Computer Vision and 3D Image Processing
Final Projects

General Instructions

- **Grading.** Each project is for 20 points. The final projects are for a total of 60 points, which translates to **5** *points* out of **10** of your final exam.
- **Report.** The report must be written in questions *then* answer style format. This means that you put the question first and then write the answer. Do not mix your answers across different parts of the report. It is preferable to write in a standard LaTeX template. Don't use IEEE paper style or other weird templates! You will get an additional $1pt$ in each project if it is a high-quality report. Each of the projects can have upto 8 ± 2 pages. Note that for individual projects, the number of pages may vary, check its description.
- **Code.** Your code must readable and should include comments. Instructions on running the code must be provided.
- **Submission.** The final projects are due on Sunday night at 23:59 hrs, **22.01.2023**. You are required to upload all the necessary files and your report on each project separately to the respective assignment section of that project on Canvas.
- **Plagiarism**
  - We are well aware of online resources that are very similar to projects. It is NOT allowed to copy-paste code/material directly and use it in your report. It is fine to use them as a reference and develop an understanding. However, the work must be independent. Cite your sources if you use something.
  - Discussion between your peers is allowed as long as code and other materials are not shared.
  - Any form of detected plagiarism will result in all projects receiving zero points.
- **Oral Exam.** You need to book an appointment with Egor for the oral exam.

Fundamentals of Neural Networks

This project covers the basics of neural networks and its working. You will implement a basic neural network and study the impact of different loss functions.

1 Multi-Layer Perceptron

In this project, you will implement a Multi-Layer Perceptron (MLP) from scratch to classify digits in the MNIST dataset[1]. Your layers must have both forward and backward pass implementations. Do **NOT** use inbuilt functions from PyTorch[2], or other similar libraries for the implementation of the following. You are allowed to load the MNIST dataset in any way of your choice.

1.1 Implementation

- Fully-connected layer with bias. $1pt$
- ReLU and Sigmoid activations. $1.5pt$
- Mini-batch Gradient Descent (SGD) and Cross-Entropy loss. $1.5pt$

Create an MLP architecture of your choice using your implementation of the above. You can either choose ReLU or Sigmoid activation for the MLP but working implementation of both must be provided. If you are not successful with the implementation using Python and Numpy, implement the MLP in PyTorch to answer the following questions.

1.2 Training and testing

1. Train the MLP and report the training and test accuracies. Explain why there are differences in training and test performance? $1pt$
2. Calculate the number of trainable parameters of your model (you can implement it or calculate by hand) and explain how you have estimated it? $1pt$
3. How would the number of parameters vary if you use a convolution layer with batch normalization layer instead of a fully connected layer (explain)? What is the relation between a convolution and fully-connected layer, when are they the same/different? $1pt$

4. Use the knowledge acquired in the course or sources online to improve the performance on the MNIST dataset. Try to achieve an accuracy of more than what a standard CNN can achieve (say more than 98.5-99.0%). You can use PyTorch for this question.

   Hint: Try data augmentations, different networks and/or losses. Check for tricks online!

   (a) Explain in detail the improvements and changes that you have added. Well reasoned answers will get more/full points. $2.0pt$

   (b) Given that you have unlimited resources for computation and no extra data, how would you improve performance on the MNIST dataset? $1.0pt$

2 Loss functions

There are several loss function that are often used in Convolutional Neural Networks (CNNs). Depending on the objective, different loss functions can be used. In this part of the project, you will derivative the gradients of these loss functions with respect to the outputs y or logits z. The softmax function is given as

$$y = \sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}.$$

Cross-Entropy Loss

$$\mathcal{L}_{ce}(y, t) = -\big(t \cdot \log(y) + (1-t) \cdot \log(1-y)\big) \tag{1}$$

Dice Loss function [3]

$$\mathcal{L}_{dice}(y, t) = 1 - \frac{2yt}{y+t} \tag{2}$$

Focal Loss

- Generic version of Focal Loss [4].

