Classification and Person Re-Identification

Bernardi Marco
University of Trento
marco.bernardi@studenti.unitn.it

Hueller Jhonny
University of Trento
jhonny.hueller@studenti.unitn.it

Saccon Enrico
University of Trento
enrico.saccon@studenti.unitn.it

Abstract

In this work, we explore the usage of PyTorch framework in order to build a neural network able to solve two computer tasks: classification and person re-identification. The first part of this project consists in building a multi-class classifier to predict attributes for each image. The second part explores the person re-identification problem where, given a collection of various images and a query one of a person, all the images that are associated with the query one should be retrieved from the collection.

1. Introduction

The history of neural networks is a troubled one, with periods of great discovery followed by ones of stagnation. The last few decades have been a period of great innovations in the field of machine learning in general, especially driven by the increment of raw computational power, e.g., given by the always more spread usage of graphics card [5] and ad-hoc devices, but also by the availability of wider datasets and also better labelled.
The goal of this work is to implement a neural network that aims at solving two well-known problems: classification and person re-identification. In particular, the work focuses on the usage of the state-of-the-art dataset known as *Market-1501* [11]. This contains images as tight crops of persons at a market and is divided in 3 directories: the first one contains pictures used for training which are also labelled with a list of attributes, the second one contains unlabelled images for testing and the last one contains images for querying. It is important to notice that in the dataset, a pedestrian can appear in multiple images.
This paper is divided into two parts: the first concentrates on the classification aspect of the work, while the second considers the details of the person re-identification task.
The aim of the classification task is to train a model in such a way that, given the image of a pedestrian, it predicts the following attributes: 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, multicolor upper-body clothing and multicolor lower-body clothing.
Instead, the person re-identification task aims at having a network that, given the image of a pedestrian, should retrieve all the images depicting that same person.

2. Classification Task

To carry out the classification task the images in the training folder were separated in two datasets, one used for training and the second one used for validation: in order to correctly train the model, the function `subdivide_train_val_samples` that splits the folder ensures that all the images of a given person are in the same dataset. For this same reason, it is not given that the ration between the lengths of the two datasets is actually respected.
Then a neural network based on the pretrained ResNet34 network [2] was devised. The usage of a pretrained backbone allows to not initialize the weights in a random way but in a more efficient way [4] and will act as a features extractor for the images that will be given in input. The features obtained will thus become the input of the classifiers, one for each attribute. The classifier is made up of one input layer of 512 nodes (corresponding to the output of the feature extractor), one hidden layer of 256 nodes and an output layer with a number of nodes that depends on the values that each attribute can take (2, 4, 9 or 10). The loss is calculated by adding all the losses obtained through the cross entropy function [1] between the predicted value of each classifier and the ground truth. Stochastic Gradient Descent [7] was used as the optimizer, for which the following parameters were set after fine-tuning them: learning rate was set to 0.001, the weight decay to 0.00001 and the momentum to 0.8. To reduce the risk of overfitting, the dropout [9] was set at 0.5 and batch normalization [3] was applied at the output of the hidden layer for each classifier. Another important aspect in deep learning neural networks is the ability of the network to learn from a small dataset without overfitting. One way

to avoid this is by using data augmentation [8], which is a technique that consists in manipulating the training images with different variations in colours, size, kernel transformations, etc. Such technique allows for a better training of the network when using smaller datasets. In this work, different transformations techniques had been tried: the training images had been resized to fit the 256x256 pixel area required by ResNet34, then a random change in the brightness, contrast, and saturation had been applied, followed by a random horizontal flip, rotation, shift and a random erasure of a portion of the image, ending with normalization. At the end though, it was decided not to apply this type of data augmentation as the results obtained were worse than using the images as they were. This is mainly due to the fact that these types of transformations can improve the results only if there is overfitting in the network training process, but in this case the Market-1501 dataset is big enough to avoid such risk.
Finally early-stopping was added to avoid overfitting the training set: such technique allows to stop the training of the model when it perceives that the accuracy over the training set is increasing, while the one over the validation set is decreasing. Such technique is also used in the second part of the work.

