Hong Kong University of Science and Technology
COMP 4211: Machine Learning
Fall 2021

Programming Assignment 2
Due: 22 October 2021, Friday, 11:59pm

1 Objective

The objective of this assignment is to practise the use of the `PyTorch` machine learning framework through implementing different methods for person re-identification (ReID) based on the residual network (ResNet) which is a widely used convolutional neural network (CNN) architecture. Throughout the assignment, students will be guided to develop the ReID model step by step and study the effects of different model configurations.

2 Major Tasks

The assignment consists of a coding part and a written report:

Coding: Build a person ReID model using `PyTorch`.

Written Report: Report the results and answer some questions.

The tasks will be elaborated in Sections 4 and 5 below. Note that [Qn] refers to a specific question (the nth question) that you need to answer in the written report.

3 Setup

- Make sure that the libraries `torch`, `torchvision`, `tensorboard`, `matplotlib` and `scikit-learn` have been installed in your machine or programming environment.
- For this assignment, it suffices to use only `PyTorch` with `Python 3.7` or above. (Specifically, `PyTorch` versions 1.7.1 and 1.9.0 have been verified to work well for this assignment.) Other machine learning frameworks including `TensorFlow` and `Keras` should not be used.
- You are highly recommended to use GPU resources like those provided by the GPU servers of the Department of Computer Science and Engineering (`https://cssystem.cse.ust.hk/Facilities/ug_cluster/gpu.html`) or the Google Colab (`https://colab.research.google.com`).
- The dataset files for this assignment are provided as a ZIP file (`pa2_data.zip`).

- imgs
  - train
    - 0
      - 1.jpg
      - …
    - …
    - 229
      - 1.jpg
      - …
  - val
    - 230
      - 1.jpg
      - …
    - …
    - 278
      - 1.jpg
      - …

- val
  - gallery
    - 1.jpg
    - …
  - query
    - 1.jpg
    - …
  - gt.csv

- test
  - gallery
    - 1.jpg
    - …
  - query
    - 1.jpg
    - …
  - pred.csv

- train_idloss.csv
- val_idloss.csv
- train_random_triplet.csv

Figure 1: Overall structure of the `pa2_data` directory.

4 Person Re-identification

ReID is one of the fundamental computer vision tasks and is widely used in many applications, such as autonomous driving and video surveillance. ReID is a task that retrieves the target object among the candidates based on the given query object. This can be achieved by measuring the similarity between the given query object and the candidate objects and choosing the objects with high similarity. We will evaluate the performance of our ReID model with **Rank-1 (R1), Rank-5 (R5) and Rank-10 (R10)** accuracy measures, where R5, for example, indicates whether the target object is in the list of top-5 similar objects. In this programming assignment, we aim to perform person ReID using the video sequences extracted from the MOT Challenge.[1] We will use **cosine similarity** as the measure to quantify the similarity between objects.

The structure of this assignment consists of three main parts:

1. Build a modified ResNet backbone network (Section 4.2).
2. Train the ResNet with the cross-entropy loss based on a person's ID (Section 4.3).
3. Train the ResNet with the triplet margin loss based on the relationships between an anchor and a positive example as well as a negative example (Section 4.4).

4.1 Dataset and Dataloader

The ReID dataset can be found in the `./pa2_data` directory. Figure 1 shows the overall structure of the directory. You are recommended to navigate through the folders to take a look at different types of files. There are three folders and three CSV files in the root directory. The `imgs` folder contains the images of each person. The number of a folder indicates the ID of the person. The `val` folder contains files for evaluating your trained ReID model. The `gt.csv` file has two columns: query and gallery. The gallery column contains the target image filename correspond-

[1]`https://motchallenge.net/data/MOT17/`

ing to the query image. In the `test` folder, you are asked to fill out `pred.csv`'s gallery column using your trained ReID model. The `train_idloss.csv` and `train_random_triplet.csv` files are used for training the ReID model. More details can be found in Sections 4.3 and 4.4, respectively.

[C1] Your first task is to implement a custom `RetrievalDataset` class for the folders `val` and `test` so that you can make inference using the ReID model and measure the retrieval performance. Please refer to an example in Figure 2 which shows 100 resized and padded gallery images. The `RetrievalDataset.get_gallery_imgs()` function should return all gallery images ($[N{\times}3{\times}128{\times}128]$), where N is the number of gallery images. The `RetrievalDataset.getitem()` function should return a query image and its corresponding ground-truth gallery image index (only applicable to the validation phase) so that you can evaluate the retrieval performance. Also, you need to apply the following data transformation operations: [C2]

1. Convert an image to a tensor with the RGB channels in the range $[0.0, 1.0]$.
2. Normalize with mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$. Every three values correspond to the RGB channels.
3. Resize to make the longer side equal to 128 while keeping the aspect ratio unchanged.
4. Pad zero values to make the image size equal to 128×128.

