, 우리는 각각의 layer  $i$ 와 layer  $n-i$  사이에 skip connections를 추가한다. 여기서  $n$ 은 layer들의 총 개수이다. 각각의 skip connection은 단지 layer  $i$ 의 모든 채널들을 layer  $n-i$ 의 모든 채널들에 연결한다.

## 11. Figure 4



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.

## 12. Markovian discriminator (PatchGAN)

L2 loss, 그리고 L1 loss가 이미지 생성 문제에서 (Figure4를 보라) blurry result를 만들어 낸다고 알려져 있다[34]. 비록 이 losses들이 high frequency crispness를 encourage하는 데 실패했지만, 많은 경우에 loss들은 그럼에도 불구하고 정확하게 low frequencies를 capture했다.<sup>3</sup> 이러한 문제 상황에서, 우리는 low frequencies에서 정확도를 강화하기 위한 완전히 새로운 framework를 필요로 하지는 않는다. L1 loss가 이미 low frequencies에서 잘 작동하고 있다.

이 때문에 GAN discriminator가, low-frequency correctness를 force하기 위해선 L1 term에 의존하면서, high-frequency structure를 model하는 데만 제한하도록 하면 된다. High-frequencies를 model하기 위해, 우리는 local image patches에서의 structure에 주의를 집중하는 것으로 충분하다. 그러므로 우리는 discriminator architecture를 설계한다-우리는 이를 PatchGAN이라 부른다-이는 scale of patches에서의 structure만을 penalize한다. 이 discriminator는 이미지 내의 각각의  $N \times N$  patch가 진짜인지, 가짜인지 구별하려고 한다. 우리는 이 disrcriminator를 convolutionally, 전체 이미지에 걸쳐 작동시킨다. 그리고 모든 responses를 평균내어 D의 최종 아웃풋을 만든다. (Patch를 전 이미지 영역에 걸쳐서 convolutionally하게 돌려서 이미지의 각 patch 영역에서의

<sup>3</sup> 이미지의 high frequency는 이미지 픽셀이 자주 변한다는 뜻으로, high frequency 영역은 경계부분이나 객체의 모서리 부분이다. Low frequency는 이미지 픽셀값이 잘 안변한다는 것으로, 객체의 내부처럼 픽셀값 변화가 많이 없는 곳이다.

response을 만들어낸다. 그리고 각 patch 영역의 response, 즉 이미지의 이 patch영역이 진짜인지 가짜인지를 판별한 정보들을 평균내어 이 이미지가 진짜인지 가짜인지를 구별한다.)

Section 4.4 (From PixelGANs to PatchGANs to ImageGANs)에서 우리는 N이 전체 이미지 사이즈보다 훨씬 작아져도 여전히 고품질의 결과를 내놓는 것을 보인다. 이는 큰 장점인데, 왜냐하면 더 작은 PatchGAN이 더 적은 parameters를 가지고 더 빠르게 작동하며 임의의 큰 이미지들에도 적용될 수 있기 때문이다.

그러한 discriminator는, 이미지 픽셀 중에서 Patch diameter(Patch 간격)보다 더 멀리 떨어져있는 픽셀 간에 independence를 가정하여 이미지를 Markov random field로써 효과적으로 모델링한다.<sup>4</sup> 이러한 연결관계는 이전에 [38] (<https://arxiv.org/abs/1604.04382>)에서 탐구된 적 있다. 그리고 이러한 연결관계는 또한 texture[17, 21]와 style[16, 25, 22, 37]의 models에선 common assumption이다. 그러므로 우리의 PatchGAN은 texture/style loss의 형태로 이해할 수 있다.

### 13. Optimization and inference

우리의 신경망을 optimize하기 위해, 우리는 [24] (GAN 논문)으로부터의 standard approach를 따른다: 우리는 D에서 한 번의 gradient descent step과 G에서의 한 번의 gradient step을 번갈아 수행한다. Original GAN 논문에서 제시되었듯이, G가  $\log(1 - D(x, G(x, z)))$ 를 최소화하도록 훈련시키기보다, G가  $\log(x, G(x, z))$ 를 최대화하도록 훈련한다. (원 논문의 설명에 따르면, G를 훈련할 때, G가  $\log(1 - D(G(z)))$ 를 최소화하도록 훈련시키는 것이 아니라 G가  $\log D(G(z))$ 를 최대화하도록 훈련시킨다. 왜냐하면 어차피  $\log(1 - D(G(z)))$ 를 최소화하도록 훈련시키나, G가  $\log D(G(z))$ 를 최대화하도록 훈련시키나, 결과적으로 훈련이 완전히 끝나서 더 이상 G와 D가 움직이지 않는 시점, 즉 fixed point of the dynamics of G and D에서, G와 D의 fixed point는 같게 된다. 그런데 훈련의 초기 시점에서, G가 poor result를 만들어 내서 D가 real과 fake를 잘 구분할 수 있는 시점에,  $D(G(z))$ 가 0에 가까운데 이 때  $\log(x)$  함수의 경우  $\log(1)$  보다  $\log(0)$ 지점에서 기울기가 훨씬 크기 때문에 충분한 gradient를 역전파하여 학습이 더 빨리 이루어질 수 있기 때문이다.) 게다가, 우리는 D를 optimize할 때 목적함수(objective)를 2로 나눠서 D의 학습속도를 G의 학습속도에 비해서 느리게 만든다. 우리는 minibatch SGD를 사용한다. 그리고 learning rate가 0.0002이고 momentum parameters가  $\beta_1 = 0.5, \beta_2 = 0.999$ 인 Adam solver[32]를 적용한다.

