Imagination to Image-nation

Based on The Paper:

Image-to-Image Translation with Conditional Adversarial Networks

By: Ofek Bernstein

Faculty of Data and Decision Sciences, Technion, Israel

Introduction and Backround

This project delves into the 2017 research paper *Image-to-Image Translation with Conditional Adversarial Networks* by a team from UC Berkeley. It highlights how conditional GANs (cGANs) serve as a versatile solution for image-to-image translation tasks, replacing the need for task-specific architectures or loss functions. The paper's method proves effective across a variety of domains—ranging from sketch-to-photo to colorization—using a single model architecture.

The idea of translating images from one representation to another has been explored in computer vision for decades, often requiring handcrafted algorithms tailored to specific tasks. However, in 2017, the paper

introduced a unified approach using cGANs.

This framework allows a single model to learn mappings for various tasks such as colorization, segmentation, and style transfer without task-specific adjustments.

When first introduced, many image transformation tasks relied on per-pixel regression models, which often resulted in blurry outputs. By leveraging adversarial training, cGANs address this issue by learning a loss function that encourages the generation of realistic and detailed images. This approach enables models to synthesize sharp, high-quality results across multiple domains.

cGANs have since become a key technique in image synthesis and enhancement, demonstrating versatility across numerous applications in computer vision.

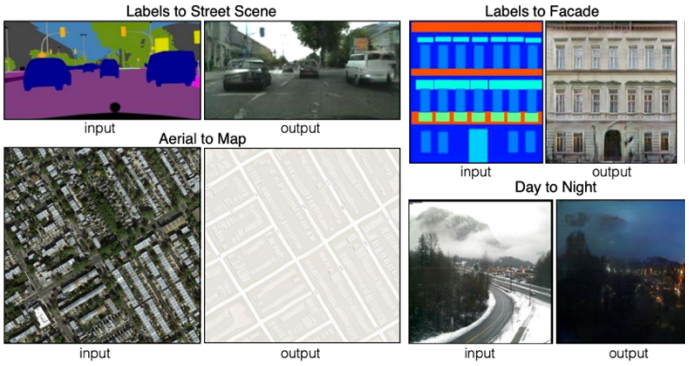


Figure 1: Example Image-to-Image Translation Tasks

Paper Summarization

The main topic of the paper

The paper *Image-to-Image Translation with Conditional Adversarial Networks* introduces conditional generative adversarial networks (cGANs) as a general framework for image-to-image translation tasks. The core idea is that many computer vision problems, such as converting grayscale images to color, generating realistic photos from sketches, and transforming daytime images into nighttime scenes, can be formulated as a mapping from one image domain to another.

Traditional approaches rely on task-specific algorithms and handcrafted loss functions. However, this paper proposes a unified approach where the network learns both the mapping function and the loss function using adversarial training. By framing image translation as a conditional GAN problem, the model generates realistic outputs conditioned on the input image, outperforming traditional pixel-wise regression methods.

Challenges Addressed and Proposed Techniques

A major challenge in image-to-image translation lies in defining a loss function that leads to visually convincing results. Per-pixel losses like L1 or L2, while simple and widely used, often produce blurry outputs—especially in scenarios where multiple correct answers exist. The authors address this by introducing adversarial loss, leveraging a discriminator to guide the generator toward producing outputs that are not just close to the ground truth, but indistinguishable from real images. This shift encourages the generation of sharper and more realistic visuals.

Another challenge is maintaining fine-grained structure in the translated images. When changing high-level properties like color, texture, or style, it's critical that the model doesn't lose spatial accuracy—like the boundaries of objects or architectural lines. To preserve such details, the paper employs a U-Net architecture. With skip connections that carry low-level information directly from the encoder to the decoder, U-Net enables the model to retain structural integrity throughout the transformation process.

Global consistency poses a third difficulty. In tasks like turning maps into satellite imagery, it's not enough for parts of the image to look realistic in isolation; the image must also make sense as a whole. To address this, the authors propose the use of a PatchGAN discriminator. Rather than evaluating the entire image, PatchGAN operates on small local patches, enforcing realism at the texture and pattern level. When applied across the image, this approach helps maintain both local detail and overall coherence.

Finally, the paper shows that its proposed framework isn't narrowly tailored to any one task. Instead, a single cGAN-based architecture generalizes effectively across a wide range of image synthesis problems—from colorization to label-to-image generation. This level of versatility not only simplifies implementation but also suggests that cGANs offer a scalable, task-agnostic solution for many challenges in visual generation.

