Deep Learning assignment report

Simone Barattin
University of Trento
simone.barattin@studenti.unitn.it
[223828]

Giovanni Da Rold
University of Trento
giovanni.darold@studenti.unitn.it
[231392]

1. Introduction

Person re-identification is a task which can have a meaningful impact in different scenarios, e.g. in security applications, given the increasing number of security cameras and the urgent demand of public safety. Starting from a query person-of-interest, the goal of Re-ID is to determine whether this person has appeared in another place at a distinct time captured by a different camera, or even the same camera at a different time instant. This problem has been tackled from a lot of researchers, obtaining incredible results. Early research efforts mainly focus on the handcrafted feature construction with body structures or distance metric learning, but with the advancement of deep learning, person Re-ID has achieved inspiring performance on the various benchmarks [5], [4], [10], [7], [3], [11], [9], [8].

For this assignment we were asked to develop a project based exactly on these two tasks, using the PyTorch [1] framework, a video-surveillance dataset of multiple pedestrians (each of which is captured multiple times by different cameras), and a set of annotations regarding their age and clothing attributes (e.g. upper and lower body clothing colour, presence of a backpack, handbag, hat, ...). The first task required us to perform a classification task to obtain information regarding these attributes. In the second part of the assignment we were asked to solve a person re-identification task, meaning associating images of the same person taken from different cameras or from the same camera in different occasions.

Re-ID is a challenging task, due to the presence of different viewpoints, low-resolution images (since they are taken from security cameras), illumination changes, pose changes and the domain shift occurring in real world applications, resulting in varying results and uncertainty. We will see however how using a simple ResNet, with a modified pipeline for this specific task, along with a simple siamese training allow us to obtain fairly good results in the re-identification.

The rest of the report follows this structure: dataset and data analysis, analysis of the first task's solution, analysis of the second task's solution, conclusions.

2. Dataset and Data analysis

The dataset used is a version of the Market-1501 [12], a large-scale public benchmark dataset for person re-identification, containing 1501 identities which are captured by six different cameras. The samples are the cropped images containing only the pedestrian, and, for each identity, there are the corresponding attributes describing id, age, gender, presence of accessories like backpack, bag, handbag, hat, length of upper and lower clothing, the type of clothing (dress or not), hair length and upper and lower clothing colour. In total then we have 28 (30 if we consider the multi-color).

Before starting, an analysis of the data must be performed. As stated before, the quality of a deep neural network is strongly influenced by the number of images, so data imbalance is an obstacle to obtain good performances. That is because, in case we have a class with the majority of the samples (the majority class), then the model would learn to classify the images always as that class, since the penalty from the loss function is given only few times for the minority classes. Given our approach, we treat separately age, upper colour, lower colour, and the other attributes and check the data distribution of these. The result of this study showed a strong imbalance for certain classes, as we can see from Figure 1 (blue histograms represent the number of samples without that attribute, the orange one represents the samples with that attribute), meaning applying a normal supervised learning approach would lead to bad performances.

3. Classification Task

Data processing

We have just seen how the annotations for each individual are managed, so we treat the problem as a multi-class classification problem. Therefore for the age attribute we create a one-hot encoding of it, obtaining then a 4 digits list, and all the other values contained in the annotation csv file are binarized, meaning the 1 values are turned to 0 and 2 into 1, and here for each attribute (except for the colours)

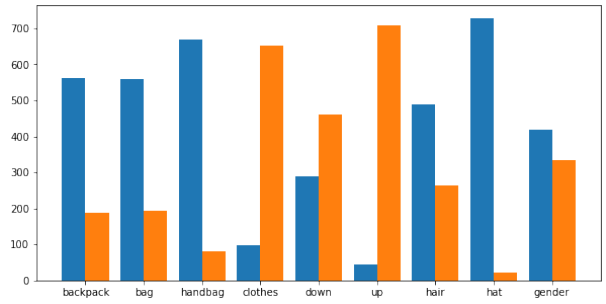

Figure 1. *Data imbalance of the attributes.*

we obtain a 2 digits list. In particular then the upper and lower clothing colours are processed as a whole (that is because the presence of a colour negates the others). The same reasoning can be applied to the other attributes, they are either present or not. Given the dataframe extracted from the csv file, we calculate also the weights of every class, which will help us in mitigating the problem of the data imbalance. First, we find the majority class and keep the number of samples of it, then we compute the weight for a class as the number of the samples of the majority class divided by the number of the current class. This way the majority class will have weight 1, the other classes instead a weight larger than 1. This gives us a tensor of weights per attribute, e.g., the attribute age will have a tensor of 4 weights (for class 1, 2, 3 and 4), while other accessory attribute will have a tensor of 2 weights (for the samples without the attribute and for the samples with the attribute). The operations described so far can be found in the **Dataset** class.