Inference time에, 우리는 generator net을 정확히 training phase에서와 같은 방식으로 실행시킨다. 이 방법은, 우리가 test time에 dropout을 적용하고, training batch를 종합한 통계가 아닌 test batch의 통계를 이용하여 batch normalization[29]을 사용한다는 점에서 통상적인 방법(usual

---

<sup>4</sup> Markov random field라고 가정한다는 것은 특정 픽셀이 어떠한 값을 가질 확률이, 이웃한 픽셀들의 값에 의해서에만 정해진다는 뜻. 상세 내용은

(<https://iskim3068.tistory.com/58>), (<https://wordbe.tistory.com/entry/PGM-part2-Markov-Random-Field>), 혹은 유튜브 강의 참고.

protocol)과 차이가 있다. Batch size가 1로 정해졌을 때 이러한 접근방식은 “instance normalization”이라고 불린다. 그리고 이 instance normalization은 image generation task에서 효과적이라고 보여졌다. [54]. 우리의 실험에선, 실험에 따라서 1~10 사이의 batch size들을 사용한다.

## 14. Experiments

cGANs의 generality를 탐구하기 위해, 우리는 photo generation과 같은 graphics tasks와, semantic segmentation과 같은 vision tasks 둘 다를 포함하여, 다양한 tasks와 datasets에 대한 방법들을 실험한다.

- *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].

이 datasets의 각각에 대한 training의 세부사항들은 supplemental materials online에 제공되어 있다. 모든 경우에서, input과 output은 단지 1-3 channel image이다. Qualitative results는 Figure 8,9,10,11,13,14,15,16,17,18,19,20에 나와있다. 몇몇 failure cases는 Figure 21에 highlight 되어있다. 더 포괄적인 결과들은 <https://phillipi.github.io/pix2pix/>에서 찾아볼 수 있다.

## Data requirements and speed

우리는 심지어 작은 datasets으로부터도, decent result(괜찮은 결과)를 얻을 수 있음을 알아냈다. 우리의 facade training set은 400개의 이미지만 가지고 있었다.(Figure 14를 보라) 그리고 day to night training set은 오직 91개의 unique webcams (Figure 15를 보라)로 구성되어 있었다. 이러한 size의 datasets에서, 학습은 정말 빠르게 이루어질 수 있다: 예를 들어 Figure 14에서 보여진 결과는 한 개의 Pascal Titan X GPU를 가지고 두 시간 이내로 시간으로 결과를 만들어낸 것이다. Test time에선 모든 model들이 이 GPU를 가지고 일 초 이내에 실행된다.

## 15. Evaluation metrics

Synthesized image(합성된 이미지, 신경망의 출력 이미지)의 quality를 평가하는 것은 아직 해결

되지 않은 어려운 문제이다[52]. per-pixel mean-squared error와 같은 전통적인 metrics(측정방식)은 결과의 joint statistics를 평가하지 않는다. 그래서 structured losses가 capture하는 것을 목표로 하는 very structure(joint statistics)를 측정하지 않는다.

더 총체적으로 우리의 visual quality를 평가하기 위해, 우리는 두개의 tactics를 사용한다. 첫째로, 우리는 “real vs fake” perceptual studies on Amazon Mechanical Turk(AMT)를 수행한다. Colorization과 photo generation과 같은 graphics problems에서, human observer에게의 그럴듯해 보이는 정도가 최종 목표가 되는 경우가 많다. 그러므로 우리는 이러한 접근방법을 사용하여 우리의 map generation, aerial photo generation, 그리고 image colorization을 시험한다.

둘째로, 우리는 off-the-shelf recognition system(아직 발매되지 않는 인식 시스템)이 우리의 synthesized cityscapes 속의 objects를 recognize할 수 있을만큼 우리의 synthesized cityscapes이 사실적인지 측정할 것이다. 이 metric은 [52]의 “inception score”, 그리고 [55]의 object detection evaluation, 그리고 [62]와 [42]에서의 “semantic interpretability” 측정(measures)과 비슷한 방식이다.

### AMT perceptual studies

우리의 AMT 실험들을 위해, 우리는 [62]의 protocol을 따른다: Turkers는 우리의 알고리즘에 의해 생성된 가짜 이미지에 반하는(against) 실제 이미지에 구멍을 남기는 일련의 trial들을 받았다.(원문: Turkers were presented with a series of trials that pitted a real image against a fake image generated by our algorithm.) 각각의 시행(trial)에서, 각각의 이미지는 1초 동안 나타났다. 그 후 이미지들은 사라지고, Turkers는 어떤 것이 가짜였는지를 판단할 수 있는 시간이 제한되지 않게 주어졌다. Turkers는 각 세션의 처음 10개의 이미지들로 연습하였고, 연습 후에 feedback을 받았다. Main experiment의 40번의 trials에선 feedback이 주어지지 않았다. 각 세션은 한 번에 한 개의 알고리즘을 시험했다. 그리고 Turkers가 한 개 이상의 session을 완료하는 것은 허용되지 않았다. ~50명의 Turkers는 각각의 알고리즘을 평가했다. [62]와 달리, 우리는 vigilance trials을 포함하지 않았다. 우리의 colorization experiments를 위해서, 실제와 가짜 이미지들은 같은 grayscale input으로 생성되었다. Map<->aerial photo에서, 실제와 가짜 이미지들은 같은 input으로부터 생성되지 않았다. Task를 더 어렵게 만들고, floor-level results를 피하기 위함이다. Map-aerial photo에서, 우리는 256 X 256 해상도의 이미지를 학습했다. 그러나 fully-convolutional translation을 이용(exploit) 하여 512 X 512 images에서 시험(test)하였다.(학습할 땐 256 X 256 image를 이용했지만, test 시에, 즉 model을 run 하여 output을 만들 때는 512 X 512 image를 만들어냈다는 뜻 같음.) 그리고 이 512 X 512 images는 이후에 256 X 256 해상도로 downsample되어 Turkers에게 보여졌다. (원문: For map<->aerial photo, we trained on 256 X 256 resolution images, but exploited fully-convolutional(described above) to test on 512 X 512 images, which were than downsampled and presented to Turkers at 256 X 256 resolution.) Colorization에서, 우리는 256 X 256 resolution images에서 train하고 test했다. 그리고 이 256 X 256의 동일한(same)해상도로 results를 Turkers에게 보여주었다.

