Coding test

Write code in python with either the pytorch or tensorflow libraries in a **single** jupyter notebook to solve the 2 tasks below. Submit the pdf of the already run notebook, so that no training is needed and all results are visible. Do not show a long print out of the loss values. Use plots and figures instead to show any important quantities. Try to explain all as much in detail as possible.
I will look at how you formulate the task and how you implement the ideas. Train your models until the performance is good enough in your opinion. Your judgement of what "good enough" is, will also be evaluated. Use COLAB if you do not have access to GPUs.

Content of the jupyter notebook:

1. Code to define and train the neural networks
2. Code to demonstrate the performance of the trained model on the **test set**
3. **Thorough visualization** of the performance of the trained model on the test set
4. **Thorough visualization** of the **quantitative** performance during training and on the test set
5. Comments to explain how the tuning and the final performance evaluation are done
6. Comments to explain your reasoning and what each function does.

Task # 1

Build and train a denoising autoencoder (with at least 4 encoding/decoding layers) for a dataset of 10 grayscale natural images (*e.g.*, images of houses, cars, persons) of size 64×64. During training use zero-mean Gaussian noise of varying amounts. Consider a meaningful maximum amount of Gaussian noise (your choice of what "meaningful maximum amount is" will be evaluated). Train until the performance of the model is really good in your opinion (your choice of what "really good" is will be evaluated). Evaluate the denoising autoencoder on a separate test set of images. Show the evaluations (see the content description above). Describe your observations in detail.

Task # 2

In this task we train a neural network to map images to a 3D point cloud, where each pixel is mapped to its corresponding 3D point in space. As training dataset use a split of any 3D dataset (pick one) such as BlendedMVS (`https://github.com/YoYo000/BlendedMVS`), or Matterport3D (`https://niessner.github.io/Matterport/#download`), or Megadepth (`https://www.cs.cornell.edu/projects/megadepth/`), just to name a few examples, and use a sufficient number of samples of the remaining part from the split as test set. Choose a meaningful split. The design of the network architecture is your choice. See examples in the literature that could be suitable and train your model from scratch. The data should allow you to build a dataset of image and corresponding 3D point cloud pairs. More precisely, the input image is defined as $x \in \mathbb{R}^{M \times N \times 3}$ and the output 3D point cloud is defined as $X \in \mathbb{R}^{M \times N \times 3}$, where M is the number of image rows and N is the number of image columns. In the image 3 refers to the number of color channels (RGB), while in the 3D point cloud X the dimension 3 refers to the number of coordinates of each 3D point. Evaluate the model on your test set. Show quantitative evaluations (the choice of the metric is up to you and I recommend looking up in the literature to see what is typically used) and also qualitative evaluations by showing useful examples and by illustrating important observations on how (well or poorly) the model performs. Describe each step of the implementation directly with the code (as in Task 1) and also provide comments on your evaluation.