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.

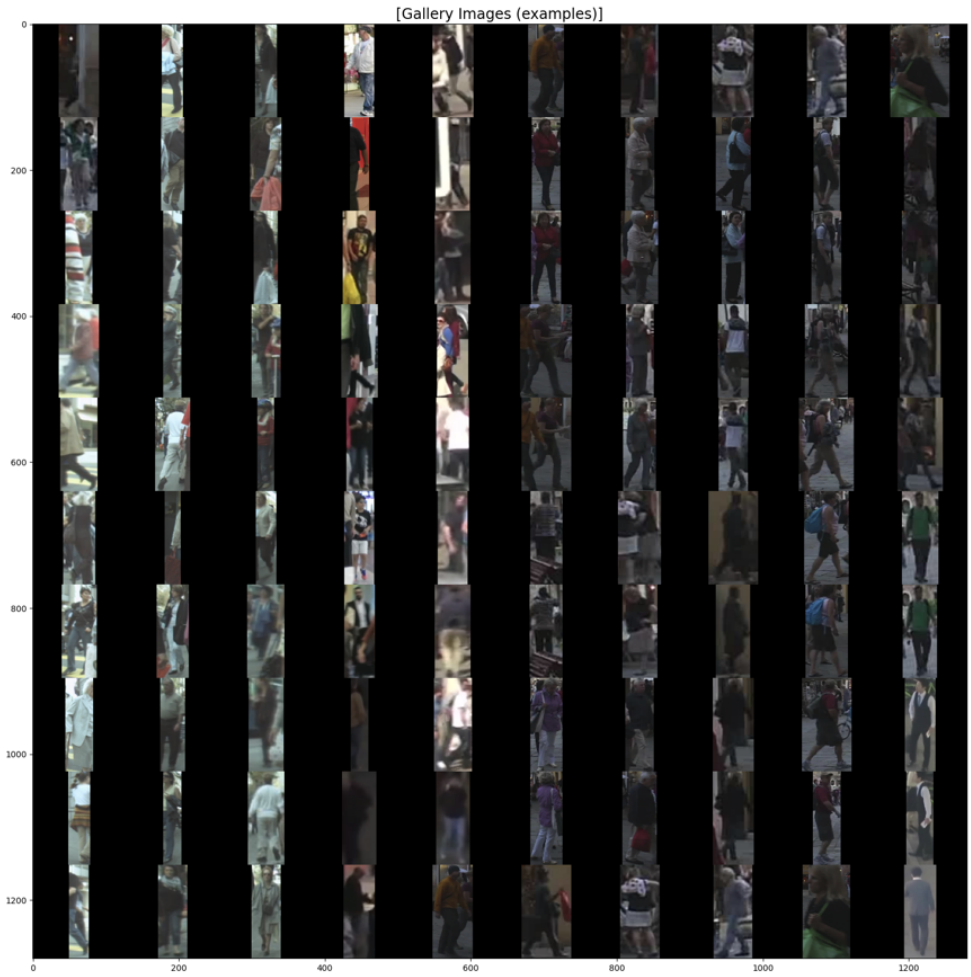

Figure 2: 100 gallery images which have been correctly resized and padded.

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