"FCN-score"

Generative model들의 양적인(quantitative) 평가(evaluation)는 어렵다고 알려져 있지만, 최근의 작업들[52, 55, 62, 42]은 pre-trained semantic classifiers를 이용하여 generated stimuli의 discriminability(식별 가능성)을 pseudo-metric으로써 측정하는 것을 시도했다. Intuition은 다음과 같다: 만약 generated images가 사실적이라면, real images로 학습한 classifiers는 합성된 이미지 또한 올바르게 classify할 수 있을 것이다. 이를 위해, 우리는 semantic segmentation을 위해 유명한 FCN-8s architecture<sup>5</sup>를 채택하고, 이를 cityscapes dataset으로 훈련하였다. 우리는 그리고나서 classification accuracy against the labels these photos were synthesized from을 이용하여 synthesized photos의 점수를 매겼다.

## 16. Analysis of the objective function

Equation 4 ( $G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$ . (4))에 나와있는 목적함수의 어떤 components가 중요한가? 우리는 ablation studies<sup>6</sup>를 수행하여 L1 term, GAN term의 효과를 isolate시켰고, using discriminator conditioned on the input (cGAN, equation1) against using an unconditional discriminator(GAN, equation2)를 비교했다.



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

<sup>5</sup> <https://medium.com/@msmapark2/fcn-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-fully-convolutional-networks-for-semantic-segmentation-81f016d76204> 참고

<sup>6</sup> Ablation study란, 제안한 요소가 모델에 어떠한 영향을 미치는지 확인하고 싶을 때, 이 요소를 포함한 모델과 포함하지 않은 모델을 비교하는 것을 말한다. 이는 딥러닝 연구에서 매우 중요한 의미를 지니는데, 시스템의 인과관계(causality)를 간단히 알아볼 수 있기 때문이다. (<https://fintecuriosity-11.tistory.com/73> 참고)

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	<b>0.66</b>	<b>0.23</b>	<b>0.17</b>
<b>Ground truth</b>	0.80	0.26	0.21

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)	<b>0.55</b>	<b>0.20</b>	<b>0.14</b>

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	<b>0.17</b>
70×70	<b>0.66</b>	<b>0.23</b>	<b>0.17</b>
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 4는 두 개의 labels-photo problems에 대한 이러한 variations의 qualitative effects를 보여준다. L1만으로도 blurry하지만 reasonable한 results를 이끌어낸다. cGAN만 사용해서(equation 4에서 lambda=0 으로 설정하면됨) 훨씬 sharper한 결과를 내놓는다. 하지만 몇몇 applications(아마 몇몇 결과 이미지들을 말하는 것 같음)에 visual artifacts를 introduces하였다.(figure 4에서 cGAN만 사용한 결과 이미지들에는 사진과 전혀 어울리지 않는 알 수 없는 물체들 몇 개가 이미지상에서 나타난다는 것을 말하는 것 같음.) cGAN term과 L1 term 둘을 더하는 것은(둘 다를 사용, 그리고

$\lambda=100$ 으로 설정) 이러한 artifacts를 줄인다.

우리는 이러한 관찰결과(observations)를 cityscapes labels->photo task에 대한 FCN-score를 사용하여 quantify하였다.(Table 1): GAN-based objectives(목적함수)가 더 높은 점수들을 성취했다. 이는 the synthesized images(the가 붙어있으므로, GAN-based objectives로 생성한 이미지를 뜻하는듯)가 더 식별가능한(recognizable) 구조를 가지고 있다는 것을 보여준다. 우리는 또한 discriminator로부터 conditioning을 제거하는 것(labeled as GAN, 즉 cGAN이 아닌 GAN을 사용했고, 이는 테이블에서 GAN이라고 label되어 있다.)의 효과를 test한다. 이 경우에, loss는 input과 output 사이의 mismatch를 penalize하지 않는다; 오직 output이 realistic인지 아닌지만 신경쓴다. 이 변형은 (variant) 형편없는 performance의 결과를 내놓았다. Results를 examining해보니, generator가 collapse(GAN에서의 모드붕괴 같은것인듯) 하여 input photograph에 상관없이 거의 같은 output을 만들게 되었다는 것이 밝혀졌다. 확실히, 이 경우에 the loss가 input과 output 사이의 quality of match를 측정해야 한다는 사실이 중요하다. 그리고 실제로 cGAN이 GAN보다 훨씬 잘 동작한다.(원문: 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.) 그러나, L1 term을 adding하는 것 또한, L1 loss가, ground truth output(정확히 input과 match함)과, synthesized output(정확히 input과 match하지 않을 수 있음)사이의 distance를 penalize하기 때문에 output이 input을 respect(중히 여기다, 즉 input과 output이 match되도록 한다는 뜻인것같음)한다는 것에 주목하라. 따라서, L1+GAN 또한 input label maps을 respect하는 realistic renderings를 만드는 데 효과적이다. 모든 terms을 합친 L1+cGAN 또한 비슷하게 잘 작동한다.