$$\mathcal{L}_{focal}(y, t) = -(1-y)^{\gamma}\, t \log(y) \tag{3}$$

Using the above equations, answer the following questions.
Hint:

$$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial y} \cdot \frac{\partial y}{\partial z}$$

1. Derive the derivative of the output y with respect to logits z. $1pt$

2. Use the condition when $i = j$, and derive the derivative of the loss function $\mathcal{L}_{ce}$ with respect to the logits z. $1pt$

3. Show that

$$\frac{\partial \mathcal{L}_{dice}}{\partial z} = \frac{(y-1)}{2y} \cdot (1 - \mathcal{L}_{dice})^2 \tag{4}$$

$1.25pt$

4. Expand the generic to the binary version of focal loss. $0.25pt$

5. Show that

$$\frac{\partial L_{focal}}{\partial y} = \frac{-1}{y \cdot (1-y)} \Big[t \cdot (1-y)^{\gamma} \cdot (\gamma \cdot \mathcal{E}_y + 1 - y) + (t-1) \cdot y^{\gamma} \cdot (\gamma \cdot \mathcal{E}_{1-y} + y) \Big] \tag{5}$$

when $\mathcal{E}_p = -p \cdot \log(p)$ $1.5pt$

6. Under what condition is

$$\frac{\partial L_{focal}}{\partial y} = \frac{\partial L_{ce}}{\partial y}$$

What happens when γ value is too high or low in focal loss? Explain in detail how it would impact performance? $0.5pt$

7. List the applications where each loss function would be useful and explain why? $0.5pt$

8. Discuss (in detail) the behavior of precision and recall for each of the listed loss functions above. Use the proofs from questions 2, 3 and 5 to support your answers. $3pt$

$$\text{precision} = \frac{T.P}{T.P + F.P}, \quad \text{recall} = \frac{T.P}{T.P + F.N}, \tag{6}$$

where *T.P, T.N, F.P, F.N* are true positives, true negatives, false positives and false negatives respectively.

Label	Prediction	Condition
1	1	True Positive
0	1	False Positive
1	0	False Negative
0	0	True Negative

Tip: If you are using LaTeX, this command will be handy to write your report.

```
\newcommand{\dpartial}[2]{\frac{\partial #1}{\partial #2}}
```

Using `\dpartial{x}{y}` produces

$$\frac{\partial x}{\partial y}$$

Anomaly Detection with Autoencoder

Introduction

Autoencoders have applications in computer vision and image editing. They are commonly used for dimensionality reduction, image denoising, image generation or anomaly detection use cases. As shown in Figure 1, it consists of 3 components: encoder, code and decoder. During the training process, the encoder learns to encode the input into a particular n-dimensional space, and the decoder learns to decode the encodings and tries to reconstruct them back to their original form. It compresses the input into a lower-dimensional code and then reconstructs the output from this representation.

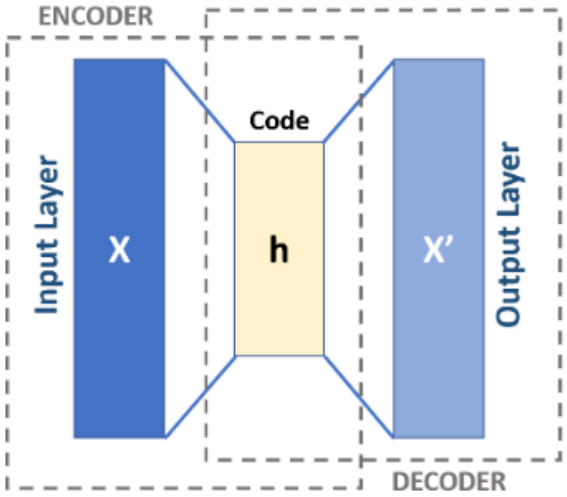


Figure 1: Autoencoder architecture.

In this project, you will implement, run and analyze an autoencoder architecture from scratch for anomaly detection purpose on MNIST dataset. The MNIST dataset contains thousands of images of handwritten digits and is commonly used for classification purposes. However, in this project, you are going to use it for outlier (anomaly) detection. We strongly recommend using the PyTorch [2] to train and evaluate your network.

