Deep learning

11.3. Conditional GAN and image translation

François Fleuret

https://fleuret.org/dlc/
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 $G$ and $D$ by a conditioning quantity $Y$.

$$V(D, G) = \mathbb{E}_{(X, Y) \sim \mu} \left[ \log D(X, Y) \right] + \mathbb{E}_{Z \sim \mathcal{N}(0, I), Y \sim \mu_Y} \left[ \log (1 - D(G(Z, Y), Y)) \right],$$
To generate MNIST characters, with

\[ Z \sim \mathcal{U}([0, 1]^{100}) , \]

and conditioned with the class \( y \), encoded as a one-hot vector of dimension 10, the model is
To generate MNIST characters, with

\[ Z \sim \mathcal{U}([0, 1]^{100}), \]

and conditioned with the class \( y \), encoded as a one-hot vector of dimension 10, the model is
To generate MNIST characters, with

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

and conditioned with the class \(y\), encoded as a one-hot vector of dimension 10, the model is
To generate MNIST characters, with

\[ Z \sim \mathcal{U}([0, 1]^{100}), \]

and conditioned with the class \( y \), encoded as a one-hot vector of dimension 10, the model is
Table 1: Parzen window-based log-likelihood estimates for MNIST. We followed the same procedure as [8] for computing these values.

The discriminator maps \( x \) to a maxout [6] layer with 240 units and 5 pieces, and \( y \) to a maxout layer with 50 units and 5 pieces. Both of the hidden layers mapped to a joint maxout layer with 240 units and 4 pieces before being fed to the sigmoid layer. (The precise architecture of the discriminator is not critical as long as it has sufficient power; we have found that maxout units are typically well suited to the task.)

The model was trained using stochastic gradient decent with mini-batches of size 100 and initial learning rate of \( 0.1 \) which was exponentially decreased down to \( 0.000001 \) with decay factor of \( 1.00004 \). Also momentum was used with initial value of \( 0.5 \) which was increased up to \( 0.7 \). Dropout [9] with probability of 0.5 was applied to both the generator and discriminator. And best estimate of log-likelihood on the validation set was used as stopping point.

Table 1 shows Gaussian Parzen window log-likelihood estimate for the MNIST dataset test data. 1000 samples were drawn from each 10 class and a Gaussian Parzen window was fitted to these samples. We then estimate the log-likelihood of the test set using the Parzen window distribution. (See [8] for more details of how this estimate is constructed.)

The conditional adversarial net results that we present are comparable with some other network based, but are outperformed by several other approaches – including non-conditional adversarial nets. We present these results more as a proof-of-concept than as demonstration of efficacy, and believe that with further exploration of hyper-parameter space and architecture that the conditional model should match or exceed the non-conditional results.

Figure 2 shows some of the generated samples. Each row is conditioned on one label and each column is a different generated sample.

(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 $Y$. For a loss function such as MSE, the best fit is $E(X|Y=y)$ 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.
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.
G tries to synthesize fake images that fool D

D tries to identify the fakes

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

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

\[ \mathcal{L}_{L1}(G) = \mathbb{E}_{(X, Y) \sim \mu, Z \sim \mathcal{N}(0, I)} \left[ \| Y - G(Z, X) \|_1 \right], \]

and

\[ G^* = \arg\min_G \max_D V(D, G) + \lambda \mathcal{L}_{L1}(G). \]
They define

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

\[ \mathcal{L}_{L^1}(G) = \mathbb{E}_{(X, Y) \sim \mu, Z \sim \mathcal{N}(0, I)} \left[ \| Y - G(Z, X) \|_1 \right] , \]

and

\[ G^* = \arg\min_G \max_D V(D, G) + \lambda \mathcal{L}_{L^1}(G). \]

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

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

\[ \mathcal{L}_{L1}(G) = \mathbb{E}_{(X, Y) \sim \mu} \left[ \| Y - G(Z, X) \|_1 \right], \]

and

\[ G^* = \arg \min_G \max_D V(D, G) + \lambda \mathcal{L}_{L1}(G). \]

The term \( \mathcal{L}_{L1} \) 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.

![Diagram of Encoder-decoder and U-Net architectures.](image)

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.

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 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.

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

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

\[
G : \mathcal{X} \rightarrow \mathcal{Y} \\
F : \mathcal{Y} \rightarrow \mathcal{X}
\]

such that

\[
G(X) \sim \mu_Y, \\
F \circ G(X) \simeq X.
\]

Where the matching in density is characterized with a discriminator \( D_Y \) and the reconstruction with the \( L^1 \) loss. They also do this both ways symmetrically.
Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, 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: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

(Zhu et al., 2017)
The loss optimized alternatively is

\[
V^*(G, F, D_X, D_Y) = V(G, D_Y, X, Y) + V(F, D_X, Y, X) \\
+ \lambda \left( \mathbb{E}\left[ \|F(G(X)) - X\|_1 \right] + \mathbb{E}\left[ \|G(F(Y)) - Y\|_1 \right] \right)
\]

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

\[
V(G, D_Y, X, Y) = \mathbb{E}\left[ (D_Y(Y) - 1)^2 \right] + \mathbb{E}\left[ D_Y(G(X))^2 \right].
\]
The loss optimized alternatively is

\[
V^*(G, F, D_X, D_Y) = V(G, D_Y, X, Y) + V(F, D_X, Y, X) + \lambda \left( \mathbb{E} \left[ \|F(G(X)) - X\|_1 \right] + \mathbb{E} \left[ \|G(F(Y)) - Y\|_1 \right] \right)
\]

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

\[
V(G, D_Y, X, Y) = \mathbb{E} \left[ (D_Y(Y) - 1)^2 \right] + \mathbb{E} \left[ D_Y(G(X))^2 \right].
\]

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)
Figure 13: Our method applied to several translation problems. These images are selected as relatively successful results—please see our website for more comprehensive and random results. In the top two rows, we show results on object transfiguration between horses and zebras, trained on 939 images from the wild horse class and 1177 images from the zebra class in Imagenet [41]. The middle two rows show results on season transfer, trained on winter and summer photos of Yosemite from Flickr. In the bottom two rows, we train our method on 996 apple images and 1020 navel orange images from ImageNet.

(Zhu et al., 2017)
Figure 13: Our method applied to several translation problems. These images are selected as relatively successful results—please see our website for more comprehensive and random results. In the top two rows, we show results on object transfiguration between horses and zebras, trained on 939 images from the wild horse class and 1177 images from the zebra class in Imagenet [41]. The middle two rows show results on season transfer, trained on winter and summer photos of Yosemite from Flickr. In the bottom two rows, we train our method on 996 apple images and 1020 navel orange images from ImageNet.

(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


