Deep learning

11.3. Conditional GAN and image translation

François Fleuret

https://fleuret.org/dlc/

UNIVERSITÉ DE GENÈVE

All the models we have seen so far model a density in high dimension and provide means to sample according to it, which is useful for synthesis only.

All the models we have seen so far model a density in high dimension and provide means to sample according to it, which is useful for synthesis only.

However, most of the practical applications require the ability to sample a **conditional distribution.** E.g.:

- Next frame prediction.
- "in-painting",
- segmentation,
- style transfer.

This would in particular address some of the shortcomings we saw in lecture 7.3. "Denoising autoencoders".

The Conditional GAN proposed by Mirza and Osindero (2014) consists of parameterizing both $\mathbf{G}$ and $\mathbf{D}$ by a conditioning quantity Y.

$$V(\mathbf{D},\mathbf{G}) = \mathbb{E}_{(X,Y)\sim\mu}\Big[\log \mathbf{D}(X,Y)\Big] + \mathbb{E}_{Z\sim\mathcal{N}(0,I),Y\sim\mu_Y}\Big[\log(1-\mathbf{D}(\mathbf{G}(Z,Y),Y))\Big],$$

To generate MNIST characters, with

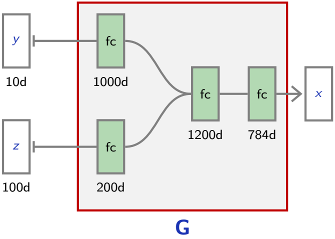
$$Z \sim \mathscr{U}\left([0,1]^{100}\right),$$

and conditioned with the class y, encoded as a one-hot vector of dimension 10, the model is

To generate MNIST characters, with

$$Z \sim \mathscr{U}\left([0,1]^{100}\right),$$

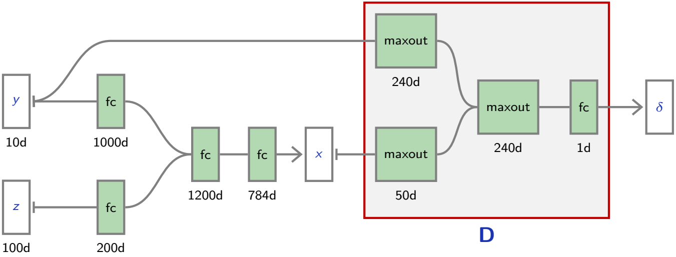
and conditioned with the class y, encoded as a one-hot vector of dimension 10, the model is



To generate MNIST characters, with

$$Z \sim \mathscr{U}\left([0,1]^{100}\right),$$

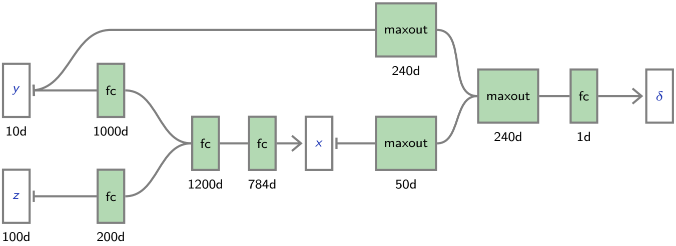
and conditioned with the class y, encoded as a one-hot vector of dimension 10, the model is



To generate MNIST characters, with

$$Z \sim \mathscr{U}\left([0,1]^{100}\right),$$

and conditioned with the class y, encoded as a one-hot vector of dimension 10, the model is



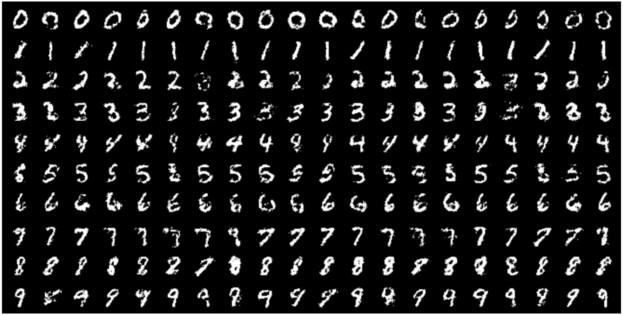


Figure 2: Generated MNIST digits, each row conditioned on one label

(Mirza and Osindero, 2014)

Another option to condition the generator consists of making the parameter of its batchnorm layers class-conditional (Dumoulin et al., 2016).

(Brock et al., 2018)

(Brock et al., 2018)

Image-to-Image translations

The main issue to generate realistic signals is that the value X to predict may remain non-deterministic given the conditioning quantity Y.

