Deep Learning Classification and Re-Identification of People

Giovanni Lorenzini
223715
giovanni.lorenzini@studenti.unitn.it

Simone Luchetta
223716
simone.luchetta@studenti.unitn.it

Diego Planchenstainer
223728
d.planchenstainer@studenti.unitn.it

University of Trento

Abstract

In this report we propose a neural network architecture for carrying out the following assignments: at first instance the network should classify a picture of a person producing a full list of attributes and then it should also provide a set of images that represent the same person of a query image. This work deals with the development of a neural network architecture based on a ResNet-50 backbone model to perform attribute classification and person Re-Identification on the Market-1501 dataset. The code is written in Python, leveraging on the PyTorch framework. For speeding up the training processes, tensors are transferred to an RTX 3080 used as CUDA device.

1. Introduction

In the field of Artificial Intelligence, there are many types of neural networks, each one with its specific purpose. For example, Convolutional Neural Networks are deployed when analysing visual inputs (images and videos), while Recurrent Neural Networks come useful in processing natural language. Most of the models are being trained via gradient-based learning methods and back-propagation. Basically, what happens is that each of the network's parameters is updated by an amount which is equal to the derivative on the error function with respect to the current network's weights.

In this work, a customized network with a ResNet-50 backbone is constructed. The aim of this project is to adopt the network for completing both a *pedestrian attribute recognition* and a *person Re-Identification* task. For both jobs, a tightly cropped image representing a pedestrian is being fed into the model. The network is firstly employed in predicting up to 41 different attributes concerning the individual depicted in the picture such as *gender*, *age*, *wearing hat* and so on. Then, the image is also fed to the algorithm that retrieves back all the images that picture the same subject.

2. Problem statement

As aforementioned, the problem is to train a neural network architecture to recognize attributes from an image, and to collect pictures portraying subjects that are close to an input query.

Pictures from the Market-1501 dataset contains a total of 1501 different identities, and are put into three different folders, namely *train*, *test* and *queries*. Note that *train* and *test* contain respectively 751 and 750 distinct person identities. Annotations for individuals inside *train* folder are given in *annotations_train.csv*.

Note that for training our model, data inside the *train* folder is divided into *training* and *validation*, so that subjects in *training* are not inserted in the *validation* split.

The two distinct tasks are addressed in upcoming sections.

2.1. Attributes classification

This task consists in predicting the list of attributes represented in the Table 1. Note that for each of these attributes, a set of values that the attributes can take is shown; these values represent different meaning for each attribute.

Remark that during the training process, data within *train* folder is divided as aforementioned, forming two splits, namely *training* and *validation*. This way it is possible to obtain accuracies for each dataset subdivision as the training procedure carries on. Then, once the model has trained for long enough epochs, an annotation file is produced containing descriptions for each image within the *test* folder, following the same format as *annotations_train.csv*.

Name	Values	Value Meaning
Gender	$\{0, 1\}$	Male, Female
Hair length	$\{0, 1\}$	Short hair, Long hair
Sleeve length	$\{0, 1\}$	Long sleeve, Short sleeve
Length lower body clothing	$\{0, 1\}$	Long clothing, Short clothing
Type lower body clothing	$\{0, 1\}$	Dress, Pants
Wearing hat	$\{0, 1\}$	No, Yes
Carrying backpack	$\{0, 1\}$	No, Yes
Carrying bag	$\{0, 1\}$	No, Yes
Carrying handbag	$\{0, 1\}$	No, Yes
Carrying handbag	$\{0, 1\}$	No, Yes
Color lower body clothing	$\{0, \ldots, 9\}$	multi-color, black, white, pink, purple, yellow, gray, blue, green, brown
Color upper body clothing	$\{0, \ldots, 8\}$	multi-color, black, white, red, purple, yellow, gray, blue, green
Age	$\{0, \ldots, 3\}$	Young, Teenager, Adult, Old

Table 1. Attribute values

A possible method to verify the presence of junk images could be to take a look at the sum of the squared values of the attributes Tensors. Low values are associated with the possible presence of distractors or junk images, because the network estimates with low probabilities all the outcomes.