[C3] For the dataloader of `RetrievalDataset`, we recommend you to set `shuffle=False`.

4.2 Residual Network (ResNet)

You have learned some popular CNN architectures in the course. Here, you are asked to implement a modified ResNet-18 architecture and use it as the backbone network of the ReID model. You will further practise how to load the ImageNet pretrained ResNet weights instead of using randomly initialized weights.

4.2.1 Build Modified ResNet-18

[C4, C5] The convolutional (2D) layers take 2D-shape data (with RGB channels in this assignment) and output a hidden state vector which is called a feature vector. Here you need to build a modified ResNet-18 according to the network architecture described in Table 1. The ResBlock and ResBlock-D are illustrated in Table 2. As you have learned in class, ResNet has a residual connection from the input to the output. Therefore, the Residual (Conv + BN) layer should be applied to Input and the Residual (Addition) layer adds two outputs from the residual connection and normal flow. Note that when you use `nn.Conv2d`, you should set `bias=False`.

[Q1] What is the shape of the output from each layer of Table 1 (in the format $[C{\times}H{\times}W]$)?

[Q2] What is the correct kernel size in the average pooling layer?

[Q3] What is an alternative way to compress the output feature map into a 1D vector, instead of using a pooling layer? There are many possible ways. Please explain any one of them in detail.

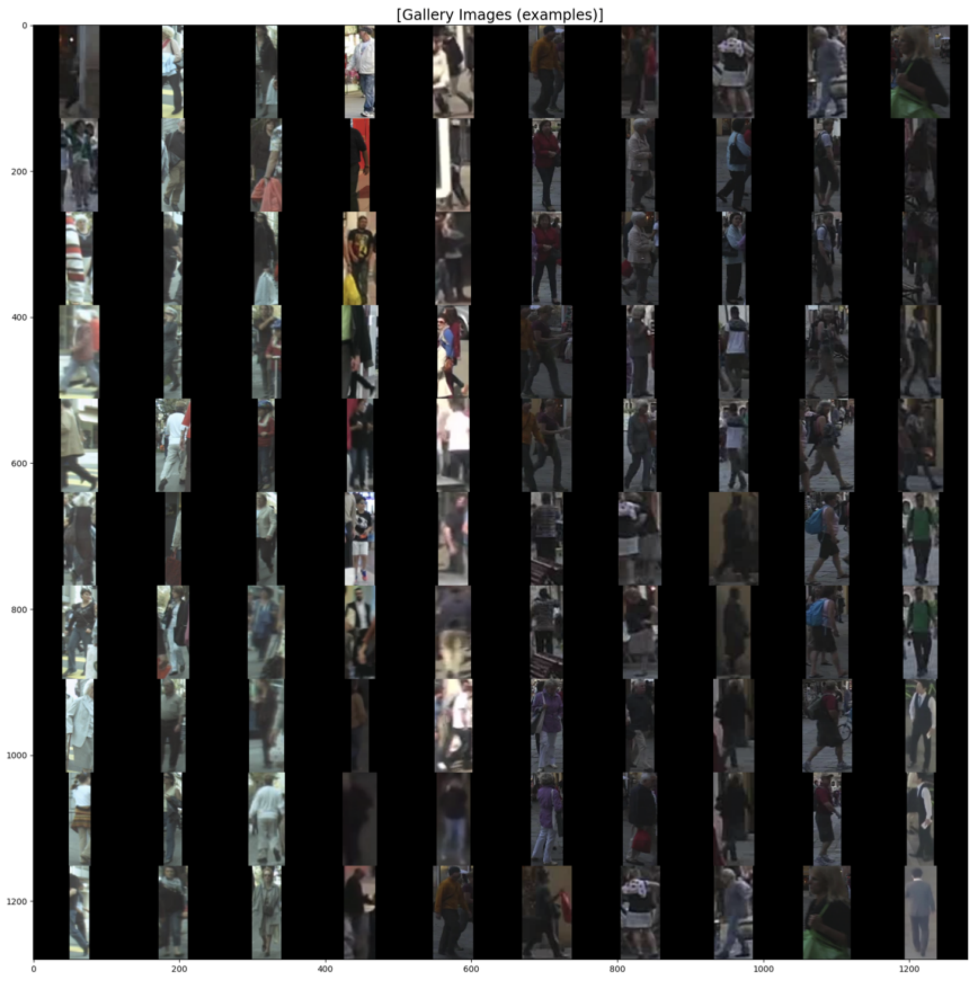

Figure 2: 100 gallery images which have been correctly resized and padded.