The main issue to generate realistic signals is that the value X to predict may remain non-deterministic given the conditioning quantity Y.

For a loss function such as MSE, the best fit is $\mathbb{E}(X|Y=y)$ which can be pretty different from the MAP, or from any reasonable sample from $\mu_{X|Y=y}$.

In practice, for images there is often remaining location indeterminacy that results into a blurry prediction.

The main issue to generate realistic signals is that the value X to predict may remain non-deterministic given the conditioning quantity Y.

For a loss function such as MSE, the best fit is $\mathbb{E}(X|Y = y)$ which can be pretty different from the MAP, or from any reasonable sample from $\mu_{X|Y=y}$.

In practice, for images there is often remaining location indeterminacy that results into a blurry prediction.

Sampling according to $\mu_{X|Y=y}$ is the proper way to address the problem.

Isola et al. (2016) use a GAN-like setup to address this issue for the "translation" of images with pixel-to-pixel correspondence:

- edges to realistic photos,
- semantic segmentation,
- gray-scales to colors, etc.

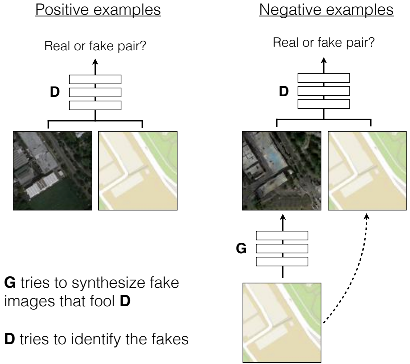


Figure 2: Training a conditional GAN to predict aerial photos from maps. The discriminator, D, learns to classify between real and synthesized pairs. The generator learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe an input image.

(Isola et al., 2016)

They define

$$V(\mathbf{D}, \mathbf{G}) = \mathbb{E}_{(X,Y)\sim\mu}\Big[\log \mathbf{D}(Y, X)\Big] + \mathbb{E}_{Z\sim\mu_Z, X\sim\mu_X}\Big[\log(1 - \mathbf{D}(\mathbf{G}(Z, X), X))\Big],$$

$$\mathscr{L}_{L^1}(\mathbf{G}) = \mathbb{E}_{(X,Y)\sim\mu, Z\sim\mathscr{N}(0,I)}\Big[\, \|Y - \mathbf{G}(Z, X)\|_1 \Big],$$

and

$$\mathbf{G}^* = \underset{\mathbf{G}}{\operatorname{argmin}} \max_{\mathbf{D}} \; V(\mathbf{D}, \mathbf{G}) + \lambda \mathscr{L}_{L^1}(\mathbf{G}).$$

They define

$$V(\mathbf{D}, \mathbf{G}) = \mathbb{E}_{(X,Y)\sim\mu}\Big[\log \mathbf{D}(Y, X)\Big] + \mathbb{E}_{Z\sim\mu_Z, X\sim\mu_X}\Big[\log(1 - \mathbf{D}(\mathbf{G}(Z, X), X))\Big],$$

$$\mathscr{L}_{L^1}(\mathbf{G}) = \mathbb{E}_{(X,Y)\sim\mu, Z\sim\mathscr{N}(0,I)}\Big[\|Y - \mathbf{G}(Z, X)\|_1\Big],$$

and

$$\mathbf{G}^* = \underset{\mathbf{G}}{\operatorname{argmin}} \max_{\mathbf{D}} \; V(\mathbf{D}, \mathbf{G}) + \lambda \mathscr{L}_{L^1}(\mathbf{G}).$$

The term $\mathscr{L}_{L^1}$ pushes toward proper pixel-wise prediction, and V makes the generator prefer realistic images to better fitting pixel-wise.

They define

$$V(\mathbf{D}, \mathbf{G}) = \mathbb{E}_{(X,Y)\sim\mu}\Big[\log \mathbf{D}(Y, X)\Big] + \mathbb{E}_{Z\sim\mu_Z, X\sim\mu_X}\Big[\log(1 - \mathbf{D}(\mathbf{G}(Z, X), X))\Big],$$

$$\mathscr{L}_{L^1}(\mathbf{G}) = \mathbb{E}_{(X,Y)\sim\mu, Z\sim\mathscr{N}(0,I)}\Big[\|Y - \mathbf{G}(Z, X)\|_1\Big],$$

and

$$\mathbf{G}^* = \underset{\mathbf{G}}{\operatorname{argmin}}\,\max_{\mathbf{D}}\; V(\mathbf{D}, \mathbf{G}) + \lambda \mathscr{L}_{L^1}(\mathbf{G}).$$

The term $\mathscr{L}_{L^1}$ pushes toward proper pixel-wise prediction, and V makes the generator prefer realistic images to better fitting pixel-wise.