### Colorfulness

Conditional GANs의 striking effect는 이 신경망이, 심지어 input label map에 존재하지 않는 spatial structure도 hallucinating하여 sharp images(선명한 사진)를 만들어낸다는 것이다. 혹자는 cGANs이 spectral dimension에서 "sharpening"과 비슷한 효과를 가지고 있다고 생각할지도 모른다 -즉 이미지들을 더 colorful하게 만든다고 생각할지도 모른다. L1이 정확히 어디에 edge를 위치시켜야 할 지 uncertain한 상황에서 blur를 incentivize할 것처럼, L1은 또한 a pixel이 몇몇 타당한 color values중 무엇을 선택해야할지 모를 때, 평균, grayish color 를 incentivize할 것이다.(원문: 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, L1은 possible colors에 대한 conditional probability density function의 median을 선택함으로써 최소화될 것이다. 반면 Adversarial loss는 이론적으로 grayish outputs이 unrealistic하다는 것을 알아차릴 수 있다. 그래서 true color distribution을 encourage한다 [24]. Figure7에서, 우리의 cGANs이 실제로 Cityscapes dataset상에서 이 효과를 달성하는지 안하는지를 조사한다. The plots은 marginal distributions over output color values in Lab color space를 보여준다. The ground truth distributions은 dotted line으로 표시되어있다. L1이 ground truth보다 narrower distribution으로 lead한다는 것이 드러난다. 그리고 이는 L1이 평균, 즉 grayish colors를

encourages한다는 가설을 확인시켜준다.



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.

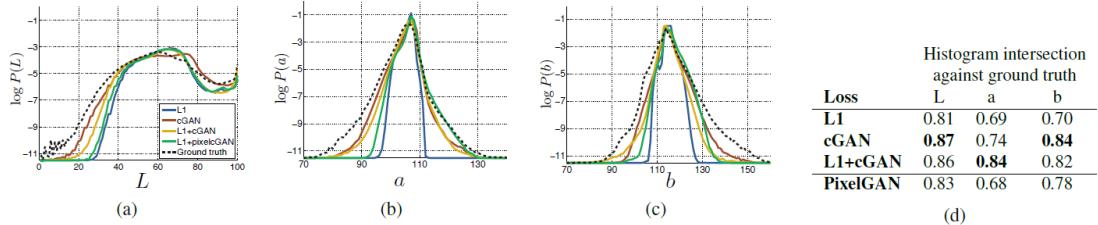


Figure 7: Color distribution matching property of the cGAN, tested on Cityscapes. (c.f. Figure 1 of the original GAN paper [24]). Note that the histogram intersection scores are dominated by differences in the high probability region, which are imperceptible in the plots, which show log probability and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

## 17. Analysis of the generator architecture

U-Net architecture는 low-level의 network를 가로질러 shortcut하게 해준다. 이것이 better results를 불러오는가? Figure 5와 Table 2는 cityscape generation에서 U-Net과 encoder-decoder를 비교한다. Encoder-decoder는 U-Net에서 단지 skip connections을 severe(끊다)함으로써 만들어진다. 우리의 실험에서, Encoder-decoder는 realistic images를 생성하도록 학습할 수 없다. U-Net의 advantages는 conditional GANs에 국한되는 것 같지는 않다(appear not to be specific to conditional GANs): U-Net과 encoder-decoder 둘 다 L1 loss를 이용하여 훈련될 때, U-Net이 다시 더 뛰어난 결과를 성취했다.(Table 2에서, (L1+cGAN)에서도 encoder-decoder보다 U-net을 사용하는 것이 효과적이었지만, (L1)에서도 encoder-decoder보다 U-net을 사용하는 것이 더 효과적이었다.)

## 18. From PixelGANs to PatchGANs to ImageGANs

우리는 우리의 discriminator receptive fields의 patch size N을, 1 X 1의 "PixelGAN"으로부터 full 286 X 286 "Image GAN"<sup>7</sup>까지 다양화하는 것의 효과를 시험해보았다. Figure 6은 이 분석에 대한 qualitative results를 보여주고 있고, Table 3는 FCN-scores를 이용하여 the effects를 quantifies한다. 이 논문의 다른 곳에서 specified되지 않는 한 모든 실험은 70 X 70 PatchGANs를 사용하고, 이

<sup>7</sup> 원 논문의 각주1 : 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 supplemental materials online.

section에선, specified되지 않는 한 모든 실험들이 L1+cGAN loss를 사용함을 note하라.

The PixelGAN은 spatial sharpness에 효과가 없었지만 results의 colorfulness를 증가시켰다(Figure 7에 quantified되어있다.) (이부분은 잘 이해가 안감..) 예를 들어, Figure 6에서의 버스는, 신경망이 L1 loss를 가지고 훈련되었을 때 회색으로 칠해져있다, 하지만 PixelGAN loss로 훈련되었을 땐 버스가 빨간색이 되어있다. Color histogram matching은 image processing에 있어서 공통의 문제이다[49], 그리고 PixelGANs은 유망한 lightweight solution이 될 수 있다.