Then, to obtain fair results we want an identity to be present either in the train or validation set, so we create a list of IDs and split it based on a chosen ratio, 0.8 in our case. However, we also want to ensure the presence of the attributes in both sets, since using a simple separation of IDs can create imbalanced situations, e.g. all the samples with a backpack are in the train set and none in the validation. To avoid this, we get all the identities with each of the attributes and split it again based on a ratio. In the end we obtain two lists of IDs, which are then used to retrieve the images from the dataset. As a final data processing step, a list of transformations are applied to the sample. This method prevents the model from focusing too much on low level semantical features and increases the number of new unseen samples, meaning that the transformations also decrease the probability of overfitting.

Architecture

To perform the task of classification we started from a ResNet50. Since ImageNet is the most used dataset for classification, we decided to use the model provided by PyTorch pretrained on the above mentioned dataset. In this way we didn't have to randomly initialize the weights of the model. Clearly we couldn't use the model as it is, so we replaced the classification. We created multiple classification heads, each focusing on a specific attribute, instead of a single classification head with 30 output neurons. This choice was merely driven by the fact that having multiple heads increased the model's performances, which has to be expected since in this way we can train each head for specific features. Thus we obtain a ResNet50 with 12 classification heads, each one with a Dropout layer with p=0.2 followed by a fully connected layer with a varying number of outputs: 4 for the age, 9 for the upper body colour, 10 for the lower body colour and 2 for the rest. Notice that we do not apply any kind of activation function, e.g. Softmax, and that is because we will resort then on the use of PyTorch's loss functions, which already implement them in the calculation. For a visual example of the model's architecture please refer to the Figure 2.

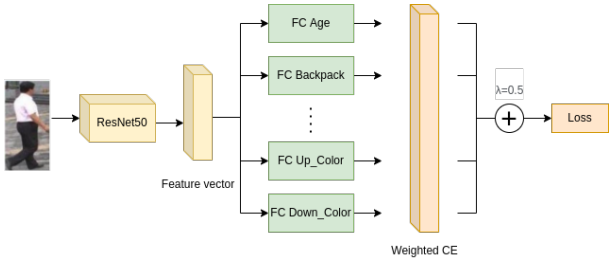

Figure 2. *Training pipeline of multi-head ResNet50.*

Train and Evaluation details

To train the model, as shown in Figure 2, the use of the Cross-Entropy, provided in the loss functions of PyTorch, is employed, as it is the standard loss for classification tasks. This loss automatically applies a LogSoftmax to the logits. For each prediction of each attribute passed to the loss function, this extracts from a dictionary the corresponding weights of the classes of that attribute and computes the weighted Cross-Entropy, giving a larger penalty to those classes with larger weights. Note that we decided to resort to the use of a class weighting approach because, given the structure of the data, other approaches would have led to the same situation. The problem with the data comes from the fact that the "accessory" attributes are part of a multi-label setting, meaning that a person can have both a hat and a backpack instead of only one; for this reason methods such as oversampling [2] or undersampling would not work. That is because while trying to oversample a minority class we may also be increasing the number of samples of another already consistent class.

Regarding the optimizer, Adam is adopted, given the good performances shown throughout different tests, where it easily outperformed SGD with and without a learning rate scheduler. The hyper-parameters used are a **learning rate**

of 1e-4, **weight decay** of 5e-4, **batch size** of 128 and the epochs for the model training are set to 100, even though the model overfits at the first epochs. That is why a naive early stopping approach is used, where if after 5 epochs the validation loss has not yet reduced, then the trend is considered to be overfitting. In the best behaviour so far the model was trained for 13 epochs before overfitting.

For the evaluation of the performances we use the mean accuracy, computed as the number of matches for each sample divided by its number. We compute also the accuracy for each attribute, in order to have a better understanding of the model's performances. This approach allowed us to obtain a solid mean accuracy of 81% during validation. Analyzing also the attributes we can see that each of them is above 60%. Some of them have very high accuracy, but unfortunately that is due to the imbalanced data; in fact even by using class weighting, the number of penalties received by the loss function is very low. Nonetheless, we can say that the proposed approach is successful for the classification task.

4. ReID Task

For the re-identification task we decided to stick to the classical approaches, and instead modify the architecture of a ResNet to solve the problem. This means that the focal point of our solution stands in the triplet loss.

