# Person classification and re-identification

Deep Learning project

Francesco Ferrini
francesco.ferrini@studenti.unitn.it
224010

Wamiq Raza
wamiq.raza@studenti.unitn.it
224824

Sebastiano Chiari
sebastiano.chiari@studenti.unitn.it
220527

## ABSTRACT

*This report describes the method used and the results obtained to complete the final project for the Deep Learning course.*
*The aim of this project is to use neural networks to solve two common computer vision tasks using the PyTorch framework.*
*The dataset [4] used for this purpose is a video-surveillance one, containing images of multiple persons: each pedestrian is captured multiple times by different cameras along with a set of annotations that specify attributes of each person such as age, gender and clothing.*
*The first section describes the method, training and evaluation strategies and the results for the first task, building a multi-class classification model. In the second section, it is described our methodology and the results obtained with regards to the second task of the project, a re-identification model.*

## 1 INTRODUCTION

Pedestrian Attribute Recognition (AR) is the task of predicting multiple attributes of pedestrian images in a video surveillance domain, instead person re-identification (in the same domain) has as objective to identify the same person in different images taken from different cameras. In both tasks, Deep Learning has shown a significant advance over the past year's because of the improvement in different Neural Networks in the field of Computer Vision. For instance, CNN's that are able to learn powerful features from images in this section we discuss the classification task and their proposed solution.

## 2 CLASSIFICATION TASK

The first task of this project is to design, train and evaluate a **multi-class classifier** to predict a set of attributes for a given pedestrian image as input. The attributes are the following:

- gender
- hair length
- sleeve length
- length of lower-body clothing
- type of lower-body clothing
- wearing hat
- carrying backpack
- carrying bag
- carrying handbag
- age
- color of upper-body clothing
- color of lower-body clothing

The annotations for the training images are provided within the `annotations_train.csv` file, while the outcome of the proposed classifier can be found within the `classification_test.csv` file.

### 2.1 Method

Before starting with the formulation of the model, we conducted a research on the state of the art, both for general purpose classifiers and for already existing works based on our dataset, Market-1501 [3]. We found that, over multiple architectures, the works delivering consistent and state of the art results were using ResNet based architecture, so we decided to stick with that.

We based our approach over the following paper [2], which proposes a state of the art method for person re-identification. We took inspiration from their first section, where they used a customized multi-class classifier with a pretrained ResNet50 as backbone.

In particular, we use M (one for each feature we want to classify) FC layers to perform the final step: they replace the FC layer of the pretrained ResNet50. Each FC layer is a `Sequential` one, composed by the following blocks:

- `dropout`, with `0.4` dropout percentage
- `linear`, `N` features to `1000` features
- `ReLU`
- `dropout`, with `0.4` dropout percentage
- `linear`, `1000` features to `n` features (depending on the feature we are looking for)

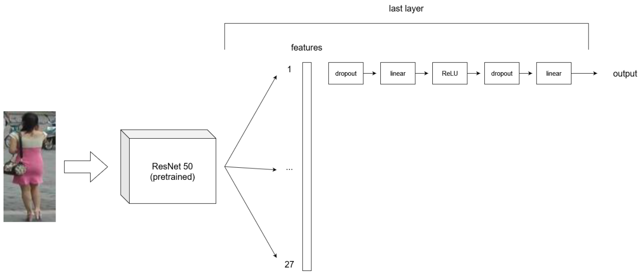


**Figure 1: Custom ResNet50 structure for classification task**

### 2.2 Training and evaluation

*Splitting the dataset.* Market1501 is divided between train and test images. More, we are given an `annotations_train.csv`, where labels are associated for each ID. Since there is no fixed amount of images for each ID, we split the training set between train and validation slices by IDs, based on an arbitrarily split percentage (tests can be seen in Section 2.3). This strategy ensures that there are no images with the same ID in both training and validation. However, it makes the size of these two slices variable.

*Data augmentation.* Given the dataset nature, with lots of very similar images with almost identical background and in order to improve the generalization performances of our model and to avoid overfitting, we introduced some data augmentation. We applied some transformations (such as RandomHorizontalFlip, RandomRotation and RandomErasing), perfomed normalization and resized the image in order to fit the network specifications.