16 X 16 PatchGAN을 사용하는 것은 sharp output을 향상시키기에 충분하여 좋은 FCN-scores를 성취한다, 그러나 또한 tiling artifacts를 lead한다.( Figure 6에서 16x16 을 사용했을 때 1 x 1 보다 선명한 이미지가 나왔지만, 네모난 딱지 타일들이 여기저기 붙어있는 것처럼 보인다.) 70 x 70 PatchGAN은 이 artifacts들을 완화(alleviate)시킨다. 그리고 약간 더 나은 점수를 기록했다. 이것보다도 더 크게 해서, 완전한 286 x 286 Image GAN은, (원문: Scaling beyond this, to the full 286 x 286 imageGAN)결과의 visual quality를 향상시키는 것처럼 보이지 않고, 사실 상당히 더 낮은 FCN score를 기록했다(Table 3). 이는 The Image GAN(Image GAN은 full patch size를 사용하는 GAN)은 더 많은 parameters를 가지고 있고, 70 x 70 PatchGAN보다 greater depth를 가지고 있기 때문에 train하기 더 힘들 수 있기 때문이다.

#### Fully-convolutional translation

PatchGAN의 장점은 fixed-size patch discriminator가 임의의 large image에도 적용될 수 있다는 것이다. 우리는 또한 generator를, generator가 학습된 것 보다도 더 큰 image들에 convolutionally apply할 수 있다. 우리는 이것을 map<->aerial photo task에서 시험한다. Generator를 256 X 256 images에서 훈련한 이후, 우리는 이를 512 X 512 images에서 시험한다. Figure 8에 있는 results가 이러한 접근방식의 효율성을 보여준다.



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

## 19. Perceptual validation

우리는 map $\leftrightarrow$ aerial photograph과 grayscale $\rightarrow$ color task에 대한 우리 results의 perceptual realism을 검증한다. map $\leftrightarrow$ photo에 대한 우리의 AMT실험 결과는 Table 4에 주어져 있다. 우리의 방법으로 생성한 Aerial photos는 18.9%의 시행에서 참가자들을 속였는데, 이는 blurry results를 만들어내고 거의 참가자들을 속이지 못했던 L1 baseline(기준)보다도 의미있는 정도로 더 위이다. 반면 photo $\rightarrow$ map direction에서, 우리의 방법은 6.1%의 시행에서만 참가자들을 속였다. 그리고 이는 L1 baseline performance보다도 그렇게 크게 차이나지 않는 것이다.(based on bootstrap test). 이는 좀 더 chaotic한 aerial photographs보다도 rigid geometry를 가지는 지도에서 minor structural errors가 더 잘 눈에 띄기 때문일 수 있다.

우리는 ImageNet[51]에서 colorization을 훈련했고, [62, 35]에서 소개된 test split에서 test했다. 우리의 방법은, L1+cGAN loss로, 22.5%의 trials에서 참가자들을 속였다(Table 5). 우리는 또한 [62]의 결과를 시험하고, L2 loss를 사용하는 그들 방식의 variant(see [62] for details)를 시험하였다. Conditional GAN은 [62]의 L2 variant와 비슷한 점수를 기록했다 (difference insignificant by bootstrap test). 그러나 우리의 실험에서, 27.8%의 시행에서 참가자들을 속였던 [62]의 full method에 비해선 약간 낮았다. 우리는 그들의 방법이 colorization에서 잘 작동하도록 specifically engineered라는 데 주목했다.(이 논문은 general purpose image-to-image translation에 다루고 있으므로 colorization에선, colorization만을 다루는 신경망에 비해 좀 낮은 수치를 기록할 수 있음을 말하는 것.)

## 20. Semantic segmentation

Conditional GANs는, 흔히 image processing과 graphics tasks에서 그려듯이, output이 highly

detailed or photographic problems에서 효과적으로 보인다. Output이 input보다도 덜 complex한 semantic segmentation과 같은 vision problems에 관해선 어떨까?

Loss	Photo → Map	Map → Photo
	% Turkers labeled <i>real</i>	% Turkers labeled <i>real</i>
L1	$2.8\% \pm 1.0\%$	$0.8\% \pm 0.3\%$
L1+cGAN	$6.1\% \pm 1.3\%$	$18.9\% \pm 2.5\%$

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

Method	% Turkers labeled <i>real</i>
L2 regression from [62]	$16.3\% \pm 2.4\%$
Zhang et al. 2016 [62]	$27.8\% \pm 2.7\%$
Ours	$22.5\% \pm 1.6\%$

Table 5: AMT “real vs fake” test on colorization.



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).