Triplet loss

The goal of the Triplet loss [6] is to maximize the joint probability among all score-pairs i.e. the product of all probabilities. To compute this loss we need three components: an anchor sample, a positive sample and a negative sample. The positive one is a sample of the same class as the anchor, while the negative one is a sample from any other class. The loss then is formulated as follows

$$L_t = max(||f_a, f_p||_2^2 - ||f_a, f_n||_2^2 + \alpha, 0)$$

First the distance between anchor-positive and anchor-negative is calculated. Here we chose to use the euclidean distance, as the cosine distance obtained lower results. Then the two results are subtracted with the addition of a margin value (alpha), and the value of the loss is the maximum between the subtraction and 0. This means that if the features are close in the embedding space, the value of the subtraction is close to 0 and then the value of the loss would be close to the margin. Since the margin is used to push away the negative samples, its value is very important. We run some tests with different margin values (1, 5, 10, 15) and, based on the results, we chose to use a margin of 10, which ensures a good separation between features in the embedding space. Considering that we want to minimize this loss, the only way is to push away the negative samples over the margin and pull together anchor and positive. In this way the result of the subtraction will be equal or larger than the margin value, leading to a result close to 0. This whole allows the model to learn how two equal samples are similar and the features that discriminate two different samples.

Data processing

As this loss requires multiple samples we need further data processing steps with respect to the ones described before. For this process we start from the **CustomTrainDataset** class reported in Section 3, meaning that for each image we save in a list a tuple *(path, id)*, with the path of that image and the id of the individual, without saving the attributes since we don't need them. The instances of the created datasets, both training and validation, are passed to the **TripletDataset**, where we extract the list of tuples and save as labels the IDs; the paths are instead treated as the data we want to retrieve. From the IDs extracted, finally, we create a mapping from ID to index (*label_to_indices*). This dictionary allows us to obtain a list with the indexes of all the samples with the same ID.

There are two main ways then to create the dataloader and the batches: we can have an online mining of triplets, where we want to extract directly from the batch the three images required, or we can have an hard triplet mining, where the triplets are generated before the use of the dataloader. In order to perform an online mining we would need to ensure the presence of at least one positive sample in the batch, meaning we need to create a batch sampler. After testing this we noticed that having a batch sampler causes problems in the retrieval of the samples (only a few of the total were sampled). Also to exploit the best of the online mining we would need more than one positive sample, increasing the complexity even more. Given that this method gave more problems than results, the hard mining method was used.

To create then the triplets, we work directly on the get_item method of the **TripletDataset**, to avoid having totally fixed triplets. Given the index of the anchor, we get the ID of the individual and use the *label_to_indices* dictionary to extract all the positive samples. Using the NumPy library, we randomly sample a positive sample and the same is done for all the negative samples. The process to obtain the train and validation set is the same as before. In the same way, we take the IDs of the validation set and split them according to a certain ratio (e.g. 0.1 in our case), to obtain a set of query images and a gallery set, which are then used to evaluate the re-identification performances.

Architecture

The architecture used to solve the re-identification problem can be seen in Figure 3. The idea was taken from [4], the architecture is based on a multi-branch structure. The

focus of this multi-branch approach is to keep the branches independent between each other, which makes the features extracted from different branches diversified. Starting from a ResNet50, the first 3 layers are left untouched, while the 4 one is duplicated. We end up then with two branches: the first one is the original pipeline with 3 bottlenecks and a global average pooling layer, the second one instead has 4 bottlenecks and uses a max pooling layer. This structure is capable of extracting global information with the average pooling, along with the most distinguishable identity information with the max pooling.

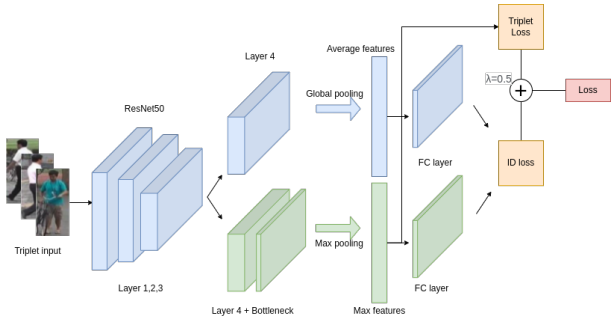

Figure 3. *Training pipeline of multi-branch ResNet50.*

The features extracted are returned for the triplet loss and passed on to two fully connected layers (one per branch). We decided to plug this classification head to perform an ID classification task.