Table 1: Description of the modified ResNet-18 model.

Layer	Kernel size	# of kernels	Stride	Padding
Input	3×128×128 image			
Conv+BN+ReLU	7×7	64	2	3
Max pooling	3×3	-	2	1
ResBlock	-	64	-	-
ResBlock	-	64	-	-
ResBlock-D	-	128	-	-
ResBlock	-	128	-	-
ResBlock-D	-	256	-	-
ResBlock	-	256	-	-
ResBlock	-	512	-	-
ResBlock	-	512	-	-
Average pooling	?×?	-	?	0
Output	1×512 vector			
FCs	Will be covered in Sections 4.3 and 4.4. Please ignore it now.			

[Q4] What is the number of trainable parameters in the first ResBlock?

Table 2: ResBlock and ResBlock-D description with C_1 input channels and C_2 kernels. Residual (Conv+BN) is added only if $C_1 \neq C_2$. ResBlock-D uses stride 2 for Conv+BN+ReLU and Residual (Conv+BN).

Layer	Kernel size	# of kernels	Stride	Padding
Input	$C_1 \times H \times W$ feature map			
Conv+BN+ReLU	3×3	C_2	1 or 2	1
Conv+BN	3×3	C_2	1	1
Residual (Conv+BN)	1×1	C_2	1 or 2	0
Residual (Addition)	Element-wise sum of the results from the two rows above			
ReLU	-	-	-	-

4.2.2 Load ImageNet Pretrained Weights

[C6] In practice, we usually do not train a model from scratch but initialize it using pretrained weights. Here, you are required to initialize the modified ResNet-18 with ResNet-18's ImageNet pretrained weights. To do so, please refer to the code below.

```
import torch.utils.model_zoo as model_zoo
url = 'https://download.pytorch.org/models/resnet18-5c106cde.pth'
pretrain_dict = model_zoo.load_url(_url)
# TODO
```

[Q5] Which layers of your modified ResNet-18 model cannot load the pretrained weights? And, which layers of the pretrained weights are discarded?

4.3 Train with ID Loss

[C7] To train the model, we need to use an appropriate loss function which is related to the objective of the target task. Here, we train our model to predict the ID of a person. In the training dataset, you are provided images of 230 different persons. In other words, you need to train the model to perform a multi-class classification task (using `CrossEntropyLoss`). You will use two fully connected (FC) layers to further encode the visual features extracted from the ResNet-18 backbone. Additional FC layers should be appended to the modified ResNet-18 backbone as shown in the last row of Table 1. Please use 512 and 256 as the output dimensionalities of the two FC layers. **Batch normalization**, **ReLU activation** and **dropout** (prob=0.2) are appended to each FC layer (i.e., FC → BN → ReLU → Dropout). We call the finally encoded visual features as ReID features since these features are used for measuring similarity for ReID. However, to train your model with the ID of a person, one more FC layer should be added to predict the score of each candidate ID. For this FC layer, please do not add batch normalization, ReLU and dropout.

[Q6] What is the output dimensionality of the last FC layer?

[Q7] What is dropout and why do we use it?

[C8] The `train_idloss.csv` and `val_idloss.csv` files contain two columns: filepath and id. Please implement a custom dataset class `IDLossDataset` that reads `train_idloss.csv` in the

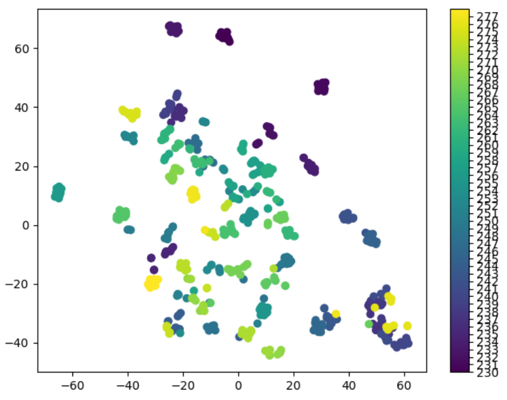

Figure 3: t-SNE visualization of the validation data.

training phase and `val_idloss.csv` in the validation phase, and then outputs the image of a person and its ID. The same data transformation operations used in Section 4.1 should be applied. However, in the training phase, additional data augmentations are required to prevent overfitting. [C9] You need to apply random horizontal flip (with $p = 0.5$) and random erasing (with $p = 0.5$ and scale= $(0.05, 0.2)$). Note that the order of random erasing is important. You should not erase the padded region which is meaningless.