1 Implementation

Implement simple autoencoder architecture in PyTorch by using linear layers (without convolutional layers).

- Encoder and decoder parts. $2pt$
- Mean Squared Error (MSE) loss function. $0.5pt$
- Stochastic Gradient Descent (SGD). $0.5pt$
- The rest of the code (training, data loader etc.) $1pt$

2 Questions

In this section, an overview of requirements and questions to be answered for your report is provided.

1. Take a subset (10%) of the MNIST test dataset and add excessive noise to corrupt images. Then, visualize four samples for original and corrupted ones. Note: The type of noise is optional. Be sure that the amount of noise is enough for finding outliers (anomalies) in question 8 with the help of the threshold value. $1pt$
2. Train your autoencoder architecture by using the MNIST training dataset. Compare the autoencoder output generated from corrupted input images with original and corrupted images. Comment on the results. What did you expect, and what did you observe? What are the reasons for similarities and differences? $1pt$
3. What happens if your dataset includes corrupted images not only in the testing dataset but also in the training dataset? How does it affect the performance of the model? $1pt$
4. Change the loss function of MSE with SSIM loss function. Explain the difference between them and compare the results. Please keep your loss function as MSE for the following questions. $1pt$
5. Implement more complex autoencoder architecture to obtain better results. Compare the results with the first architecture you implemented and comment on the results. $1pt$
6. Implement the convolutional autoencoder model. Compare your results with the basic and complex architectures you implemented and comment on the results. $2pt$
7. Choose the best implementation of them by explaining the reason for that decision. Plot the training and validation loss functions of your architecture to epoch numbers. What is over-fitting? How can one detect and avoid over-fitting? $2pt$

8. Decide a threshold value in reconstruction error for detecting outliers (corrupted images) by drawing the loss distribution. This threshold value needs to be used to determine whether the input sample is an anomaly or not. Then, explain why you decided on that threshold value by comparing it with other threshold values. $1.5pt$

9. Provide true positives(TP), true negatives(TN), false positives(FP), and false negatives(FN) values for the threshold that you decided in the previous question. Plot the confusion matrix. $1.5pt$

10. Calculate precision, recall, and F1 score. Explain in detail how you can improve the F1 score. What is the effect of reconstruction error threshold value for precision and recall? $1pt$

11. What are the ROC curve and AUC? What is the difference between them? How can you use them to improve your model performance? $1pt$

12. What is the false alarm rate? Calculate the false alarm rate for this problem. Explain in detail the importance of false alarm rate for anomaly detection problems and what should be the ideal value for that. $1pt$

13. What are the cons of autoencoder structure? What could be another solution for the outlier detection problem? Explain it in detail. $1pt$

Process these items in your report. Additionally, hand in the code of your implementation, including a checkpoint file of your fully trained network. If your submitted code does not run, we can not grade your report, so make sure you hand in a version that actually runs.

3D Point Cloud Filtering

Introduction

In this project, you will design your own Robot Operating System (ROS) node to filter a noisy cumulative point cloud. It is recommended to use the included virtual machine, as ROS is pre-installed and the required dataset is ready to use. For assistance regarding the virtual machine and ROS please refer to the section "Instruction of Virtual Machine and ROS".

1 Point cloud importing

You have learned how to play back the point cloud topic from the section of "Instruction of Virtual Machine and ROS".

1. The video that used to generate this point cloud was recorded from ZED Mini (a stereo camera). Recall what you learned in the lecture, could you briefly describe the process of point cloud generation using a stereo camera? $2pt$

2. As you can see in RVIZ, this is a very noisy point cloud. Could you explain the reason why there are so many noises? $2pt$

3. Before designing the filtering node, your node first needs to subscribe to the point cloud topic. Recall what you have learned from the "listener.py" example. Try to create a node that subscribes point cloud topics. $3pt$

2 Design your filtering node in ROS

Now you have a noisy raw point cloud as input. It is time to design your own filtering node. The filtered point cloud should be sufficient for us to recognize the main structures of the reconstructed building, for example, walls, windows and doors. We are not very interested in furniture, ceiling and floor (since this is a single floor building). You are free to code in Python or C++.