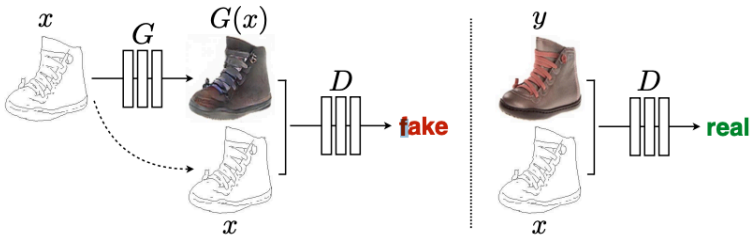


Figure 2: Conditional GAN Training Framework

Main Results of the Paper

The paper evaluates the effectiveness of cGANs on several datasets and tasks, demonstrating significant improvements over traditional methods:

- **High-quality image synthesis:** The model successfully translates images across various domains, including sketches to photos, segmentation maps to real-world scenes, and black and white images to color.
- **Better perceptual quality:** Compared to L1/L2 regression, the adversarial training strategy produces sharper and more realistic results, avoiding the typical blurriness of traditional approaches.
- **Quantitative evaluation:** The authors use FCN-score (evaluating how well a classifier can recognize objects in generated images) and Amazon Mechanical Turk (AMT) perceptual studies to show that their method outperforms baseline models in generating realistic images.
- **Efficient generalization:** Despite using a single architecture and objective function, the model

performs well across diverse datasets, showcasing its flexibility.

Key Theoretical Contributions

While the paper primarily focuses on empirical results, it presents an important theoretical foundation for image-to-image translation using GANs. The key contributions include:

1. **Adversarial Loss for Image Translation**

   The paper formulates the objective function of a conditional GAN as:

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

   where G tries to generate realistic images $G(x, z)$ conditioned on the input x, while D attempts to distinguish real image pairs (x, y) from generated pairs $(x, G(x, z))$.

2. **Combining cGAN Loss with L1 Regularization**

   To prevent mode collapse, the authors propose adding an L1 loss term:

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

   The final objective function balances both terms:

$$G^* = \arg\min_{\mathrm{G}} \max_{\mathrm{D}} \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(\mathrm{G})$$

   This ensures that the generator produces images that are both perceptually

Figure 4: Different Loss Functions Compared

3. **PatchGAN Discriminator**

   Instead of classifying an entire image as real or fake, the discriminator operates on small patches, enforcing local realism. The authors show that a 70×70 PatchGAN strikes a good balance between capturing fine details and ensuring large-scale consistency.

Implementations of Main Results of The Paper

To implement the main ideas from Image-to-Image Translation with Conditional Adversarial Networks, I used an existing open-source implementation of the paper, which replicates the architecture and training methodology proposed by Isola et al.

The original paper introduced a general framework using Conditional GANs (cGANs) for a wide range of image-to-image translation tasks. The implementation I worked with follows this architecture and uses the official pix2pix model structure, which includes:

- A U-Net generator: an encoder-decoder architecture with skip connections, allowing low-level information from the input to be directly passed to the output.
- A PatchGAN discriminator: rather than classifying the whole image as real or fake, this model judges small image patches, enforcing local realism.

Workflow Summary

1. **Environment Setup**

I set up the project using Python with libraries such as PyTorch, NumPy, OpenCV, and Matplotlib. The environment included GPU acceleration for efficient training and testing.

2. **Data Preparation**

I worked with an existing dataset of paired images, where each image consists of a side-by-side layout: the left half being the real image and the right half being the condition (e.g., sketch, edge map, or label). These images were split down the middle to form the input-output pairs required for training and evaluation.

3. **Model Inference**

The trained model was used to generate outputs from test images. Each test image was split into the conditional input and the real target image. The conditional image was passed to the generator to create a translated output.

4. **Visualization**

I visualized the results by displaying the real image (ground truth), the conditional input, and the generated output side-by-side to assess the model's performance qualitatively.

This implementation reproduces the core functionality of the original paper and sets the stage for additional experimentation.

Alteration to The Architecture

To explore the impact of different reconstruction losses on the generator's output quality, I modified the original pix2pix framework by replacing the traditional L1 loss with a Cosine Similarity-based loss. The goal was to evaluate whether this change could yield perceptual or structural improvements in the generated images.