⚠ Note that contrary to Mirza and Osindero's convention, here X is the conditioning quantity and Y the signal to generate.

For **G**, they start with Radford et al. (2015)'s DCGAN architecture and add skip connections from layer i to layer $D - i$ that concatenate channels.

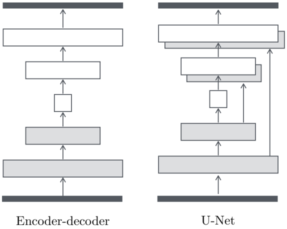


Figure 3: Two choices for the architecture of the generator. The "U-Net" [34] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

(Isola et al., 2016)

For **G**, they start with Radford et al. (2015)'s DCGAN architecture and add skip connections from layer i to layer $D - i$ that concatenate channels.

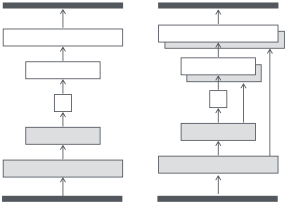


Figure 3: Two choices for the architecture of the generator. The "U-Net" [34] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

(Isola et al., 2016)

Randomness Z is provided through dropout, and not as an additional input.

The discriminator **D** is a regular convnet which scores overlapping patches of size $N \times N$ and averages the scores for the final one.

This controls the network's complexity, while allowing to detect any inconsistency of the generated image (e.g. blurriness).

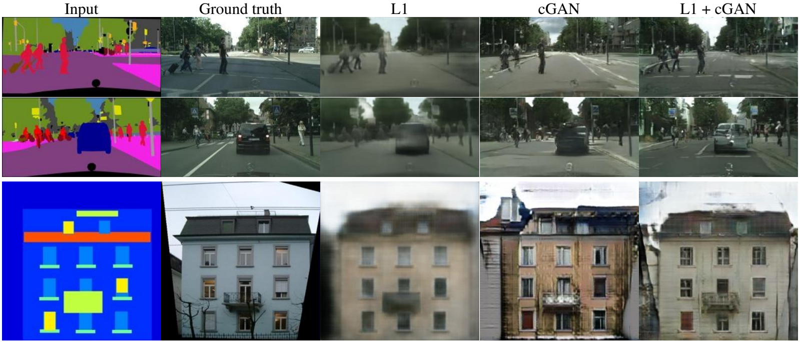


Figure 4: Different losses induce different quality of results. Each column shows results trained under a different loss. Please see `https://phillipi.github.io/pix2pix/` for additional examples.

(Isola et al., 2016)

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 70x70 PatchGAN forces outputs that are sharp, even if incorrect, in both the spatial and spectral (coforfulness) dimensions. The full 256x256 ImageGAN produces results that are visually similar to the 70x70 PatchGAN, but somewhat lower quality according to our FCN-score metric (Table 2). Please see `https://phillipi.github.io/pix2pix/` for additional examples.

(Isola et al., 2016)

Figure 8: Example results on Google Maps at 512x512 resolution (model was trained on images at 256x256 resolution, and run convolutionally on the larger images at test time). Contrast adjusted for clarity.

(Isola et al., 2016)

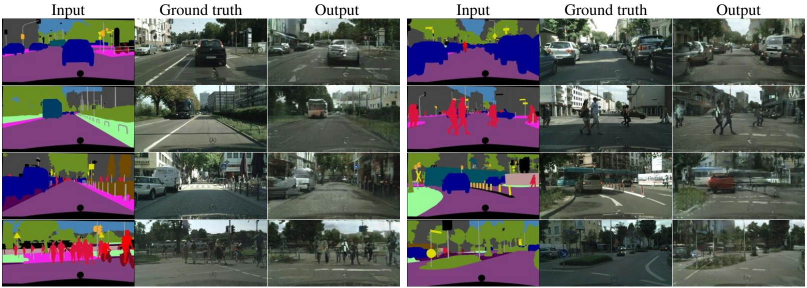


Figure 11: Example results of our method on Cityscapes labels→photo, compared to ground truth.