2.2. Re-Identification

In the Re-ID task the training process is carried out on the *training* split, while the *validation* split contains both test and query pictures for which we have ground truths. So, the *validation* split contains instances of images that are essentially treated as if they were all queries and all tests for Re-Identification.

For of all the images that constitute the validation split, the feature values are computed as explained in 3.5. After that, 100 images of the validation split are randomly taken, and are used as query. Hence, for each of these, the algorithm looks for the people who have resemblances and similarities in the whole validation. This procedure allows us to calculate the mean average precision, given the fact that the labels relating to people's identities are available within the validation split.

In order to evaluate the goodness of the predictions, mean average precision is computed as metric [*mAP*].

$$mAP = \frac{\sum_{q=1}^{Q} AveragePrecisionOf(q)}{Q} \qquad (1)$$

The intuition behind the formula is simple: given query, q, we calculate its corresponding $AveragePrecision$, and then the mean of the all these $AveragePrecision$ scores would give us a single number, called the mAP, which quantifies how good our model is at performing the query.

Once the model has trained, input taken from the *query* folder are passed to the neural network. The output for each of the images is basically an ordered list of images in the *test* folder, which should correspond to the identity of the person searched for in the *query* folder. The ultimate goal is to generate a file containing this list of images for each query image. The answers for the Re-ID task are written in a separate *reid_test.txt* file.

3. Model

The implemented network consists in a backbone based on the ResNet-50 architecture and some layers that have been created in order to fit the best with the task. We chose this type of architecture as it is one of the most common for this task, as stated in [5] at 2.1.5.

3.1. ResNet

The residual neural network (ResNet) takes the inspiration for its principal characteristic from the human brain by implementing a mechanism similar to the one that happens into the pyramidal cell. This particular characteristic is called *skip connection* (Image 2) and allows the network to skip some layers, speeding up training processes.

In this project we implemented the ResNet-50 architecture, proposed by He *et al*. [1] in 2016 in the 34-layer version.

After the standard ResNet-50 architecture that ends with the flattening of the average pooling of the last layer, some additional layers for the two tasks are added, the whole network can be seen in Figure 1.

3.2. After ResNet

After the flattening we obtain a vector of 2048 features. To compress even more the information a fully connected layer of 512 units is placed followed by batch normalization and ReLU activation. The result of this layer follows two different paths. In the first path the attributes of the per-

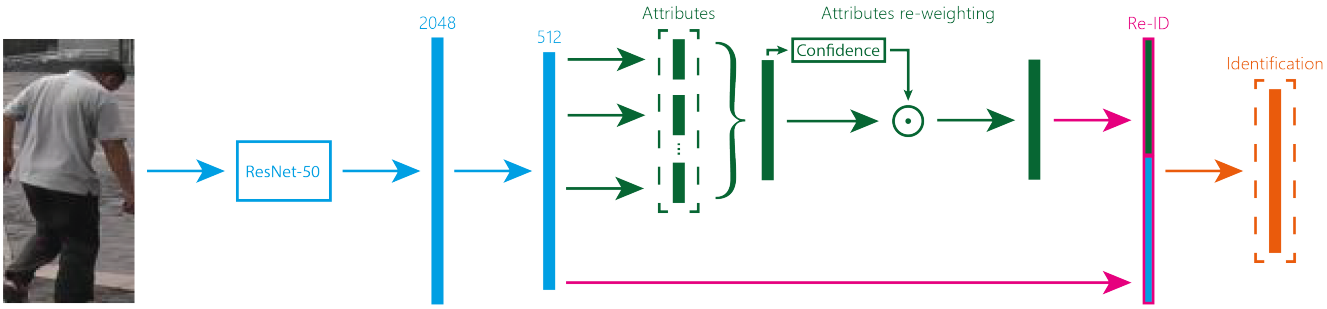

Figure 1. Network structure. The dashed boxes point the losses, the first is relative to attributes classification, the second to Re-Identification.