[C10] Train the model using the Adam optimizer with a batch size of 128 for 50 epochs. In the validation phase, you need to report the validation loss per epoch based on **IDLossDataset** implemented in this section. [C11] Additionally, you need to report the R1, R5 and R10 accuracy measures every five epochs based on **RetrievalDataset** implemented in Section 4.1. Therefore, you are required to plot the following three graphs using `Tensorboard` or `matplotlib`: 1) training loss per iteration; 2) training/validation loss per epoch; 3) R1, R5 and R10 accuracy measures every five epochs in the validation phase. Please find the learning rate that can achieve an R5 accuracy above 75%.

[Q8] Attach the three graphs described above with the model that achieves the required R5 accuracy. And, report the final epoch's R1, R5 and R10. Please describe the trend of training/-validation loss and R1, R5 and R10.

[C12] Now, your model can effectively extract ReID features from an image to distinguish different persons. Examine them visually by mapping the high-dimensional features into 2D using t-SNE[2] and plotting using `matplotlib`. The `val_idloss.csv` file has been prepared for this task. You can use the `show_2d_tsne` function to generate a t-SNE plot. Moreover, **use your student ID as the `random_state` for t-SNE** and a small perplexity, e.g., 10. Figure 3 is an example of the visualization result.

[Q9] Attach your plot of the t-SNE result.

[P1] Finally, you complete developing the person ReID model and submit the result on the test set by completing `test/pred.csv`. Please complete the gallery column with the filenames of the top-10 similar persons for each query person. The filenames should be sorted by the cosine similarity score in descending order. Each filename should be separated by a hyphen (-) (e.g., `test/gallery/1.jpg-test/gallery/2.jpg-...-test/gallery/10.jpg`). Save the result file as `id_loss_pred.csv`.

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`

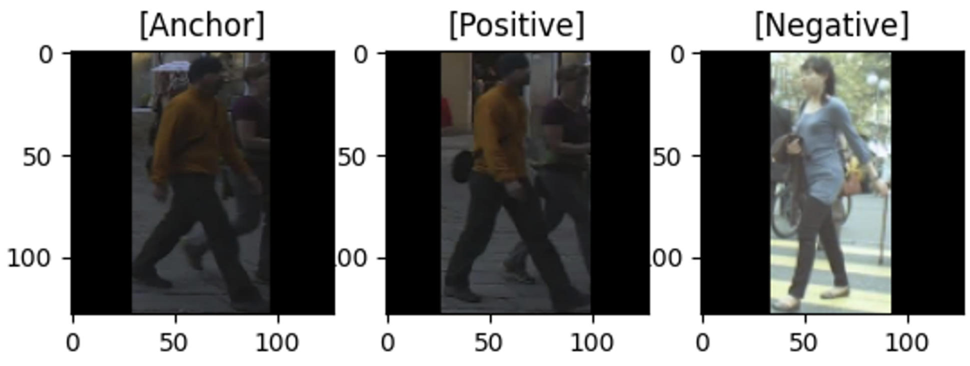

Figure 4: Three images for computing the triplet loss.

4.4 Train with Triplet Loss

Now, we use a different loss function to train the ReID model. In the previous section, we used a single image for each training example to compute the loss and train the model. In this section, we use three images instead. We define the triplet loss[3] based on triplets, each of which consists of an anchor, a positive image, and a negative image. As shown in Figure 4, the positive image shows the same person as that in the anchor image while the negative image shows a person not in the anchor image. As we use cosine similarity for ReID, we want the triplet loss to give higher similarity between the anchor and positive images than the anchor and negative images. In this section, you are asked to implement such a loss function.

[C13] Please implement `TripletLossDataset` that reads `train_random_triplet.csv` and outputs the anchor, positive and negative images. The same data augmentations used in Section 4.3 are also applied here. The `train_random_triplet.csv` file contains 11,500 triplets which were randomly generated.

[Q10] What is the number of possible triplets in the training dataset (230 unique IDs with 10 images for each ID)?

[C14] Regarding the loss function, we will use `TripletMarginWithDistanceLoss` to compute the triplet loss[4]. Since we use cosine similarity for ranking, you need to implement the related customized distance function instead of using the pairwise Euclidean distance function which is the default choice. You are suggested to use a margin of 0.2. Please feel free to explore further by understanding what the margin means and tuning its value, although this part will not be graded.

[C15] As in Section 4.3, please add FC layers to the ResNet backbone. Here, we use 512, 256 and 256 as output dimensionalities of the FC layers. No batchnorm, relu and dropout will be used in the last FC layer. While you used the output features before the last FC layer in Section 4.3, you should use the output features of the last FC layer in this section.

