University of Michigan
EECS 504: Foundations of Computer Vision
Winter 2020. Instructor: Andrew Owens.

Problem Set 7: Image Translation

Posted: Tuesday, March 10, 2020 **Due:** Tuesday, March 17, 2020

For Problem 7.2 and 7.3, please submit your written solution to Gradescope as a `.pdf` file. For Problem 7.1, please submit your solution to Canvas as a notebook file (`.ipynb`), containing the visualizations that we requested. Your `.ipynb` notebook should be named as `<unique_name>_<umid>.ipynb`. Example: `adam_01100001.ipynb`. Also, please remember to put your name and unique name in the first text block of the notebook.

The starter code can be found at:
https://drive.google.com/open?id=178F3h1DEW9cn5WVohHiCxlfPBJqCeLs_

We recommend editing and running your code in Google Colab, although you are welcome to use your local machine instead.

Problem 7.1 *Implement pix2pix*

In this problem set, we will implement an image-to-image translation method, based on pix2pix [1]. We hope that this will give you experience reading a modern computer vision paper and implementing the architecture described within it.

We'll train a model to generate pictures of facades from label maps, using the CMP Facade Database [2]. This dataset contains 606 rectified images of facades from various sources, which have been manually annotated. Below is a sample image from the database. In this homework, we will implement GAN to translate labels into facade images.

We recommend reading the paper first and understanding how pix2pix works. Below, we describe some key steps for you implementation. You should refer to the comments in the provided notebook for further implementation details.

1. We will first build data loaders for training and testing, using the CMP Facade Database. (1 point).

   For the training process, we will use batch size equal to 1, as in the original paper. During testing, we will process 10 images in a single batch, so that we can visualize several results at once.

2. In the next step, we will define the network based on the architectures from the paper. Please check the Appendix of the paper for the details of this architecture. [1]

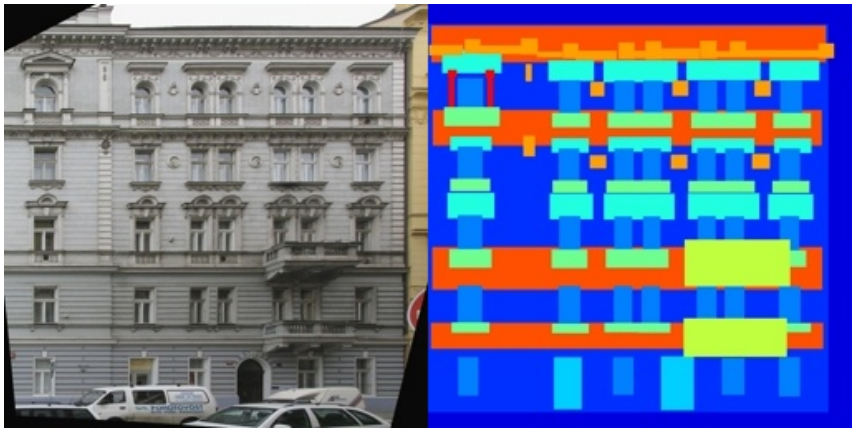

Figure 1: An example from the Facades dataset. In this problem set, you will generate images images, like the one on the left, from the label map on the right.

As a reminder: let Ck denote a Convolution-BatchNorm-ReLU layer with k filters. CDk denotes a Convolution-BatchNormDropout-ReLU layer with a dropout rate of 50%. All convolutions are 4 x 4 spatial filters applied with stride 2. Convolutions in the encoder, and in the discriminator, downsample the input by a factor of 2, whereas in the decoder they upsample the input by a factor of 2.

(a) Generator architectures (2 points)

The U-Net encoder-decoder architecture consists of:
U-Net encoder:
C64-C128-C256-C512-C512-C512-C512-C512
U-Net decoder:
CD512-CD1024-CD1024-C1024-C1024-C512-C256-C128

After the last layer in the decoder, a convolution is applied to map to the number of output channels, which is 3 in our problem, followed by a tanh function. As a special case, batch normalization 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.

(b) Discriminator architectures (2 points)

The 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, batch normalization is not applied to the first C64 layer. All ReLUs are leaky, with slope 0.2.

Hint: Using torch.nn.functional.leaky_relu for leaky ReLU.

3. For optimization, we'll use the Adam optimizer, with a learning rate of 0.0002, and momentum parameters $\beta_1 = 0.5, \beta_2 = 0.999$. (0.5 point)

4. Now, we will implement the training process. (2 points)

   In each epoch, first train discriminator D by using the average loss of real image and fake image. Then train generator by using the following loss equation.

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \tag{1}$$

   In this homework, you will train two different models: one with only L1 loss, the other use the equation above with $\lambda = 100$. As a sanity check, please compare your result with those in the paper to make sure they roughly match.

Figure 2: Different losses induce different quality of results. Each column shows results trained under a different loss.[1]

5. Train the network for at least 20 epochs (10 epochs for the model with only L1 loss,) but you are welcome to train longer to obtain better results. (In the paper, they train for 200 epoch.) Note: training each epoch will take less than 2 minutes.

6. In the end, plot the G/D loss history v.s. Iteration of two models in seperate plot. (0.5 point)

Some tips and resources:

1. Class lecture slides on image synthesis.

2. The pix2pix paper [1] and pix2pix website.

Problem 7.2 *Understand pix2pix: part 1*

Given the input image with the size $256 \times 256 \times 3$ (6 for discriminator), if we use the network architectures above, write down the size of each neuron's receptive field after each layer **(Update:) only for the Discriminator**. (1.5 points)

e.g. $Input \longrightarrow C64 \longrightarrow (?) \longrightarrow C128 \longrightarrow (?) \longrightarrow C256 \longrightarrow (?) \longrightarrow C512 \longrightarrow (?)$

Update: Make sure you calculate receptive field, not the size of the image.

Problem 7.3 *Understand pix2pix: part 2*

Explain why only minimizing the L1 loss results in a blurry image. (0.5 point)

References

[1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[2] Radim Tyleček and Radim Šára. Spatial pattern templates for recognition of objects with regular structure. In *Proc. GCPR*, Saarbrucken, Germany, 2013.