1. Design your filtering node and record your filtered point cloud in a .bag file. Don't forget to show images of the complete filtered point cloud in the report. $5pt$

2. Explain details of your filtering procedure. You may also draw a flow chart to support your text. $3pt$

3. Do you have any idea on how to further improve your filtering node in terms of speed and performance? Explain your idea here. $2pt$

3 Evaluation

For this project, you have to send your code and write a report (6 pages single column maximum) addressing the following points:

1. Your answers/explanations for each question:
   - 1.1 Describe the process of point cloud generation using a stereo camera.
   - 1.2 Explain the reason why there are so many noises in the raw point cloud.
   - 2.2 Explain details of your filtering procedure.
   - 2.3 Explain ideas on how to further improve your filtering node.
2. Use rosbag commend to record your filtered point cloud. The bag file should be sent as a link (Google Drive, Dropbox, etc.) to the compressed file.
3. Figures depicting the intermediate steps and results should also be included in the report.
   - 1.1 A figure of the complete raw point cloud.
   - 2.1 A figure of the complete filtered point cloud.
4. Points will also be awarded based on the overall quality of the report:
   - Adddressing all topics. $1pt$
   - Quality of the filtered point cloud. $1pt$
   - Readability, figures and use of language. $1pt$

Instruction of Virtual Machine and ROS

Overview

In the "3D Point Cloud Filtering" project, you will design your own Robot Operating System (ROS) package to filter a noisy cumulative point cloud. It is recommended to use the provided virtual machine, as ROS is pre-installed and the required dataset is ready to use. In this section, you will find out how to install and use the virtual machine. Besides, we also provide exercises that let you get familiar with ROS.

1 Installing Oracle VM VirtualBox

We will use the Oracle VM VirtualBox application for "3D Point Cloud Filtering" project. The application can be downloaded from here: https://www.virtualbox.org/wiki/Downloads

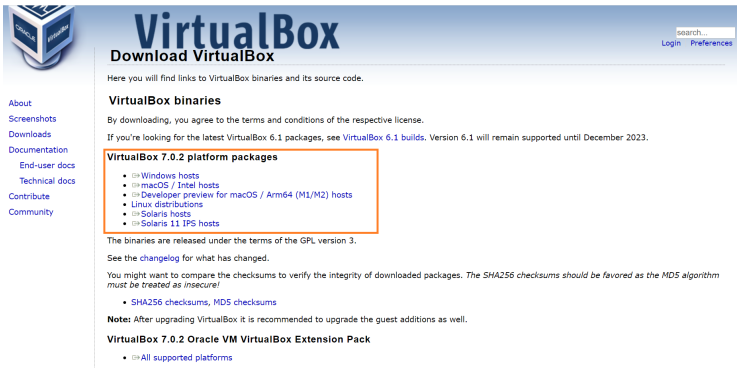
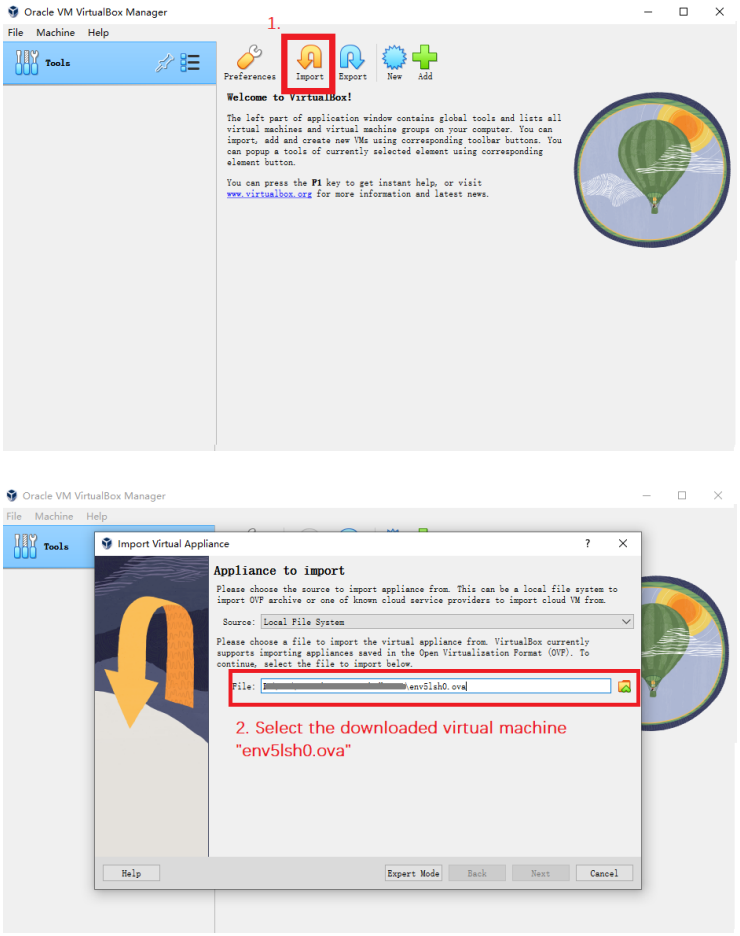