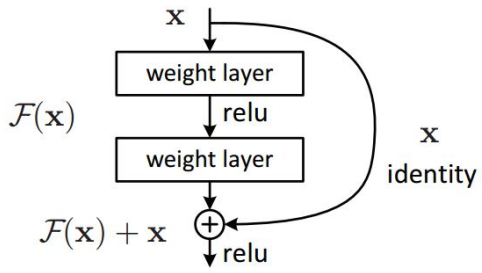

Figure 2. Skip connection.

son are extracted. Those attributes are then re-weighted as explained in 3.4. The second path takes the output of the fully connected layer and concatenates it with the results of the attribute re-weighting. The new concatenated feature vector is then passed to a fully connected layer called *id-fier* to extract the person identity in the training phase. The concatenated feature vector is used in the test step to obtain the similarity between two different images.

Initially, we implemented the layers after the ResNet and the logic ourselves, but after a meeting with the professor we followed [2] to improve the network using the attribute classification.

3.3. Attributes classification

To classify the attributes, the information of the 512-fully connected layer is exploited. After this layer, a parallel sequence of layers is instantiated in a dictionary with key that corresponds to the parameter to be estimated and value a linear layer of output dimension equal to the possible combination of outputs of that key. For example, if we are predicting the *age* attributes the layer will be composed by 4 different output since the possible labels spans from 0 to 3. The loss metric used to train the model is the cross-entropy loss as it is one of the major metrics used in classification tasks.

3.4. Attributes re-weighting

To exploit the information obtained by the attributes classification, a re-weighting has been made as in Lin *et al*. [2]. The set of attribute predictions $\{\tilde{a}^1, \tilde{a}^2, \ldots, \tilde{a}^m\}$ are concatenated to obtain a prediction score vector $\tilde{a}$. Then, the confidence score c for the prediction $\tilde{a}$ is learned:

$$c = \sigma(v\tilde{a} + b) \tag{2}$$

where v is the matrix of the learned weights and b is the bias. Here, we multiply element wise the attributes vector $\tilde{a}$ with the newly learned confidence vector:

$$a = c \cdot \tilde{a} \tag{3}$$

This new vector is then concatenated to the features of the 512-fully connected layer for further identity classification. The idea of this re-weighting is that some attributes can have a relation between them that is not expressed, so re-weighting allows to model this kind of relationship.

3.5. Re-Identification

For every image in the test dataset the network predicts the concatenated feature vector. When the query is fed into the network, the model computes the mean square distance between the query feature vector and the other images feature vectors. The images are then sorted by dissimilarity, the lower dissimilarity value, the grater the probability to be the same person. For evaluation time purposes, only a small amount of images are returned in the list of the results.

This part of the network is trained on the cross-entropy loss as was considered more appropriate to this task with respect to the triplet loss. Zhai *et al*. [6] discussed in detail this choice in their paper. Furthermore, in [5] they state that one of the possible approaches is to treat the Re-ID problem as a classification problem, where each identity is a unique class. Since cross-entropy is good in classification problems this approach was chosen.

3.6. Classes and modules

The files that compose the project are 9 and will be explained below. One of them is the evaluator, since it has not been implemented by us it will not be explained.

3.6.1 Bottleneck

This class constitutes the basic building block of the ResNet-50. It consists in convolution, layer normalization operations and ReLU activations. Here is also implemented the skip connection shown in Figure 2. (Taken from Pytorch repository [4]).