Why Cosine Similarity?

In the original pix2pix implementation, L1 loss penalizes the absolute pixel-wise difference between the generated image and the ground truth. However, this can often lead to blurry outputs, especially in areas where multiple plausible outputs exist.

Cosine Similarity, on the other hand, compares the directional alignment between two vectors rather than their absolute difference. By using this measure over flattened images, the model is encouraged to preserve overall structure and high-level visual consistency, even if the exact pixel values differ.

Implementation Summary

- I used PyTorch's nn.CosineSimilarity function over the flattened batch of generated and real images.
- The cosine similarity is converted into a loss:

$$\mathcal{L}_G = \mathcal{L}_{GAN} + \lambda \cdot \mathcal{L}_{\text{cosine}}$$

Where $\lambda = 100$, same as the original L1 weights

Discussion: Comparing Results with and without Cosine Similarity

To evaluate the impact of replacing the standard L1 loss with Cosine Similarity, I compared both quantitative training behavior and qualitative image generation results. Below is an analysis based on loss curves and output samples from both the original model and the altered version.

1. **Generator Loss Trends**

The generator loss in the original model decreased steadily over time, while the Cosine-based model showed minimal loss reduction, indicating slower or less effective convergence.

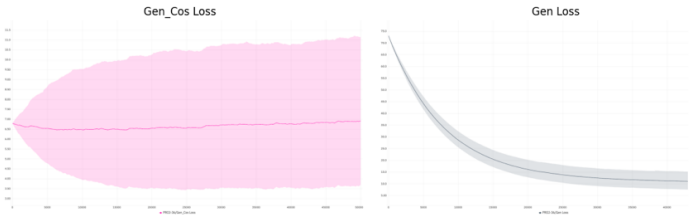
Original Model (L1-based):

The generator loss showed a consistent exponential decline over time, indicating stable convergence. As expected, the combination of L1 and GAN loss

encouraged the model to reduce absolute pixel differences quickly.

Cosine Loss Model:

The generator loss was higher overall and showed a slower rate of improvement, leveling off around a plateau. This behavior is likely due to the nature of Cosine loss, which penalizes angular dissimilarity rather than pixel magnitude — resulting in smaller gradients and slower convergence, especially early in training.

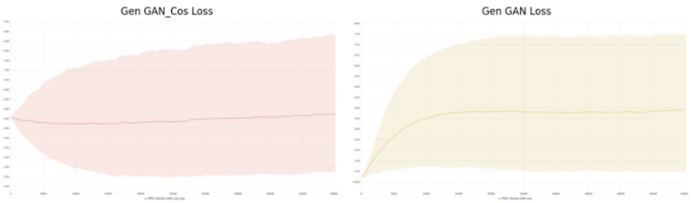


2. Adversarial Loss (GAN Component)

In both versions, the GAN loss (for both generator and discriminator) stabilized after initial fluctuations.

Interestingly, Cosine-based models exhibited slightly higher GAN loss, which could indicate that the generator found it more challenging to fool the discriminator compared to the L1 version.

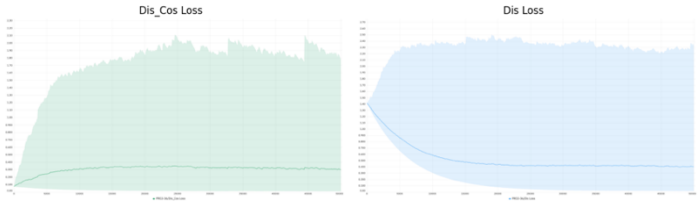
This may suggest that Cosine similarity alone doesn't fully capture the visual realism the discriminator is sensitive to, making adversarial training slightly harder.



3. Discriminator Behavior

The discriminator loss for the original model decreased more rapidly and stayed lower, suggesting confident classification between real and fake samples.

In contrast, the Cosine version's discriminator loss increased in the beginning, then decreased more slowly, likely due to more subtle or inconsistent outputs from the generator — especially early in training — which made the real/fake boundary less clear.