2.1. Results

The neural network was trained for up to 200 epochs using early stopping. The best case was obtained by setting `lr=0.001`: the training led the average accuracy on the validation set (the average of the accuracies obtained by each classifier) from 40.27% up to an average value of 81.88%. On the training set the accuracy obtained after the training was 96.73%, while the total loss on the training set is 0.0584 and on the validation set is 0.0968. In Table 1, it is possible to see all the accuracies obtained for each single attribute, both for the training set and for the validation set. The accuracies obtained with a few tested learning rates are shown in Figure 1.

2.2. Descriptions of Main Classes and Modules

In the following section, a brief description of the main components for the classification task is carried out.
MarketDataset: this is a dataset class that inherits from `torch.utils.data.Dataset` and is initialized giving a list of ids, a list of images, the annotations and the possible transformations to be used. When requesting an item, it will return a tuple containing the image and a dictionary of the labels for that image. Two auxiliary functions, `get_prediction` and `format_prediction`, are used when a prediction has been made to retrieve the labels of the image as a dictionary and to correctly format the prediction, either to be written on a file such as `test.csv`, which follows the same format as `annotations_train.csv`, or to be printed to `stdout`.
MarketClassifier: this is the neural network for the classification task made up of the feature extractor and 12 classifiers, one for each attribute. While the general structure of the network has been described in the introduction of Section 2, it is worth better describing the components. The feature extractor is a class that inherits from `torch.nn.Module` and it simply consists in ResNet34 with the last layer changed to a `Linear` layer with 512 output features. The classifier layer inherits from `torch.nn.ModuleDict` so that it can be hold all the submodules in a dictionary for an easier access. Each submodule is given by the `LabelClassifier` module which has been described before. The final output of this network is a dictionary containing the predictions for each attribute using linear layers with as many output features as the possible outcomes of the attribute.
EarlyStopping: this class allows to verify whether the network is decreasing the accuracy over the validation set for a number of times, namely `patience`, set to 5 by default. After the network has dropped the accuracy on the validation set for `patience` epochs, the training is stopped.

3. Person Re-Identification

The validation set and the training set were obtained in the same way as in the classification task.
To carry out this task a neural network based on the one already used and trained in the classification task as been devised by adding a new block called aggregator, which receives as input the outputs of all the classifiers as well as the output of the features extractor. The diagram of the network can be seen in Figure 2. The outputs of the classifiers are 41 since an attribute can assume 10 values (lower-body color), another can assume 9 values (upper-body color), another 4 values (age) and 9 attributes that can assume only 2 values (gender, hair length, sleeve length, length of lower-body clothing, type of lower-body clothing, wearing hat, carrying backpack and carrying bag). The output of the feature extractor instead is 512 nodes so the aggregator input layer will be composed of a total of 543 nodes ($512 + 41$). Following the input layer, a batch normalization is applied and the aggregator output consists of 128 nodes.
The optimizer used is Stochastic Gradient Descent [7] as before, while the loss function is a triplet margin loss [10]. The Triplet margin loss allows to compare an input (called anchor) to both a matching input (called positive) and a non-matching input (called negative). The goal is the to minimize the distance from the anchor to the positive and at the same time to maximize the distance from the anchor to the negative input. When the net is trained, 3 images are extracted: an image (anchor) corresponding to an image of the person ID it needs to recognize, an image belonging to

	age	backpack	bag	handbag	clothes	down	up	hair	hat	gender	upcolor	downcolor
training set	87.99	86.54	79.86	90.00	94.93	94.36	94.95	91.63	97.45	94.36	77.95	72.04
validation set	77.95	85.51	72.94	86.39	89.95	85.95	88.47	84.07	96.47	84.75	67.70	52.18

Table 1. Accuracies[%] obtained from the classification task after having trained the network with `lr=0.001`.

