CMPUT 328 Fall 2022

Assignment 9 Generative Models

Worth 5 + 5% of the total weight

Part 1: Variational Autoencoder [5% of the total weight]:

In this section, you are to complete the implementations of Variational Autoencoder (VAE) and Conditional Variation Autoencoder (CVAE) in the provided template script (A9_vae.py). You will perform training and testing with Cifar10 dataset.

In total, you are required to complete the following three sections in order to gain full marks.

Section 1: Encoder-Decoder Architecture

In this section, you are required to **complete the network architecture for both VAE and CVAE**. The encoder-decoder structures should use the overall similar architecture, except for CVAE, some additional layers will be needed to encode classes as conditional inputs.

The encoder-decoder architecture in this assignment is not completely fixed. You will design the encoder-decoder architecture yourself. However, besides a correct architecture to perform encoding and decoding for VAE, there are some requirements that your network architecture must satisfy:

- Your encoder-decoder architecture must be of a **U-Net architectural design**. However, for VAE (and CVAE), note that, **you should not use skip connections between encoder and decoder features**!
- Additionally, in the encoder architecture, you must have a) at least 3 convolution layers or depth-wise separable convolution layers to encode image features, and b) at least 2 down-sampling layers (max pooling, strided convolution layers, etc).
- Additionally, in the decoder architecture, you must have at least 2 up-sampling layers (UpSample layers with Bilinear Interpolation or transposed convolution layers).
- Complete forward methods for both VAE and CVAE model. In the provided CVAE model class, you are to insert any additional layers to correctly embed class information to condition your decoder.

Failure to satisfy any of the requirements will directly disqualify the evaluation of 1% mark assigned to evaluate the total reconstruction loss, with further potential reduction of 4% mark assigned to evaluate code correctness.

Section 2: Training & Testing

In this section, you are required to complete the missing training code of your VAE and CVAE. You should be able to swap between the two types of variational autoencoders by using the flag “conditional”.

The missing parts consist of:

- **A joint “reconstruction loss”.** In addition to a L2, you are to use two additional losses, **binary cross-entropy and structural similarity index measure (SSIM)**, **and use the summation of the three losses as a joint loss**, to further regularize the quality of your reconstructed images. The joint reconstruction loss is formulated as follows:

$$L_{reconstruction}(x, \hat{x}) = MSE(x, \hat{x}) + L_{SSIM}(x, \hat{x}) + L_{BCE}(\hat{x}, x)$$

For binary cross-entropy loss, you are free to use PyTorch's own implementation, "BCELoss".
For SSIM, a pre-implemented class is provided in the file "ssim.py", you are to complete the calculation of this loss **with a window size of (11, 11)** as:

$$L_{ssim}(x, \hat{x}) = 1 - SSIM_{(11,11)}(x, \hat{x})$$

Failure to implement the joint reconstruction loss will directly disqualify the evaluation of 1% mark assigned to evaluate the total reconstruction loss, with further potential reduction of 4% mark assigned to evaluate code correctness.

- One train step of your VAE and CVAE. You are required to correctly implementation the forward and backward process of your VAE and CVAE, determined by the summation of "reconstruction loss" and "KL-Divergence loss" (A wrapper for KL Divergence loss is provided) terms as:

$$L_{VAE} = L_{reconstruction}(x, \hat{x}) + scalar * L_{KLD}(\mu, \sigma)$$

- **KL-Annealing.** You are required to **dynamically adjust the scaling ratio of your KL-Divergence loss based on the epoch number you are at during training**. **This will tremendously help with the convergence of your VAE and CVAE.** Start with a scalar of 0.0 (treating VAE as a vanilla autoencoder instead), then increase the scalar to 0.0001, then gradually increase it to 1.0.
- **Denormalization.** You are required to complete the mathematical process to **reverse the pre-processing of any image in range [0, 1], back to the data type of "numpy.uint8".** The provided visualization code in matplotlib will help you visually determine the correctness of your implementation.
- **Generation.** You are to randomly sample from a Gaussian distribution of 0 mean and 1 standard deviation and perform reconstruction with your decoder. The "generate" method is provided in class method of VAE and CVAE. You are required to complete them in order for the last visualization part of the random generated samples to work.

Expected Performance: A correctly implemented, and somewhat lightly-tuned version of this algorithm will have an total joint reconstruction loss of 0.68 for VAE, 0.73 for CVAE or lower, using Cifar10 test set. Please write down your achieved performance (average total reconstruction loss per sample on Cifar10 test set) for both VAE and CVAE, on top of the A9_vae.py file in the comment section, along with your name and CCID.