(Isola et al., 2016)

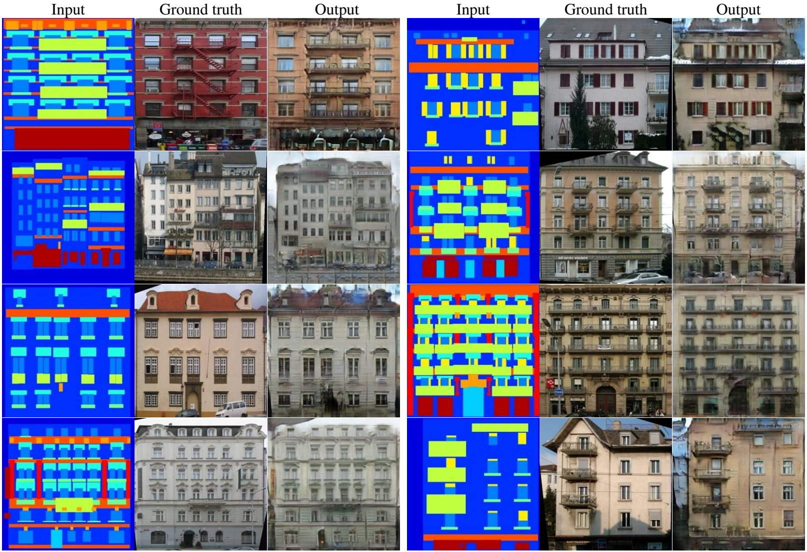


Figure 12: Example results of our method on facades labels→photo, compared to ground truth

(Isola et al., 2016)

Figure 13: Example results of our method on day→night, compared to ground truth.

(Isola et al., 2016)

Figure 14: Example results of our method on automatically detected edges→handbags, compared to ground truth.

(Isola et al., 2016)

Figure 16: Example results of the edges→photo models applied to human-drawn sketches from [10]. Note that the models were trained on automatically detected edges, but generalize to human drawings

(Isola et al., 2016)

The main drawback of this technique is that it requires pairs of samples with pixel-to-pixel correspondence.

In many cases, one has at its disposal examples from two densities and wants to translate a sample from the first ("images of apples") into a sample likely under the second ("images of oranges").

We consider X r.v. on $\mathscr{X}$ a sample from the first data-set, and Y r.v. on $\mathscr{Y}$ a sample for the second data-set. Zhu et al. (2017) propose to train at the same time two mappings

$$\begin{aligned} \mathbf{G} &: \mathscr{X} \to \mathscr{Y} \\ \mathbf{F} &: \mathscr{Y} \to \mathscr{X} \end{aligned}$$

such that

$$\begin{aligned} \mathbf{G}(X) &\sim \mu_Y, \\ \mathbf{F} \circ \mathbf{G}(X) &\simeq X. \end{aligned}$$

Where the matching in density is characterized with a discriminator $\mathbf{D}_Y$ and the reconstruction with the L^1 loss. They also do this both ways symmetrically.

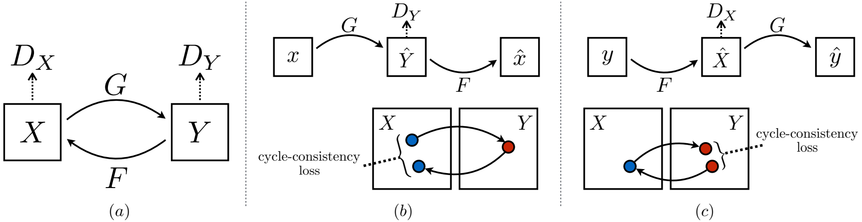
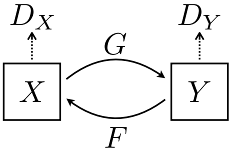


Figure 3: (a) Our model contains two mapping functions $G : X \to Y$ and $F : Y \to X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \to G(x) \to F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \to F(y) \to G(F(y)) \approx y$

(Zhu et al., 2017)

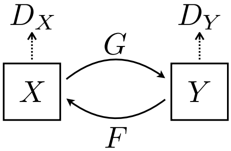


The loss optimized alternatively is

$$\begin{aligned} V^*(\mathbf{G}, \mathbf{F}, \mathbf{D}_X, \mathbf{D}_Y) =& V(\mathbf{G}, \mathbf{D}_Y, X, Y) + V(\mathbf{F}, \mathbf{D}_X, Y, X) \\ & + \lambda \left(\mathbb{E}\Big[\|\mathbf{F}(\mathbf{G}(X)) - X\|_1 \Big] + \mathbb{E}\Big[\|\mathbf{G}(\mathbf{F}(Y)) - Y\|_1 \Big] \right) \end{aligned}$$

where V is a quadratic loss, instead of the usual $\log$ (Mao et al., 2016)

$$V(\mathbf{G}, \mathbf{D}_Y, X, Y) = \mathbb{E}\Big[(\mathbf{D}_Y(Y) - 1)^2 \Big] + \mathbb{E}\Big[\mathbf{D}_Y(\mathbf{G}(X))^2 \Big].$$