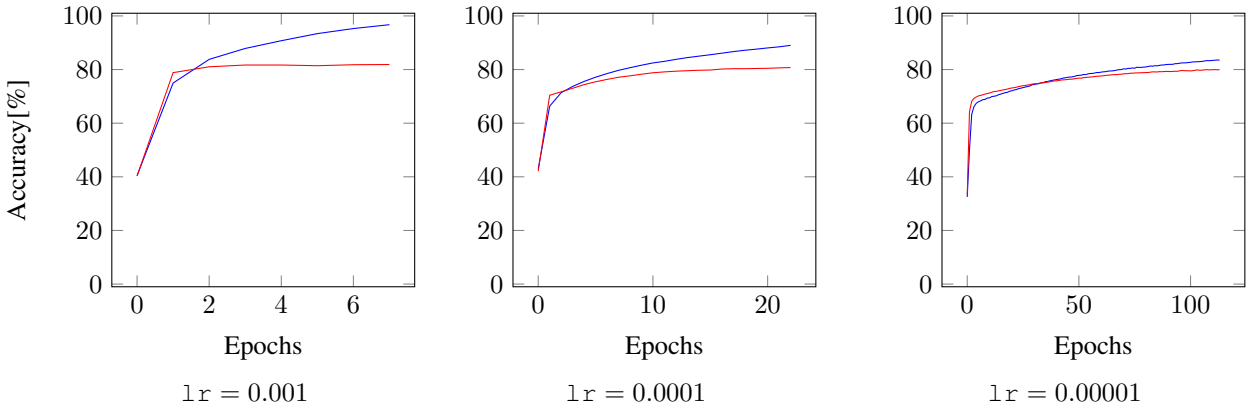

Figure 1. Plotted accuracies[%] (blue for training set and red for validation set) over number of epochs for different learning rates (`lr`).

the same person (positive) and finally a randomly chosen image belonging to a person with a different ID (negative). The output of the network is calculated from each of these 3 images: the similarity between the anchor and the positive and also between the anchor and the negative is calculated using the triplet margin loss cost function. Finally, the distance (Euclidean distance) between the network output is calculated using the output of the anchor and the output of the negative. What the network will do is try to maximize this distance as much as possible.

Once satisfied with the performance achieved by the network, a dictionary is created where each key represents the path of the query image while the value contains the paths of the images which, according to the network, identify the person of the query image ordered in ascending order by distance.

In order to set the distance interval in which an image can be consider as to identify the person in the query image, the images in the training folder were used. In particular, a dictionary has been created where each key represents an image with a different ID and the values represent all the images with the same ID as the respective key. Then by trying different threshold values, obtained by simply increasing it of 0.1 each time, it was possible to find the threshold value within which the best results in terms of mean average precision (mAP). The best mAP results obtained was 12.73% on the validation set and 15.53% on the training set with a threshold value of 6.

3.1. Results

The network was trained up to 200 epochs, but early-stopping interrupted the training at the 64th iteration showing loss values of 0.00022 (mean distance of 12.81) for training and of 0.00022 (mean distance of 12.51) for the validation set, when the trained model was used again on the sets.

3.2. Description of Classes and Modules

In the following section, a brief description of the main components for the re-identification task is carried out.

AggregatorDataset: this is a dataset class that inherits from `toch.utils.data.Dataset`. It first creates a dictionary where the keys are the IDs of the pedestrians and the values are lists of photos corresponding to the IDs. Then it compresses them into a list of lists so that they can easily be retrieved through the `__getitem__` function. This returns an image (anchor), another randomly chosen image (positive) corresponding to the same ID of the previous image and a random image belonging to a person with a different ID (negative).