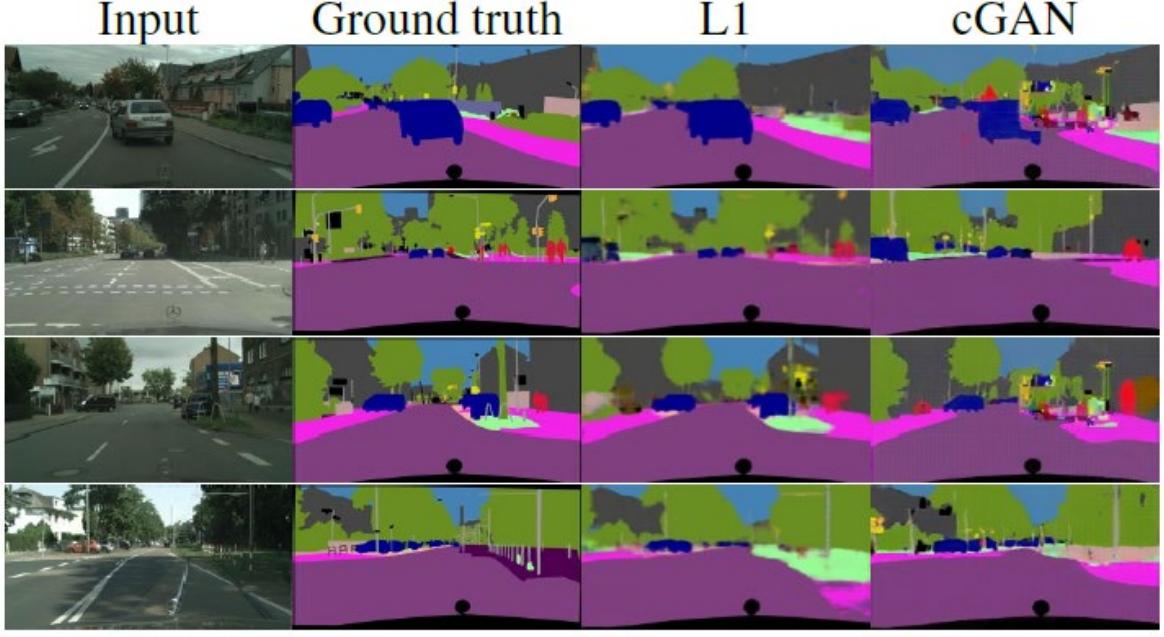


Figure 10: Applying a conditional GAN to semantic segmentation. The cGAN produces sharp images that look at glance like the ground truth, but in fact include many small, hallucinated objects.

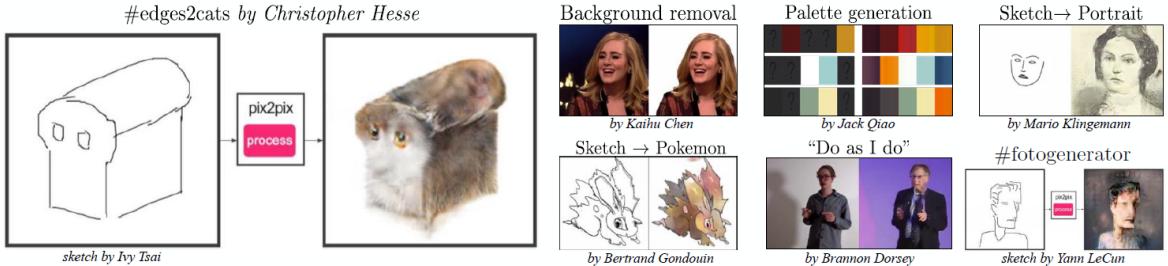


Figure 11: Example applications developed by online community based on our pix2pix codebase: #edges2cats [3] by Christopher Hesse, Background removal [6] by Kaihu Chen, Palette generation [5] by Jack Qiao, Sketch → Portrait [7] by Mario Klingemann, Sketch→Pokemon [1] by Bertrand Gondouin, “Do As I Do”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

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

Table 6: Performance of photo→labels on cityscapes.



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.

the pix2pix code we released. Nonetheless, they demonstrate the promise of our approach as a generic commodity tool for image-to-image translation problems.

이 test를 시작하기 위해 우리는 cityscape photo->labels에서 cGAN(with/without L1 loss)을 train 한다. Figure10이 qualitative results를 보여준다. 그리고 Table 6에 quantitative classification accuracies가 나타나있다. 흥미롭게도, L1 loss없이 훈련된 cGANs가 reasonable degree of accuracy로 이 문제를 풀었다. 우리가 아는 바로는, 이것이 GANs의 continuous-valued variation<sup>8</sup>을 가진 images가 아닌, nearly discrete한 value variation을 "labels"를 성공적으로 생성해 낸 첫 번째 demonstration이다. 비록 cGANs가 몇몇 성공들을 성취했지만, 그들은 이 문제를 푸는 최선의 방법과는 거리가 멀다: Table 6에 나와있듯이, cGAN을 사용하는 것보다 단순히 L1 regression을 사용하는 것이 더 나은 점수를 얻었다. 우리는 vision problem에 대해, 목표(i.e. ground truth)에 가까운 output을 예측하는 것)가 graphics tasks보다 덜 ambiguous하고, 그래서 L1과 같은 reconstruction losses<sup>9</sup>가 mostly sufficient하다고 주장한다.

---

<sup>8</sup> 원문 각주 2: Note that 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.

<sup>9</sup> 원래 Reconstruction loss는, 아웃풋이 입력과 얼마나 유사한지를 나타내는 loss이다. L1 loss는 식 (3)과 식 (4)에서 보듯이 아웃풋과 target이 얼마나 유사한지를 나타낼 수 있는 loss기법임. L1이 왜 reconstruction loss인가? 어쨌든 맥락 이해에는 문제없다.