The loss optimized alternatively is

$$
\begin{aligned}
V^*(\mathbf{G}, \mathbf{F}, \mathbf{D}_X, \mathbf{D}_Y) = & V(\mathbf{G}, \mathbf{D}_Y, X, Y) + V(\mathbf{F}, \mathbf{D}_X, Y, X) \\
& + \lambda \left(\mathbb{E}\Big[\|\mathbf{F}(\mathbf{G}(X)) - X\|_1 \Big] + \mathbb{E}\Big[\|\mathbf{G}(\mathbf{F}(Y)) - Y\|_1 \Big] \right)
\end{aligned}
$$

where V is a quadratic loss, instead of the usual log (Mao et al., 2016)

$$
V(\mathbf{G}, \mathbf{D}_Y, X, Y) = \mathbb{E}\Big[(\mathbf{D}_Y(Y) - 1)^2 \Big] + \mathbb{E}\Big[\mathbf{D}_Y(\mathbf{G}(X))^2 \Big].
$$

The generator is from Johnson et al. (2016), an updated version of Radford et al. (2015)'s DCGAN, with plenty of specific tricks, e.g. using an history of generated images (Shrivastava et al., 2016).

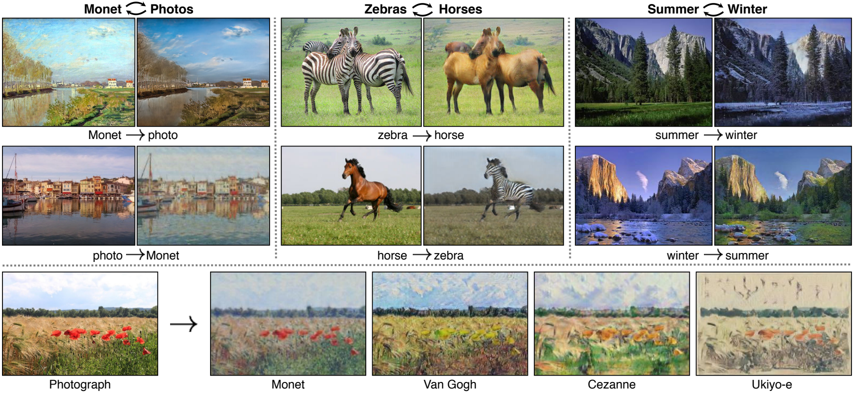


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

(Zhu et al., 2017)


zebra → horse


winter Yosemite → summer Yosemite


summer Yosemite → winter Yosemite

(Zhu et al., 2017)

apple → orange

orange → apple

(Zhu et al., 2017)

While GANs are often used for their [theoretical] ability to model a distribution, generating consistent samples is enough for image-to-image translation.

In particular, this application does not suffer much from mode collapse, as long as the generated images "look nice".

The key aspect of the GAN here is the "perceptual loss" that the discriminator implements, more than the theoretical convergence to the true distribution.

The end

References

A. Brock, J. Donahue, and K. Simonyan. **Large scale GAN training for high fidelity natural image synthesis**. CoRR, abs/1809.11096, 2018.

V. Dumoulin, J. Shlens, and M. Kudlur. **A learned representation for artistic style**. CoRR, abs/1610.07629, 2016.

P. Isola, J. Zhu, T. Zhou, and A. A. Efros. **Image-to-image translation with conditional adversarial networks**. CoRR, abs/1611.07004, 2016.

J. Johnson, A. Alahi, and L. Fei-Fei. **Perceptual losses for real-time style transfer and super-resolution**. In European Conference on Computer Vision (ECCV), 2016.

X. Mao, Q. Li, H. Xie, R. Lau, Z. Wang, and S. Smolley. **Least squares generative adversarial networks**. CoRR, abs/1611.04076, 2016.

M. Mirza and S. Osindero. **Conditional generative adversarial nets**. CoRR, abs/1411.1784, 2014.

A. Radford, L. Metz, and S. Chintala. **Unsupervised representation learning with deep convolutional generative adversarial networks**. CoRR, abs/1511.06434, 2015.

A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. **Learning from simulated and unsupervised images through adversarial training**. CoRR, abs/1612.07828, 2016.

J. Zhu, T. Park, P. Isola, and A. Efros. **Unpaired image-to-image translation using cycle-consistent adversarial networks**. CoRR, abs/1703.10593, 2017.