Train and Evaluation details

To train the model, as shown in Figure 3, we use two losses weighted by a lambda factor as

$$L = \lambda L_{Tmax} + \lambda L_{Tglob} + \lambda L_{CEmax} + \lambda L_{CEglob}$$

where with T we refer to the triplet loss, CE refers to the cross entropy, *glob* refers to the average pool branch, and *max* refers to the max pool branch. We set λ to 0.5. For each sample given by the dataloader, along with the positive and negative samples, the features are extracted from the output of both global and max pooling layers, and are then used to calculate the triplet loss; next, after passing the features through a batch normalization for regularization, the features are passed to the classification heads which give a prediction of the person's ID. The prediction is used to calculate the ID loss. We decided to use an additional loss because it is known that multi-task learning can improve by a lot the performances of a neural network. To prove that our approach improves the model's performances, Table 1 reports an ablation study of the model with and without the additional loss.

The optimizer used is again Adam, since it showed the best behaviour. The hyper-parameters used are a **learning rate** of 1e-4, **weight decay** of 5e-4, **batch size** of 32 (this is due to the computational resource limits), and the maximum number of epochs is set to 25. Again here we use the same early stopping as in the classification task, however after the 13th epoch the model converges and the validation loss has low reductions.

	Accuracy		
	mAP	Train loss	Validation loss
No ID loss	0.64	**0.01670**	**0.01563**
With ID loss	**0.81**	0.01931	0.17935

Table 1. Ablation study of the model in the reID task.

For the evaluation we used only the mAP (Mean Average Precision), which is a popular evaluation metric used for object detection (i.e. localisation and classification). In our case we used the code provided to us, which encodes the ability of the model to retrieve the right samples given a query image. We compute the mAP after every validation step in every epoch to understand how well the model is learning. We performed also here an analysis on the data to understand how many images where present for each identity. This study reported that the average number of images per person is 17, but we cannot use the average value as a threshold as we would cut off the majority of the samples. The highest number of images for an individual is 72, so we decided to use 80 as the threshold of samples, ordered by the distance value, to process during the computation of the mAP, to handle possible identities with a number larger than 72.

After the model is trained we obtain the results on the validation set, and we were able to reach a very good mAP of 82% at peak performance, with a validation loss curve which converges and does not show signs of overfitting. To better understand if the model really learns to distinguish the different identities, we plotted the t-SNE of 10 identities with the largest number of samples. The result shows that indeed the model clusters together the samples of the same identity with some entanglement between classes (which has to be expected to a certain extent).

5. Conclusions

We have seen that using a simple model with the right approaches we were able to obtain good results in the Market-1501 dataset for both the classification and the re-identification tasks. Of course, this does not mean that our solution could be deployed to a real world application, or even obtain good results in other datasets. Indeed, in presence of a domain shift, a drop in performances has to be expected.

References

[1] Pytorch documentation. https://pytorch.org/docs/stable/index.html. 1

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, Jun 2002. 2

[3] Guangyi Chen, Tianpei Gu, Jiwen Lu, Jin-An Bao, and Jie Zhou. Person re-identification via attention pyramid. *IEEE Transactions on Image Processing*, 30:7663–7676, 2021. 1

[4] Hao Chen, Benoit Lagadec, and Francois Bremond. Enhancing diversity in teacher-student networks via asymmetric branches for unsupervised person re-identification, 2020. 1, 3

[5] De Cheng, Yihong Gong, Sanping Zhou, Jinjun Wang, and Nanning Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1335–1344, 2016. 1

[6] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification, 2017. 3

[7] Yang Li, Huahu Xu, Minjie Bian, and Junsheng Xiao. Attention based cnn-convlstm for pedestrian attribute recognition. *Sensors*, 20(3), 2020. 1

[8] Hao Luo, Pichao Wang, Yi Xu, Feng Ding, Yanxin Zhou, Fan Wang, Hao Li, and Rong Jin. Self-supervised pre-training for transformer-based person re-identification, 2021. 1

[9] Guangcong Wang, Jianhuang Lai, Peigen Huang, and Xiaohua Xie. Spatial-temporal person re-identification, 2018. 1

[10] Mikolaj Wieczorek, Barbara Rychalska, and Jacek Dabrowski. On the unreasonable effectiveness of centroids in image retrieval, 2021. 1

[11] Xianghao Zang, Ge Li, Wei Gao, and Xiujun Shu. Learning to disentangle scenes for person re-identification, 2021. 1

[12] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1116–1124, 2015. 1