IERG4160 2023-24 Spring: Individual Project

Image and Video Processing

Due date: April. 21st (Sunday), 11:59pm

**Introduction:**

In this project, this is an individual project. You will finish a task related to neural network training, submit your code, and write a written report.

Projects are assessed based on:

* **Code and results**. You are required to submit your colab, together with results, as **PDF**.
* **Reports**. You are required to write a few-page report to summarize your effort and your results.

Optional project. At very last, we also provide an **alternative optional bonus project** (if you choose that one, you do not need to work on the main project). It requires some efforts to install additional libraries on Linux machine, and figure out how to use some softwares, including colab and NeRFstudio. Therefore, only choose that option if you are very confident that you are familiar with Linux and 3D geometry.

**General project. Neural network training**

In Assignment 3 (in-class lab), you already learn how to run a basic neural network training job. In the project, you will delve in details.

You are asked to finish the following tasks, run the program, and write a report based on your results.

**Task 1**. Try to make following changes to the network:

* Run the CNN-based network training job with enough epochs, say 10 epochs, and plot the epoch(x-axis)-accuracy(y-axis) curve.
* For both CNN-based networks, try different batch size and plot both: batchsize-accuracy and batchSize-runtime curve.
* Explore few other parameters, like learning rate, number of intermediate channels, and see how accuracy changes. You could also alternatively make some changes to the network, like switching ReLU layer by other non-linear operators.

**Task 2 [Optional bonus question]**.

* Utilize the input images in the MNIST dataset (ignore the labels) to build a denoising training dataset and train a denoising network. This includes:
  + **Create** a training dataset. You need to manually add Gaussian noise to input, and the standard deviation (sigma) of the Gaussian should be in a reasonable range.
    - If the image is uint8, the sigma should be about 40-100.
    - If the image is float32, the sigma should be about 0.2-0.4.

See suggestions below.

* + **Create** a training model. One network structure you could try is the U-Net structure in Assignment 3.
  + **Train** your model.
  + Test your model on the **test dataset**, and **visualize** some results, including input noisy images and denoised output.

The trained model should have a reasonable performance on testing images, that is, the output images should have less noise than the input and no bad artifacts are above. Here is one example:

A number in a row

Description automatically generated with medium confidence

Here are some suggestions that you can try (not requirement):

* To access images in the MNIST dataset, you may need to modify the dataset loader. Following function may be helpful:

|  |
| --- |
| **To create a dataset from input\_tensor and ground\_truth\_output\_tensor:**  dataset = torch.utils.data.TensorDataset(input\_tensor, ground\_truth\_output\_tensor)  dataloader = DataLoader(dataset, batch\_size=batch\_size)  **To assess the tensor from a dataset:**  training\_data.train\_data  test\_data.test\_data  **You can try:**  print(training\_data.train\_data.shape)  **To add a noise to an image, you can try:**  np.random.normal  To convert the uint8 image to floating  To access an image in the dataset, you can try |

* To create a training dataset, you need to add some synthetic noise to images by yourself:

|  |
| --- |
| **To add a noise to an image, you can try (sigma is :**  noisy\_images = clean\_images + sigma \* np.random.normal(num\_samples, h, w)  Also, it may be helpful to convert uint8 images to float32, like:  im = im.astype(np.float32) / 255 |

* The suggested lose function is suggested to use MSE loss function:

|  |
| --- |
| loss = torch.nn.MSELoss() |

* You can refer to this example about how to create a network:

<https://pytorch.org/tutorials/beginner/introyt/modelsyt_tutorial.html>

* Some useful function for create the network:

|  |
| --- |
| **Upsampling:**  nn.ConvTranspose2d  nn.Upsample  **Concatenation:**  nn.cat |

**Optional bonus project. 3D Reconstruction.**

In this project, you are asked to generate a 3D geometry from a set of 2D images.

**Requirement for selecting this one:**

* You are very familiar with Linux system. You need to install some libraries on Linux yourself. Your TA may not be able to help you on that.
* You are very familiar with 3D geometry and neural radiance field (check tutorial ).

**Requirement for selecting this one:**

* **Step 1. Install** Colmap and NeRFStudio on your Linux machine. Please refer to this documentation for more details:

<https://docs.nerf.studio/quickstart/installation.html>

* **Step 2. Capture** a set of images of a static object, like a bag, from different angles. **Suggestions**: try objects with little reflection and try as much angles as you can. You cannot use online dataset. Capture by yourself.
* **Step 3**. Use NeRFstudio to reconstruct the 3D geometry of this object and render a new video. It should look like the videos shown in the first row:  
  <https://www.matthewtancik.com/nerf>
* **Step 4.** Try step 2 and 3 on a outdoor building structure, like pavilion.
* In your submission, please include:
  + Your captured images. If it is too large to upload, you could optionally put them on an online drive and share the link.
  + The rendered videos.
  + A report about your work. It should include **a screenshot of the geometry you reconstructed in NeRFStudio.**
  + You do not need to submit any code for this project.