Figure 2: Oracle VM VirtualBox download page.

Download the latest version (7.0.2) and then install the application by following the setup guidance.

2 Importing the Virtual Machine

Now you need to download and import the virtual machine "env5lsh0.ova". You can import the virtual machine by following the steps:



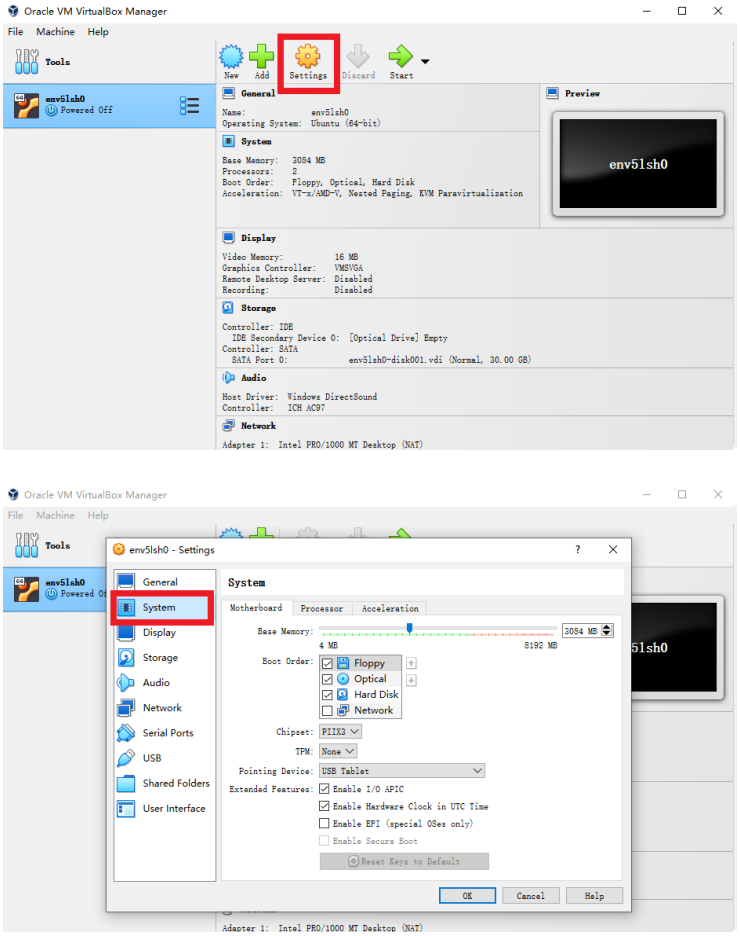
3 Virtual Machine Settings

It is recommended that before running it, you check the settings and adapt them to your local machine (i.e. RAM, CPUs>=2)

Now you should able to "Start" the virtual machine. The password is: 5lsh0

4 Exercises

In these exercises, you will learn how to create a simple ROS node, recording and playing back data. We strongly encourage you to finish these exercises before starting the "3D Point Cloud Filtering" project.