*Optimizer.* As optimizers, we decided to try two of the most famous and performing optimizers for this kind of tasks, SGD with momentum and Adam. We empirically pick the one performing the best on our model (Section 2.3).

*Adaptive Learning Rate.* We introduced also an adaptive learning rate, in order to reduced the steps when the network reaches the final iterations. We used a Pytorch ReduceLROnPlateau function, to lower the learning rate when a metric (the validation loss) has stopped improving.

*Cost function.* We used CrossEntropy as loss function in our model. A BinaryCrossEntropy loss would have been a perfectly fitting choice since most of our classification outputs are binary: on the other hand, the age feature ranges between 0 to 4, so it was not possible without changing the whole structure, with for example a one hot encoding strategy.

*Early stopping.* We set 100 epochs as starting point but, since our model is prone to overfit we introduced an early stopping strategy: after each training epoch, the validation loss is compared with the previous one and if the model does not improve the performances for n consecutive epochs, where n is the patience, we stop the training and save the best model.

## 2.3 Results

First, we have to find out which is the best combination for optimizer and split percentages. We run some tests with standard initialization with regard to hyperparameters in order to compare the different configurations. Since our datasets was not huge in terms of available train images, we decided to stick with a batchsize equal to 64.

| optimizer | split % | train_loss | train_acc | val_loss | val_acc |
|---|---|---|---|---|---|
| Adam | 70 | 0.273 | 88.986 | 0.285 | 88.753 |
| Adam | 80 | 0.254 | 89.881 | 0.291 | 88.295 |
| SGD | 70 | **0.157** | **93.824** | **0.236** | **90.916** |
| SGD | 80 | 0.166 | 93.447 | 0.245 | 90.559 |

**Table 1: Optimizers and split percentages comparisons**

We can easily see that SGD with momentum performs better than Adam, which is less accurate and produces higher losses. It is interesting to see that performances are not that different between different split percentages.

Once we chose the optimizer and how to split the dataset, we began to investigate different configurations of hyperparameters (specifically, momentum and weight decay).

| training epochs | momentum | train_loss | train_acc | val_loss | val_acc |
|---|---|---|---|---|---|
| 14 | 0.5 | 0.166 | 93.482 | 0.241 | 90.783 |
| 15 | 0.7 | 0.196 | 92.201 | 0.243 | 90.581 |
| 40 | 0.9 | **0.157** | **93.824** | **0.236** | **90.916** |

**Table 2: Comparison between different momentum values**

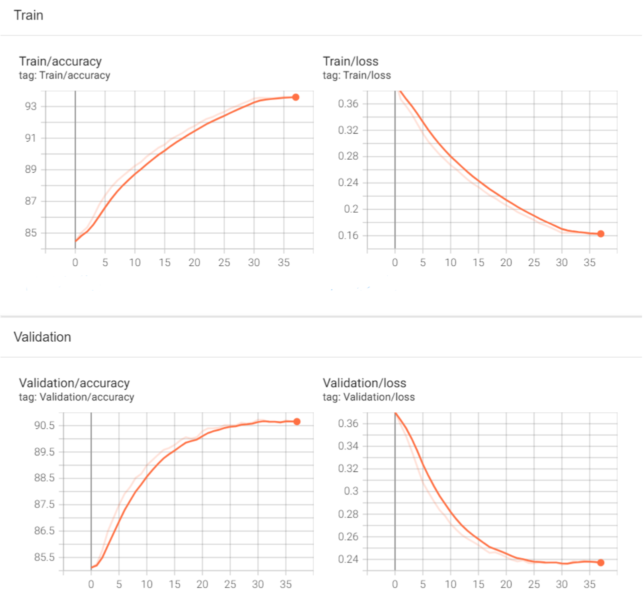
With a lower momentum, the network converges faster than with an higher one, but on the other side it starts to overfit very quickly and early stopping kicks in after about 7 epochs. The performance on the validation set are close but it is preferable to use an higher momentum.

Since we use an adaptive learning rate, we fixed the learning rate starting point at 0.01, leaving to the model the decisions on this parameter based on the progress of the training. weight decay was set based on empirical tests too: in the end, we found 0.01 as the best value for our model.