Aggregator: this is the network for the whole project. It is made up of feature extractors and classifiers already trained in the classification task plus an aggregator which receives the outputs of the feature extractor and of each classifiers as input. The output consists of 128 features that will be used by the distance function to calculate the similarity between two images. The aggregator module is composed of a linear module to combine the inputs followed by a Rectified Linear Unit (ReLU) [6] and a dropout layer for better training performance.

measure_distances: this function creates a dictionary where each key is the path of the query image while the value is a further dictionary setting the paths of the images in the test folder as keys and the distances with respect to

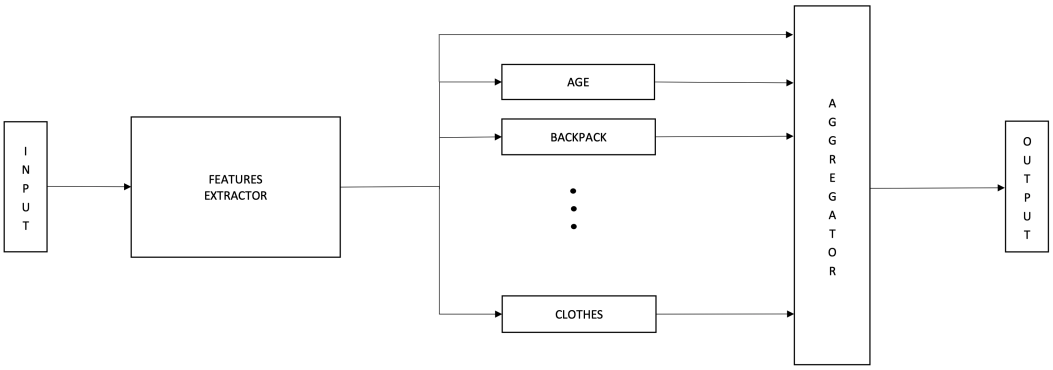

Figure 2. An overview of the network structure.

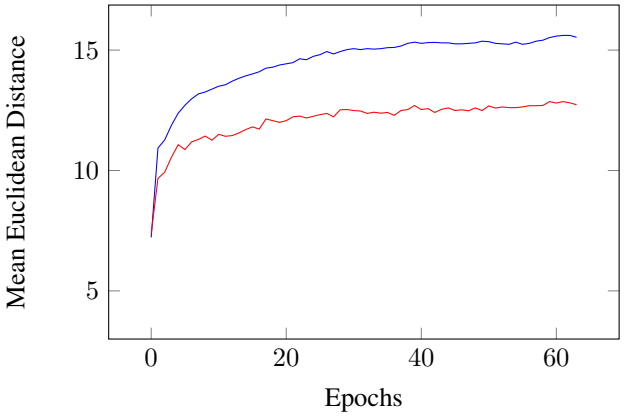

Figure 3. How the mean euclidean distance varies while training the re-identification model (`lr` $= 0.001$), in blue the distance over the training set and in red the distance over the validation set.

the query image as values. This dictionary is sorted from the value of the smallest distance to the largest. If the value of the parameter `TRAIN_MODE` is `True`, then the threshold value is increased by 0.1 starting from 1 to 15: all the images whose distance is below the threshold are kept. The threshold value that obtains the highest value in terms of mAP is chosen. At this point `TRAIN_MODE` is set to `False` and the threshold value is set to the best value found previously. Finally, the `reid_test.txt` file is produced containing all the images associated with the query images.

Evaluator: this class contains the method `evaluate_map` which computes the mAP score for the predicted value w.r.t. the ground truth elements.

4. Conclusion

While this work does not meet state-of-the-art performance, it provides a nice oversight of the most popular techniques used in deep learning to solve two of the most widespread tasks: classification and person re-identification.

References

[1] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, Feb 2005. 1

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 1

[4] Brady Kieffer, Morteza Babaie, Shivam Kalra, and H. R. Tizhoosh. Convolutional neural networks for histopathology image classification: Training vs. using pre-trained networks. In *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6, 2017. 1

[5] Sparsh Mittal and Shraiysh Vaishay. A survey of techniques for optimizing deep learning on gpus. *Journal of Systems Architecture*, 99:101635, 2019. 1

[6] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. 3

[7] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. 1, 2

[8] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, Jul 2019. 2

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 1

[10] Daniel Ponsa Vassileios Balntas, Edgar Riba and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 119.1–119.11. BMVA Press, September 2016. 2

[11] 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