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, **23.01.2022**. 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. 2.0*pt*

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

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. 1*pt*

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

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

Image retrieval and re-identification

An image retrieval system is a computer system for browsing, searching and retrieving images from a large database of digital images [5]. Image retrieval has been a prominent and challenging problem of computer vision with numerous applications. For instance, person re-identification is a sub-problem of image retrieval that tries to achieve identity matching among non-overlapping cameras. The ultimate goal there is to track persons in a multi-camera network, which is substantially more difficult than tracking a person within a single camera frame.

1 How Retrieval Works

In image retrieval tasks, there exists a database of possible matches, which is called a *gallery*. For person re-ID, this is a database of known people. A sample that needs to be matched at test time (e.g. an unknown person), is commonly called either a *query* or a *probe*. The process of retrieval can be visualized in Figure 1, for the sub-problem of person re-identification. In more detail, our gallery set consists of bounding-box images of people whose identities are known. Note that, in this context, *identity* refers only to an assigned number. At test time, the person re-ID system receives an image whose identity is unknown. Then, the algorithm processes it and returns the *gallery* images ranked by their similarity to the given *query*. The identity of the *gallery* image at the top of this list, i.e. the most similar image to the *query*, is selected as the identity of the *query* person.

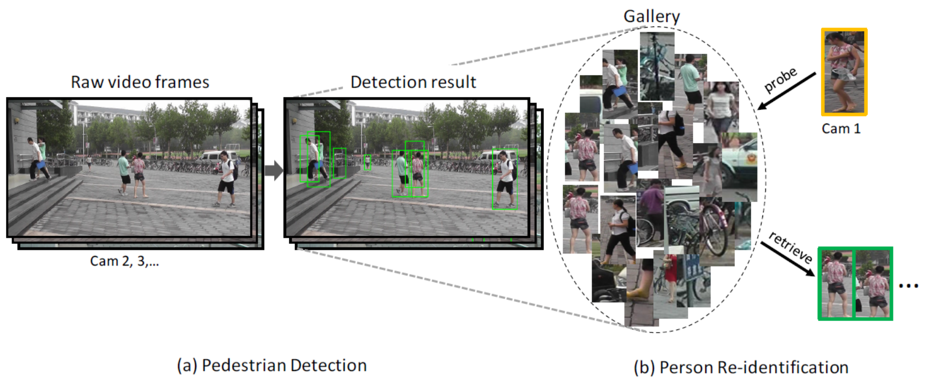

Figure 1: Depiction of person re-identification [6].

2 Network Architecture

In this project, you will implement, run and analyze a deep learning-based image retrieval algorithm that focuses on the person re-identification (re-ID) application. More specifically, you are going to train a simplified neural network on the MNIST dataset (to limit the amount of time spent on training). The MNIST dataset contains thousands of images of handwritten digits and is commonly used for classification purposes. However, in this assignment, we are going to use it for person (digit) re-identification instead. We recommend using the PyTorch Framework [2] to train and evaluate your network.

The network you are required to train is defined as follows, where all convolutional layers use stride /1, ensure by zero-padding that the spatial dimensions of the input and output feature maps remain the same:

Table 1: Structure of the network to train

Layer name	**Layer definition**	**Output size**
conv1	5x5, 16	28x28x16
	ReLU	28x28x16
	2x2 max pool, /2	14x14x16
conv2	3x3, 32	14x14x32
	ReLU	14x14x32
	2x2 max pool, /2	7x7x32
conv3	3x3, 32	7x7x32
	ReLU	7x7x32
	global max pool	1x1x32
	2-D FC	1x1x2
	10-D FC	1x1x10

Notice the final 10-dimensional fully-connected (FC) layer. The output of this layer is used to apply one-hot encoding, which results in a prediction of the class (digit) of the input image. The output of the 2-D FC layer will act as the feature vector of the input image (i.e. the embedding of the input image). The previously mentioned ranking of the *gallery* set by the similarity to the given *query* is measured using this feature vector. That is, the *gallery* image with the smallest distance between its feature vector and that of the *query* image is the most similar one.

To train the network, you will have to use the train set of the MNIST dataset. To evaluate the performance of your trained network, you can use the train set of the MNIST dataset as your gallery and the test set of the MNIST dataset as your query set. More details will follow in the next two sections.

3 Training the Network

During this phase, you will have to use the train set of the MNIST dataset. As is usual in deep learning, you will distribute this dataset in batches to the network, apply it, calculate the loss and apply backpropagation. After completion of each epoch, you are asked to visualize the embedding space, as created by the network, of a random subset of the training set. So, in other words, apply your fully trained network on each image in your subset to plot their feature vectors. Recall that the feature vectors are 2-D, so the result is a simple 2-D plot. To prevent clutter, a small dot per image is sufficient, where each digit has a different colored dot. Furthermore, to make the visualization more meaningful, make sure you only randomly select a subset of training images once, such that you can see how the same set of images move in embedding space as training progresses. 20 samples of each digit should be sufficient.

You can use the following code for plotting the points:

```python
# Plot and show
colors = matplotlib.cm.Paired(np.linspace(0, 1, len(all_points)))
fig, ax = plt.subplots()
for (points, color, digit) in zip(all_points, colors, range(10)):
    ax.scatter([item[0] for item in points],
               [item[1] for item in points],
               color=color, label='digit {}'.format(digit))
    ax.grid(True)
ax.legend()
plt.show()
```

4 Performance evaluation

Once the network has completed training, we need to evaluate its performance. In this project, you will measure both its classification performance and its re-ID performance. First, to determine the classification performance, apply your network on the test set of the MNIST dataset and determine for each digit the fraction of images that is correctly classified. Second, to measure the re-ID performance, you can use the training set of the MNIST dataset as the gallery and its test set as the query set. Then, for each query, determine its feature vector distance to each gallery image, and rank the gallery accordingly. Finally, use the 20 most-similar gallery images to determine the fraction of gallery images that are of the correct digit, i.e. the top-20 re-ID performance. Typically, in person re-ID literature there are more advanced performance metrics used, but that is not required for most of this project. There is just a single task on this topic, which is described in the next section. Once you have implemented everything so far, make sure that, for each iteration of the training process, you print both the loss value and the classification accuracy of the batch. And make sure that, for each epoch, the classification accuracy of the whole test set and the top-20 re-ID performance is printed. Once you have a fully trained network you are asked to plot these for your report.