4.1 Create a publisher and subscriber

We have installed ROS in the provided virtual machine. In order to get familiar with ROS, you need to create your own publisher and subscriber ROS node step by step. You will create a node called "talker" that publishes "hello world" topics. Then another node "listener" subscribes to the published topics and prints them.

*In this website (http://wiki.ros.org/ROS/Tutorials#Beginner_Level), you can find many ROS tutorials. We encourage you to go through all beginner-level tutorials.

1. First, open a terminal and let's create and build a ROS workspace "test_ws". Details can be found in the tutorial "Installing and Configuring Your ROS Environment".

- mkdir -p ~/test_ws/src

- cd ∼/test_ws/
- catkin_make

2. Then create and build a catkin package "test_package". Details can be found in the tutorial "Creating a ROS Package".

- cd ∼/test_ws/src
- catkin_create_pkg test_package std_msgs rospy roscpp
- cd ∼/test_ws/
- catkin_make

3. Download the example scripts to the src folder of the created package and make them executable. Details can be found in the tutorial "Writing a Simple Publisher and Subscriber (Python)".

- cd ∼/test_ws/src/test_package/src
- wget https://raw.github.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/talker.py
- chmod +x talker.py
- wget https://raw.github.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/listener.py
- chmod +x listener.py

4. Before running, you need to source your workspace's setup.sh file.

- cd ∼/test_ws/
- source ./devel/setup.bash

5. Open three terminal windows and test your ROS nodes. Details can be found in the tutorial "Examining the Simple Publisher and Subscriber"

- First window: roscore
- Second window: rosrun test_package listener.py (Now the listener waits for published ROS topics)
- Third window: rosrun test_package talker.py (Topics are publishing! Check the window of the listener.)

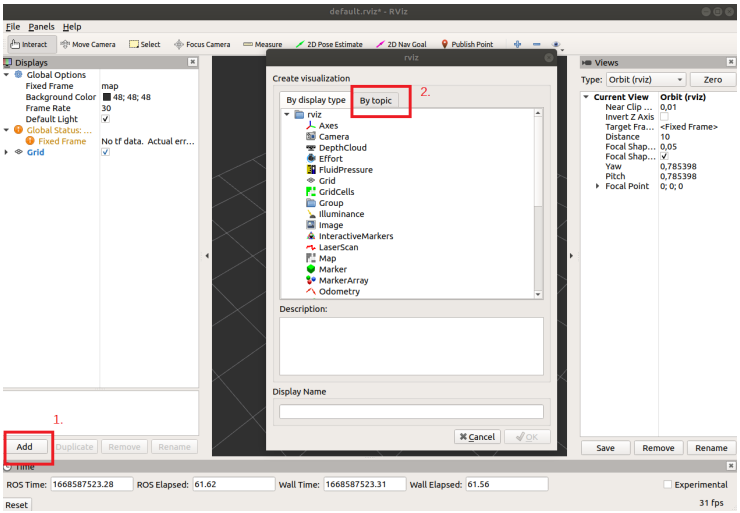
4.2 Recording and playing back data

In ROS we always record data from a running ROS system into a .bag file. Then play back the data to produce similar behavior in a running system.

1. Let's first run the talker node. What is the name of the published topic?

2. Can you find a proper way to record topics? Then try to play back the recorded .bag file. (Make sure that a roscore is up and running)(http://wiki.ros.org/rosbag/Commandline)

3. Later in the "3D Point Cloud Filtering" project, you will need to filter a pre-recorded point cloud topic. The "Raw_Point_Cloud.bag" file is located in the Download folder. The topic name is "/cloud_map". Try to play back the point cloud topic and visualize it using RVIZ (a 3D visualization tool for ROS). **You may need a new terminal window and then run "rosrun rviz rviz". Then add "/cloud_map" topic (steps are shown in the figure below).



References

[1] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[2] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[3] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 240–248.

[4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.