To conclude, our best model, achieving a validation accuracy of **91%** has the following configuration:

| split % | batch size | learning rate | weight decay | momentum |
|---|---|---|---|---|
| 70 | 64 | 0.01 | 0.01 | 0.9 |

**Table 3: Final configuration**



| attribute | validation accuracy |
|---|---|
| age | 80.719 |
| backpack | 78.200 |
| bag | 71.065 |
| handbag | 88.877 |
| clothes | 91.238 |
| down | 86.857 |
| up | 94.963 |
| hair | 84.339 |
| hat | 96.983 |
| gender | 84.129 |

**Table 4: Some attribute accuracy with best performing model**

# 3 PERSON RE-IDENTIFICATION

The second task of this project consists in building a **person re-identification model**: for each image in the query directory, the model has to produce an ordered list of images from the test directory that it believes correspond to the same person identity depicted in the query image. Each query image has at least a corresponding image in the test directory, but the number of corresponding images varies from query to query. The test directory also contains a set of junk images that should never be returned as the result of a query.

The outcome of this model can be found within the `reid_test.txt` file: each row corresponds to one *query* image, where it is appended to the query image filename the list of image filenames the model depicted as the same person.

## 3.1 Method

For the re-identification task we reused a pretrained ResNet-50 but this time without the final classification layer since now we don't need to perform classification. In this way we obtain for each image a feature vector which is then used in the cost function that in our case is Triplet Loss [1]. During the training we used an early stopping with a patience of 7 so that whether our model accuracy's doesn't improve for 7 epochs, the training stops and the best model is returned. In order to make predictions as last step of the re-identification, we re-loaded the trained model and extracted the embedding of the images in the query and in the test folder. Then each embedding of the query images is compared with the embedding of the test images using as measure the cosine similarity. We decided that 2 embeddings refer to the same person whether the cosine is higher than a cosine similarity threshold of 0.99.

## 3.2 Training and evaluation

*Splitting the dataset.* The splitting of the dataset is the same as in the previous task, hence 80% of the ids are used for training and the rest 20% for validation.

*Data Augmentation.* As data preparation we only performed Data Augmentation as in the first task.

*Optimizer.* We empirically selected to use **RMSprop**, over Adam and SGD, with a learning rate of 0.001 and an alpha value of 0.9. After running some tests, we discovered that this configuration was the one performing the best.

*Cost function.* In order to train the network we used as cost function the Triplet Loss where the formula is the following:

$$L(a, p, n) = \sum_{a,p,n} m + D_{a,p} - D_{a,n} \qquad (1)$$

where $m$ is the margin, $D_{a,p}$ is the distance between the embedding of the anchor image and the embedding of the positive image and $D_{a,n}$ is the distance between the embedding of the anchor image and the embedding of the negative image. The objective of this function is to keep the distance between the anchor image and positive one smaller than the distance between the anchor image and the negative one. There are several distance measures that can be used in this framework and we decided to use the Euclidean Distance.

*Parameters.* As for the classification task, we used a batch size of 64 and 50 epochs for the training.

## 3.3 Results

To evaluate the model we used the mAP (mean Average Precision) which is a metric that computes the average precision value for recall value over 0 to 1. In order to calculate it we passed the validation samples to the model extracting for each image a vector of 2048 features. Then each image features' are compared with the others using the cosine similarity measure. We considered that two images contains the same subject (hence the system has re-identified a person) when the cosine similarity value is higher that 0.99.
Our configuration achieves a 36% mAP score.

# REFERENCES

[1] Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In Defense of the Triplet Loss for Person Re-Identification. *CoRR* abs/1703.07737 (2017). arXiv:1703.07737 http://arxiv.org/abs/1703.07737

[2] Yutian Lin, Liang Zheng, Zhedong Zheng, Yu Wu, Zhilan Hu, Chenggang Yan, and Yi Yang. 2019. Improving person re-identification by attribute and identity learning. *Pattern Recognition* 95 (Nov 2019), 151–161. https://doi.org/10.1016/j.patcog.2019.06.006

[3] Person Re-Identification on Market-1501. 2022. https://paperswithcode.com/dataset/market-1501

[4] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. 2022. Market-1501 Dataset. http://zheng-lab.cecs.anu.edu.au/Project/project_reid.html