DEEP LEARNING 2022

IMAGE GENERATION WITH GENERATIVE ADVERSARIAL NETWORKS PROJECT NO. 3 PLAN

May 16, 2022

Elżbieta Jowik (298821), Agata Makarewicz (298827)

1 Datasets

The dataset used in this project is a 10% sample of the bedroom category from the *Large-scale Image Dataset (LSUN)* dataset [1]. The original dataset contains around one million labelled images for each of 10 scene categories and 20 object categories. The sample used for this project consists of 303125 images presenting bedroom scenes and is sourced from *Kaggle*.

2 Data preparation

Within the data preparation step, we plan to perform the following sequence of actions:

1. use the `generate_noise` function to generate noise vectors to be passed into the generator for image generation,
2. resize and centre crop all images,
3. augment dataset, for instance, applying a random zoom,
4. normalize images with ImageNet stats.

Optionally, unless it turns out to be too computationally burdensome, we will reproduce crappification logic presented in one of the `fastai` course solutions.

3 Network architectures

GANs standing for Generative Adversarial Nets is a framework for teaching a DL model to capture the training data's distribution so we can generate new data from that same distribution. The concept is that there are two models trained at the same time: a generator and a critic. The generator will try to make new images similar to the ones from the given dataset, and the critic will try to classify authentic pictures from the ones the generator does.
Models are trained against each other in the sense that at each step (more or less), the generator is frozen and the critic is trained by:

- getting one batch of real images,
- generating one batch of fake images,
- having the critic evaluate each batch and compute a loss function from that; the important part is that it rewards positively the detection of real images and penalizes the fake ones,
- update the weights of the critic with the gradients of this loss

The generator is designed to return images, while the critic is a binary classification network that takes an image as input and outputs a scalar probability that the input image is real (as opposed to fake).

We are going to test and compare 3 different network architectures:

- `Wassertein GAN` (`fastai` package)
- `GAN` (`fastai` package)
- `DC-GAN` (`pytorch` package)

[1]LSUN Dataset `http://www.yf.io/p/lsun`

To implement the first of the proposed approaches we will define a `generator` and a `critic` to be passed to `gan_learner`.
To define a `GAN Learner` (the 2nd approach) we will combine (previously pre-trained) generator and critic learner objects. For this purpose, we will also specify the switcher, which is a callback that decides when to switch between discriminator and generator. We will follow a literature-sourced approach of doing as many iterations of the discriminator as needed to get its loss back < 0.5 then one iteration of the generator. As the loss of the generator, we plan to adopt MSE loss (`MSELossFlat`).
The last idea will be taken as a state-of-the-art approach sourced directly from `pytorch` documentation. We assume that DCGAN, which is a direct extension of the GAN applied to the given dataset, will converge to generate satisfactory images.

4 Learning process parametrization & evaluation

As an optimizer, we plan to use the ADAM method. We will investigate the influence of hyperparameters (such as learning rate) on obtained results, and try to find a set of hyperparameters which help in overcoming training and mode collapse. Models will be evaluated using both quantitative metrics (we are going to calculate the Frechet Inception Distance (FID) and compare it to results from literature) and qualitative assessment (we plan to use the nearest neighbours approach or simple Euclidean distance between images pixel data).