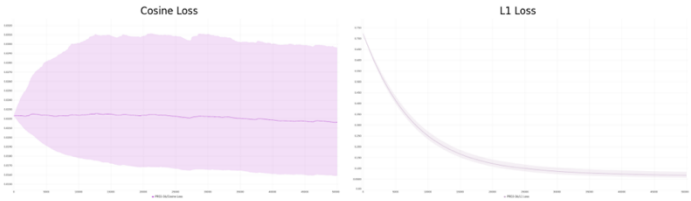


4. Cosine vs. L1 Loss Values

As expected, the Cosine loss stayed relatively low throughout training (around 0.02–0.03), while the L1 loss dropped significantly (from ~0.7 to ~0.05). This

reflects the difference in scale and sensitivity of the two loss types.

Cosine loss values showed less dramatic improvement, but remained stable — suggesting the model maintained directionally consistent features, even if not perfect pixel alignment.



5. Qualitative Results

The L1-based model produced sharper and more color-accurate outputs that closely followed the target label map.

The Cosine-based model preserved overall structure but had more texture noise and color bleeding. In some cases, fine details (like windows and borders) were less distinct.

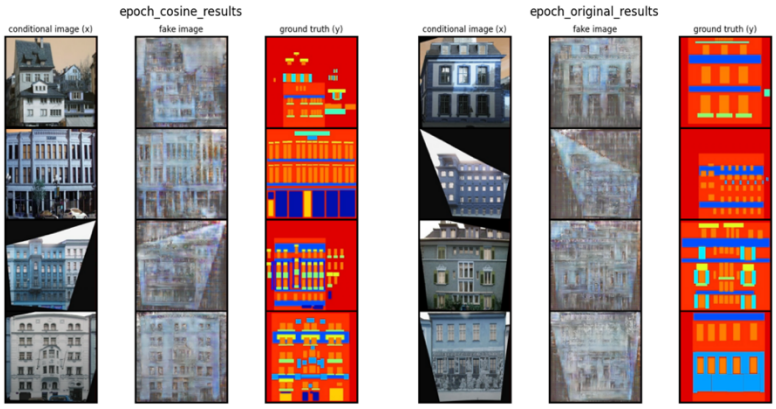
However, the Cosine model occasionally demonstrated greater tolerance to unusual inputs or lighting variations — implying that structure alignment was prioritized over strict pixel copying.

6. Alignment with the Paper's Hypothesis

The original paper argues that pixel-wise losses like L1 or L2 are not enough to produce visually realistic images, as they often result in blurry outputs. To address this, the authors introduced an adversarial loss that encourages sharper, more detailed results by training a discriminator to distinguish real from fake images. Combined with L1 loss, this setup ensures both structural alignment and perceptual quality.

In my experiment, I replaced the L1 loss with Cosine Similarity Loss, which compares the direction of the predicted and ground-truth images rather than their raw pixel values. This change aimed to prioritize global structural consistency over exact intensity matching. The results showed that while Cosine loss helped preserve overall layout and shape, it struggled with texture and color fidelity, leading to outputs that were sometimes noisier or washed out. Additionally, the generator converged more slowly and less clearly when trained with Cosine loss, as the loss remained relatively flat and noisy throughout training, likely due to the weaker and more ambiguous gradients it provides compared to L1.

These findings reinforce the paper's hypothesis: no single loss function is sufficient. The Cosine-based model still required adversarial training to produce meaningful textures, and without a strong pixel- or feature-based loss, fine details were lacking. Overall, the experiment confirms that a combination of losses — one for structure, one for realism — is key to successful image-to-image translation, just as the original paper proposed.



Conclusion

This project revisited the foundational paper *Image-to-Image Translation with Conditional Adversarial Networks* and implemented its main contributions using a standard pix2pix setup. By replacing the traditional L1 loss with Cosine Similarity, I explored how different reconstruction objectives affect learning and output quality. While the Cosine-based model preserved overall structure, it lacked fine detail and converged more slowly — reinforcing the paper's claim that carefully designed loss functions are essential for high-quality image synthesis. These findings highlight the continued importance of balancing perceptual realism and structural accuracy in generative modeling.

References

1. Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). *Image-to-image translation with conditional adversarial networks*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1125–1134.
2. Phillipi, C. (n.d.). *pix2pix official implementation*. GitHub. https://github.com/phillipi/pix2pix
3. Nilesh Barla. *Pix2pix: Key Model Architecture Decisions*. https://neptune.ai/blog/pix2pix-key-model-architecture-decisions
4. Training results were visualized and tracked using Neptune.ai (Run ID: PROJ-36). https://app.neptune.ai/o/ofekber/org/Project/runs/details?viewId=standard-view&detailsTab=metadata&shortId=PROJ-36&type=run