3.6.2 Network

This class defines the architecture of the network. It starts with a convolutional input layer, a layer normalization, a ReLU activation and a MaxPool. Then there are 4 Bottleneck layers with different parameters followed by an adaptive average pooling and the flattening of the last image obtained after all the steps. This part composes the ResNet backbone. Afterwards, a fully connected layer is applied, then the network is branched into two parts as in Figure 1.

The network's class also contains methods for retrieving losses and accuracies for both training and testing processes. Loss during training is obtained by comparing the output predictions for both the attributes and the person Re-IDs with their corresponding ground truths, using cross-entropy. While training, the accuracy for a single example can be seen as the summation of every correct attribute plus the term given by the Re-ID, all of which is divided by the summation of the number of total attributes plus the ID component. During test ran on *validation* split, the loss and accuracy are evaluated using cross-entropy only by tracking the attributes.

3.6.3 Dataset

This class loads the images and returns them together with their labels for *training dataset*, instead for *query* and *test*, it loads image plus filename.

3.6.4 Dataloader

Here, different transformations are applied to all the images, as *Normalize* and *Resize*, or only to *training* images as *RandomRotation*, *RandomCrop* and *RandomHorizontalFlip*. For better improving the Re-ID task, we implemented the *random erasing* data-augmentation technique as suggested in *et al*. [3]. Whereas it might have boosted the performance in the Re-ID task, it yielded worse results for attribute classification, as salient part of the pictures could have been occluded, hence was discarded.

We need to split the training set in training set and validation set. Here, it is important to remark that the validation folder should not contain persons that are also in the train folder. To do so, we put in the training set the 70% of the images plus some other images until the person change. Then, the rest go in the validation set.

Finally, the training and validation set are obtained without doing particular operations.

3.6.5 Training

This file manages the training step. Here inputs are fed into the network and the loss related with the predictions is computed. Then the backpropagation of the loss is executed and the optimizer updates the parameters. The parameters of the optimizer have been empirically selected. The test function can be called in order to get training loss and training accuracy.

3.6.6 Classification

Within this file, there are algorithms for computing accuracies and losses for the classification task. On top of that, there is a function that loads the dataset and passes it to the network; for each input, its output is obtained along with the matching set of attributes. An auxiliary method is used to make the CSV output file the same as the one used for training, and then information is stored in *classification_test.csv*.

3.6.7 Re-ID

Here, there is an algorithm that computes the mean squared error [MSE] between each Re-ID Tensor for both the predictions and the reference image. All predictions for an image are sorted according to the MSE (from smallest to largest) and put into an ordered list of images nouns which is then returned.

During training, a function gets the images within the *validation* split of the *train* dataset, passes them to the network and obtains the Re-ID Tensors. 100 images within the *validation* split serve as query; each of them is passed to the aforementioned algorithm and the mAP is computed on top of their returned list.

During testing, images from the *query* and *test* folder are fed into the network; MSE between the query and the test images is computed. In the end, a list of the first 50 images of the *test* dataset which are presumably corresponding to the same person in the query image is obtained.

3.6.8 Main

The network is instantiated and trained, the model with the best outcomes is saved for further use. Then, the test step generate the annotation csv and the Re-ID txt files.

4. Results

We have obtained a network based on ResNet-50 architecture that is able to classify attributes with 84.7% accuracy and retrieve person identity given a query with a mAP equals to 0.739.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 2

[2] Yutian Lin, Liang Zheng, Zhedong Zheng, Yu Wu, and Yi Yang. Improving person re-identification by attribute and identity learning. *CoRR*, abs/1703.07220, 2017. 3

[3] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification, 2019. 4

[4] PyTorch. Pytorch source code. https://github.com/pytorch/pytorch. 4

[5] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven C. H. Hoi. Deep learning for person re-identification: A survey and outlook. *CoRR*, abs/2001.04193, 2020. 2, 3

[6] Yao Zhai, Xun Guo, Yan Lu, and Houqiang Li. In defense of the classification loss for person re-identification. *CoRR*, abs/1809.05864, 2018. 3