## 21. Community-driven Research

이 논문과 우리의 pix2pix codebase가 처음 release된 이후로, visual artists뿐만 아니라 computer vision과 graphics practitioners들을 포함한 Twitter community가, 성공적으로 우리의 framework를, 원 논문의 scope를 훨씬 뛰어넘는 기발한 image-to-image translation tasks에 적용하였다. Figure 11과 Figure 12는 특히 인기있는 #edges2cats and #fotogenerator 뿐만 아니라 Background removal, Palette generation, Sketch->Portrait, Sketch->Pokemon, "Do as I Do" pose transfer, Learning to see: Gloomy Sunday, 을 포함하는 #pix2pix hashtag로부터 가져온 몇 가지 예시를 보여준다. 이 application들이, creative projects이고, 통제된, scientific 조건 하에서 얻어진 것이 아니라는 것, 그리고 우리가 release한 pix2pix code의 some modifications에 rely on 할 수 있다는 점을 note하라. 그럼에도 불구하고, 그들은 우리의 image-to-image translation problems에 대한 generic commodity tool로써의 우리의 접근방식에 대한 유망함을 보여주었다.

## 22. Conclusion

논문의 결과는 conditional adversarial network가 다양한 image-to-image translation에 대해, 특히 highly structured graphical output에 대해 유망한 접근법임을 보인다. 이 신경망들은 현재 당면한 task와 데이터에 알맞은 loss를 학습하는데, 이로 인해 다양한 환경에서도 작동할 수 있다.

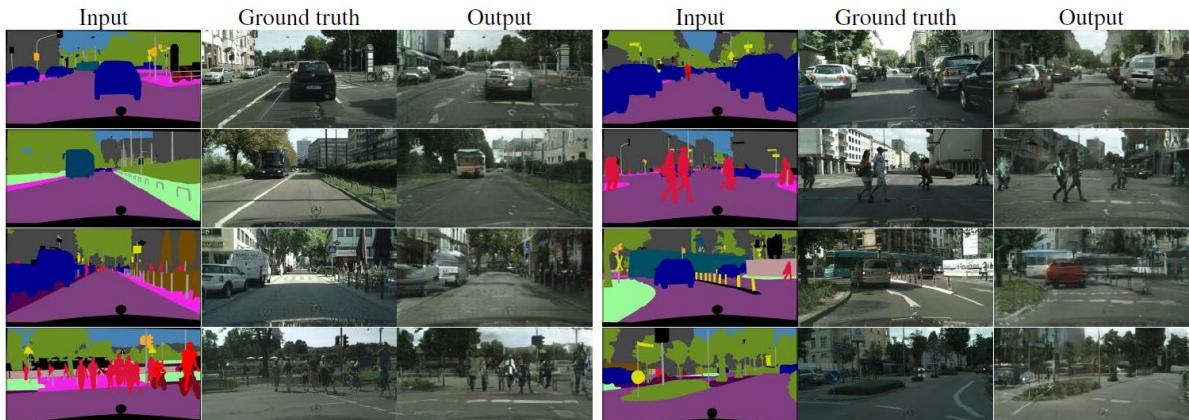


Figure 13: Example results of our method on Cityscapes labels→photo, compared to ground truth.

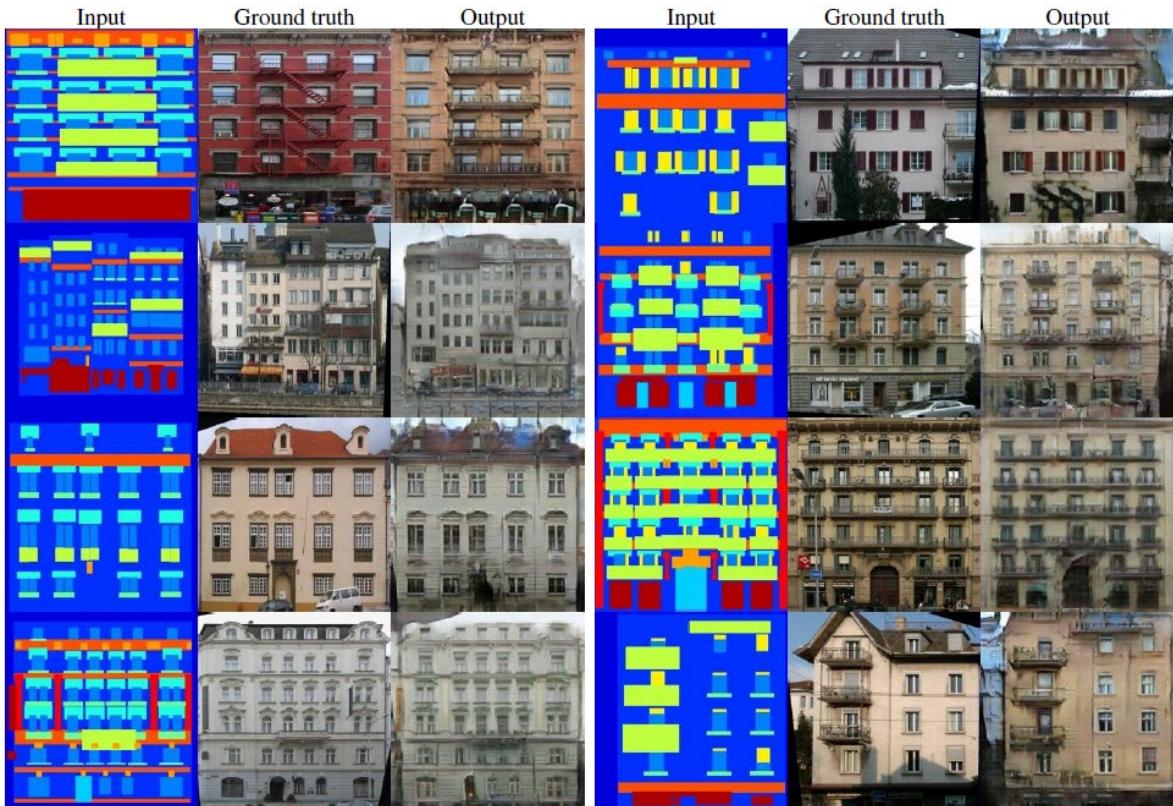


Figure 14: Example results of our method on facades labels→photo, compared to ground truth.