5 Questions

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

1. Explain the relationship between the batch size and learning rate. What is over-fitting? How can one detect and avoid over-fitting? 1*pt*

2. Why is the final network layer 10-dimensional? What is the difference between the outputs of the 2-D layer and the 10-D layer? Why is the 10-D layer required at all? 1.5*pt*

3. Why do we need the presence of disjoint training and test sets? And what would happen if we use the MNIST test set as both the gallery and query set? 1*pt*

4. What is the difference between classification and image retrieval/re-ID? And why can we train a network for image retrieval using a loss for classification? 1.5*pt*

5. Provide the following graphs of your training in your report: 2*pt*

   - Batch loss and batch classification accuracy (y-axis) for each training iteration (x-axis).
   - Classification accuracy on the whole test set and the top-20 re-ID performance (y-axis) for each training epoch (x-axis).

   Include your final classification and re-identification accuracy numerically in your report too.

6. Include the embedding space visualization you made during training for a subset of train images after each epoch. You don't have to include the visualizations of each epoch, one before training commences, one after the first training epoch, one in the middle of the training process and one after the final training epoch should be enough. Comment on the changes you observe as the training progresses. 2*pt*

7. Using this visualization, can you conclude that your network is properly trained? Why? 1*pt*

8. Comment on the performance difference between the achieved classification and retrieval performance (top-20 re-ID performance). 1*pt*

9. Use your own handwriting to write digits from 0-9. Change your own hand-written digits into MNIST format and run inference on them using your trained network. For each hand-written digit, show the image in your report, including the top-five matches that your network returned for that digit. Additionally, plot the embedding space as before, but also including the embeddings of your own digits. 3*pt*

10. Determine and provide the re-ID performance of your trained network using the mAP metric as well. Make sure you use the proper mAP metric for the re-identification task, as there are different adaptations of the same metric to other problems. $1pt$

11. Increase the embedding dimensionality and report on differences in performance. Apply PCA to reduce the embedding dimensionality back to 2-D, such that you can plot the embedding space in a similar 2D plot (you are allowed to use a third-party implementation of PCA for this). Include these in your report too. $2pt$

12. Implement and train the network using the triplet loss and comment on its performance. Although it is the common practice, you may skip the hard-triplet mining for this task. $2pt$ (bonus)

13. For well-structured nicely readable reports that explain all necessary details for your implementation and justification to your answers. $1pt$

Process these items in your report (max. 6 pages). Additionally, also 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 assignment, you will design your own Robot Operating System (ROS) package to filter a noisy cumulative point cloud. It is recommended to use the included virtual machine image, as ROS is pre-installed and the required dataset is ready to use. For assistance regarding the installation of the virtual machine please refer to the appendix.

1 Point cloud importing

In the first activity, you will need to get familiar with 'rosbag' commands since you need them for point cloud importing and the final assignment submission:

1. The "Raw_Point_Cloud.bag" file is located in the Download folder. Use 'rosbag play /cloud_map' to read the raw point cloud. You can use RVIZ (a 3D visualization tool for ROS) to visualize the point cloud. This cumulative point cloud was generated by RTABMap (An open-source SLAM baseline). Wait until the point cloud is complete, then add a figure of this raw point cloud in the report. $1pt$

2. The video that used to generate this point cloud was recorded from ZED Mini (a stereo camera). Recall what you learned in class, could you briefly describe the process of point cloud generation using a stereo camera? $3pt$

3. As you can see in RVIZ, this is a very noisy point cloud. Could you explain the reason why there are so many noises? $3pt$

2 Design your filtering package in ROS

Now you have a noisy raw point cloud as input. It is time to design your own filtering package. 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). To do this, you need to set up your own catkin workspace and then build your filtering package (You can follow tutorials: http://wiki.ros.org/cn/catkin/Tutorials/create_a_workspace). You are free to code in Python or C++.

1. Design your filtering package and use 'rosbag record /XXX' to record your filtered point cloud in a .bag file. Don't forget to show images of the complete filtered point cloud in the report . 5*pt*

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

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

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:
   - 1.2 Describe the process of point cloud generation using a stereo camera.
   - 1.3 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 idea on how to further improve your filtering package.

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. 1*pt*
   - Quality of the filtered point cloud. 1*pt*
   - Readability, figures and use of language. 1*pt*

4 Appendix

4.1 Virtual image installation

We will use Oracle's Virtual Box. Download the virtual machine image file 'env5lsh0'. You can import this into VirtualBox using the 'Import appliance' command. It is recommended that before running it, you check the settings and adapt them to your local machine (i.e. RAM, CPUs¿=2). You also need to install the 'VM Extension Pack'. In case you need to install anything, the password is '5lsh0'.

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.

[5] Image retrieval, “Image retrieval — Wikipedia, the free encyclopedia,” 2010, [Online; accessed 20-November-2019]. [Online]. Available: https://en.wikipedia.org/wiki/Image_retrieval

[6] L. Zheng, H. Zhang, S. Sun, M. Chandraker, Y. Yang, and Q. Tian, “Person re-identification in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1367–1376.