Part 2: AC-GAN [A Bonus 5% of the total weight]

In this section, you are to implement AC-GAN in the provided template script (A9_acgan.py). You will perform training and testing with Cifar10 dataset.

In total, you are required to complete the following two sections in order to gain full marks.

Section 1: Generator-Discriminator Architecture

In this section, you are required to **complete the network architecture for your AC-GAN architecture**. The GAN architecture in this assignment is not completely fixed. However, there are a few restrictions:

- For the generator architecture, it should be **mostly similar to the decoder architecture in your part 1**, with a few twists in non-linear activation functions (e.g. LeakyReLU instead of ReLU).
- For the discriminator architecture, you should follow the design principle of the AC-GAN, and insert any additional layers to **produce both discrimination score and auxiliary classification score**.

Failure to satisfy any of the requirements will directly disqualify the evaluation of 1% mark assigned to evaluate the auxiliary classification score, with further potential reduction of 4% mark assigned to evaluate code correctness.

Section 2: Training & Testing

In this section, you are required to complete the missing parts to train and test your AC-GAN.

The missing parts consist of:

- **One train step of your AC-GAN. For the training of your discriminator, you should calculate losses for the discriminator and the auxiliary classifier**.
- **Denormalization.** The requirement is the same as Part 1. You are required to complete the mathematical process to reverse the pre-processing of any image in range [0, 1], back to the data type of "numpy.uint8". The provided visualization code in matplotlib will help you visually determine the correctness of your implementation.
- **Visualization.** You should **randomly sample from a Gaussian distribution of mean 0 and standard deviation of 1, with 20 samples**. This will be used as the latent representation input into your generator, **generating 20 images from your AC-GAN for visualization purpose**.

Expected Performance: A correctly implemented, and somewhat lightly-tuned version of this algorithm will have an auxiliary classification score of 69% or higher on Cifar10 test set. Please write down your achieved performance (auxiliary test accuracy on Cifar10 test set) on top of the A9_acgan.py file in the comment section, along with your name and CCID.

Template Code:

You are provided with template code in the form of 3 files: "A9_vae.py", "A9_acgan.py", and "ssim.py".

You need to complete the required missing sections for both "A9_vae.py" and "A9_acgan.py".

The code can be run using *python3 A9_vae.py* for part 1 and *python3 A9_acgan.py* for part 2 on your own machine or *!python3 "<full path to A9_vae.py or A9_ acgan.py>"* from a code cell in Colab.

You can also create notebooks on Colab and copy all code from the template files to cells.

Submission:

For Part 1, **you need to submit the following files to be qualified for evaluation:**

- Your completed code for "A9_vae.py"
- One reconstructed visualization from both VAE and CVAE (a total of two visualizations) from your best performing model, and a reconstructed visualization of randomly sampled latent vectors from both VAE and CVAE from your best model. All visualizations are saved under "visualize/vae" or "visualize/cvae".
- Plots for "reconstructed losses" and "total train vs. test loss". This is also saved under "visualize/vae" or "visualize/cvae".

For Part 2, y**ou need to submit the following files to be qualified for evaluation:**

- Your completed code for "A9_acgan.py".
- One generated visualization from your AC-GAN. This is saved under "visualize/gan".
- Plots for "generator loss vs. discriminator loss". This is also saved under "visualize/gan"

Marking:

Part 1

- 4% of the total marks will be given based on the correctness of your implementation.
- 1% of the total marks will be given based on the total reconstruction loss. You should achieve a minimum of 0.69 or lower average total reconstruction loss per sample in the test set with VAE, and a minimum of 0.73 or lower with CVAE, to get the maximum score. Score will scale linearly from 0.72-0.69 for VAE, and 0.73-0.76 for CVAE on the Cifar10 test set reconstruction.

Part 2

- 4% of the total marks will be given based on the correctness of your implementation.
- 1% of the total marks will be given based on the auxiliary classification accuracy. You should achieve a minimum of 69% or higher test accuracy on the Cifar10 test set, to get the maximum score. Score will scale linearly from 65% - 69% on the test set.

Collaboration:

Using existing code: Some of the functionality required for this part might be present lecture notebooks. You are advised to avoid any copy-pasting from there but if you do, make sure to document it in enough detail to make it absolutely clear to the marking TA that you understand what is going on.

This applies to copying from any source in general, but this particular case is important since any instance of undocumented copying from lecture notebooks will **lead to heavy penalty (beyond the one third reserved for documentation) and might even lead to zero marks**.