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

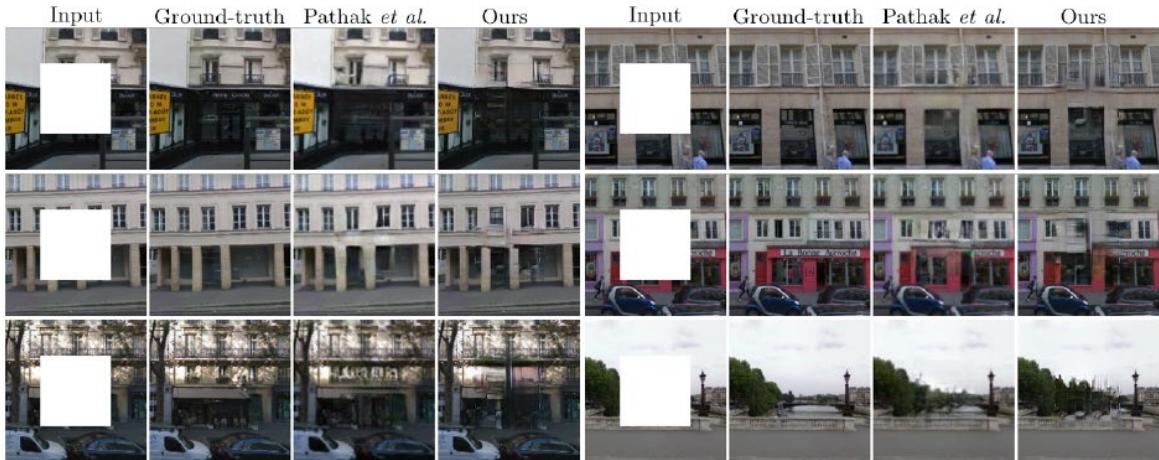


Figure 19: Example results on photo inpainting, compared to [43], on the Paris StreetView dataset [14]. This experiment demonstrates that the U-net architecture can be effective even when the predicted pixels are not geometrically aligned with the information in the input – the information used to fill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

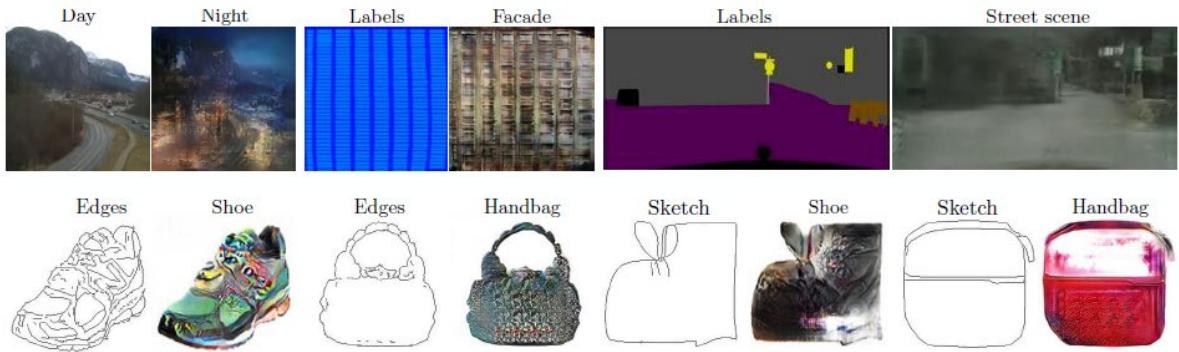


Figure 21: Example failure cases. Each pair of images shows input on the left and output on the right. These examples are selected as some of the worst results on our tasks. Common failures include artifacts in regions where the input image is sparse, and difficulty in handling unusual inputs. Please 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
- [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

- [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. 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
- [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
- [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
- [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

- [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

## 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  $C_k$  denote a Convolution-BatchNorm-ReLU layer with  $k$  filters.  $CD_k$  denotes a Convolution-BatchNorm-Dropout-ReLU layer with a dropout rate of 50%. All convolutions are  $4 \times 4$  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, Batch-Norm 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:

$286 \times 286$  discriminator:

$C64-C128-C256-C512-C512-C512$

### 6.2. Training details

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

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 and around New York City. Data were

`decoder.`

**U-Net decoder:**

CD512–CD1024–CD1024–C1024–C1024–C512  
–C256–C128

### 6.1.2 Discriminator architectures

The  $70 \times 70$  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:

**$1 \times 1$  discriminator:**

C64–C128 (note, in this special case, all convolutions are  $1 \times 1$  spatial filters)

**$16 \times 16$  discriminator:**

C64–C128

**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 unnecessary downsampling). FCN-scores computed using scripts at [https://github.com/phillipi/pix2pix/tree/master/scripts/eval\\_cityscapes](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.

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.