[C16] Train the model using the Adam optimizer with a batch size of 128 and a learning rate of 0.00001 for 30 epochs. Report the R1, R5 and R10 accuracy measures for each epoch. As in Section 4.3, you need to plot the training loss and validation accuracy measures. Note that the validation loss is not computed in this section.

[3] `https://towardsdatascience.com/metric-learning-loss-functions-5b67b3da99a5`
[4] `https://pytorch.org/docs/stable/generated/torch.nn.TripletMarginWithDistanceLoss.html`

[Q11] Attach three graphs: 1) training loss per iteration; 2) training loss per epoch; 3) R1, R5 and R10 accuracy measures every epoch in the validation phase. And, report the final epoch's R1, R5 and R10. Please describe the trends of the training loss, R1, R5 and R10.

[P2] Finally, you complete developing the person ReID model based on the triplet margin loss. Please follow the same steps as in Section 4.3 to submit the result of the test set. Save the result file as `triplet_loss_pred.csv`.

5 Written Report

Answer [Q1] to [Q11] in the report.

6 Some Programming Tips

As is always the case, good programming practices should be applied when coding your program. Below are some common ones but they are by no means complete:

- Using functions to structure program clearly
- Using meaningful variable and function names to improve readability
- Using consistent styles
- Including concise but informative comments
- Using a small subset of data to test the code
- Using checkpoints to save partially trained models

7 Assignment Submission

Assignment submission should only be done electronically in the Canvas course site.

There should be three files in your submission with the following naming convention required:

1. **Report** (with filename `report.pdf`): in PDF format.

2. **Prediction** (with filename `prediction.zip`): with the CSV files `id_loss_pred.csv` and `triplet_loss_pred.csv` compressed into a single ZIP file.

3. **Source code** (with filename `code.zip`): all necessary code, preferably in the form of Jupyter notebooks, compressed into a single ZIP file. The data should not be submitted to keep the file size small.

When multiple versions with the same filename are submitted, only the latest version according to the timestamp will be used for grading. Files not adhering to the naming convention above will be ignored.

8 Grading Scheme

This programming assignment will be counted towards 15% of your final course grade. The maximum scores for different tasks are shown below:

Table 3: [C]: Code, [Q]: Written report, [P]: Prediction

Grading Scheme	Code (56)	Report (34)	Prediction (10)
Dataset and Dataloader (8)			
- [C1] `RetrievalDataset` class	4		
- [C2] Data transformation	2		
- [C3] Dataloader	2		
ResNet (20)			
- [C4] Build ResBlock	2		
- [C5] Build ResNet-18	6		
- [Q1] Output shape		2	
- [Q2] Pooling layer size		2	
- [Q3] Alternative of pooling layer		2	
- [Q4] ResBlock # params		2	
- [C6] Load pretrained weights + [Q5]	2	2	
Train with ID loss (43)			
- [C7] Add head to ResNet backbone	2		
- [C8] `IDLossDataset` class	2		
- [C9] Data augmentation	2		
- [C10] Training loop with optimizer	6		
- [C11] Evaluation of ranking	6		
- [C12] t-SNE and visualization	4		
- [Q6] Output FC dimensionality		2	
- [Q7] Dropout		4	
- [Q8] Model training and validation results		8	
- [Q9] t-SNE result		2	
- [P1] Submission			5
Train with Triplet loss (29)			
- [C13] `TripletLossDataset` class	2		
- [C14] Loss function with customized distance	8		
- [C15] Add head to ResNet backbone	2		
- [C16] Training loop with optimizer	4		
- [Q10] Number of possible triplets		2	
- [Q11] Model training and validation results		6	
- [P2] Submission			5

Late submission will be accepted but with penalty. The late penalty is deduction of one point (out of a maximum of 100 points) for every minute late after 11:59pm. Being late for a fraction of a minute is considered a full minute. For example, two points will be deducted if the submission time is 00:00:34.

At most one NQA coupon may be used to entitle you to submit this assignment late for one day without grade penalty. You must download the file named `NQA.pdf` from Canvas and submit it by the original deadline to show that you want one coupon to be used.

9 Academic Integrity

Please refer to the regulations for student conduct and academic integrity on this webpage: `https://acadreg.ust.hk/generalreg`.

While you may discuss with your classmates on general ideas about the assignment, your submission should be based on your own independent effort. In case you seek help from any person or reference source, you should state it clearly in your submission. Failure to do so is considered plagiarism which will lead to appropriate disciplinary actions.