Deep Learning course assignment: an implementation of the *Attribute-Person Recognition* network

Eliana Battisti
223701
eliana.battisti-1@studenti.unitn.it

Davide Dalla Stella
223727
davide.dallastella@studenti.unitn.it

Francesco Trono
221723
francesco.trono@studenti.unitn.it

Abstract

This work aims to develop a neural network capable of solving two tasks: to predict a set of attributes, such as age, gender or colors of clothes, from the image of a person and then to find that same person in a certain set of gallery images shot from different perspectives. In our work, the PyTorch framework is used to develop a custom implementation of the Attribute-Person Recognition *network, originally proposed by Lin et al. (2017) [1], which is capable of predicting all necessary features for both tasks. All the code is written in Python, alternating the different sections using a Jupyter notebook. The network is trained on Google Colab using a Tesla K80 GPU.*

1. Introduction

The objective of this work is to solve two common computer vision tasks using a neural network built with the PyTorch framework [2]. We are given a video-surveillance dataset called *Market-1501_Attribute* [3], containing images of multiple pedestrians, each of which is captured multiple times and from multiple points of view. These images have a set of annotations that specify 27 different attributes of each person. The dataset has three directories:

- *Train*: it contains 12.989 images, with size 64x128px. Each image is annotated with a *person_id* that uniquely identifies the person in the image. The total number of distinct IDs present in the training set is 751.
- *Test*: it contains 19.679 images related to 750 person identities, always with size 64x128px. These identities are *different* from the ones present in the *train* folder. It also has a set of distractors and a set of junk images that, for this specific project, are not placed separately from the actual people images.
- *Queries*: it contains 2.248 images of pedestrians, always with size 64x128px. The IDs in this folder are the same present in the *test* set.

In the first part of the work (*Task 1*), we built a multiclass classifier to predict each of the 27 attributes for each image (*attributes prediction*). The list of attributes is available in Exhibit 1 in the Appendix - note that we purposely combined some binary attributes related to two same categories into two multiclass attributes. In the second part (*Task 2*), we improved our architecture to solve the problem of *person re-identification* ("*Re-ID*"), which aims at identifying a person of interest (*query*) across a *gallery* set of images caught by multiple non-overlapping cameras.

Before introducing the architecture of our choice, we will now present a quick overview of the current literature methods that are available to solve these two tasks. We base our literature review on the survey published by Ye et al. (2020) [4].

Literature review

Our two tasks fall into the *closed-world* setting of the people Re-ID problem. This setting assumes the validity of 5 key conditions: (i) single-modality data (i.e. camera images only); (ii) existing bounding boxes for the regions of interest; (iii) sufficient annotated data across the various cameras; (iv) the annotations are correct (i.e. absence of noise/errors); (v) the *query* person to find is actually present in the *gallery* set. These assumptions allow the development, test and refinement of powerful computer vision models that the research is currently trying to extend to the more natural and complicated *open-world setting* [4].

In the closed-world person Re-ID setting, three main feature learning strategies are currently used by the literature: (a) *global feature representation learning*, which extracts a feature vector for each person image; (b) *local feature representation learning*, which learns to extract body parts with the use of powerful pose estimation methods or simply dividing the image into regions, depending on the macro-area of interest, and then aggregates the features together; (c) *auxiliary feature representation learning*, which reinforces the feature representation with the use of additional annotated information. In our case, the presence of an annotated dataset with pedestrian attributes represents additional, vital semantic information that could guide the learning of better image features. For this reason, we wanted to find a way to exploit this semantic information not only for the *attributes prediction* Task, but also (and especially) for the actual person Re-ID Task.

To achieve this goal, we studied some loss functions that are widely used to drive the feature representation learning, to identify the one that was more suitable for our needs. Before introducing the losses, we must mention that the *pedestrian retrieval* operation is generally performed by extracting the feature representation for the *query* image (person of interest) and the *gallery* images using the Re-ID network built, and then generating a retrieved ranking list by sorting the calculated query-to-gallery similarity – a commonly used metric is the *Euclidean distance*, which can be calculated as the square root of the sum of the squared differences between the two feature tensors.

Among the losses we studied to improve this ranking process, two of them, in particular, caught our attention:

- *Triplet Loss*: it can be integrated directly within the retrieval ranking process: given a sample image (*anchor "i"*), a sample image of the same ID (*positive sample "j"*) and a sample image of a completely different ID (*negative sample "k"*) are taken, forming a *triplet* of samples. In this setting, the *Euclidean distance* "*d(.)"* between the anchor and the positive sample should be smaller than the distance between the anchor and the negative sample by a predefined margin parameter "ρ".

$$\mathcal{L}_{tri}(i,j,k) = max\big(\rho + d_{ij} - d_{ik}, 0\big)$$

- *Identity Loss*: simpler than the *triplet loss*, during the training process it treats person Re-ID as an *image classification problem* where each identity is a distinct class. The output of the backbone is adopted as a feature extractor; then, the identity loss is computed by the simple cross-entropy. The main advantage of using identity loss is that it is easy to train and automatically extracts the hard samples during the training process. Being a standard cross-entropy loss, it requires labeled data.

$$\mathcal{L}_{id} = -\frac{1}{n}\sum_{i=1}^{n} log\big(p(y_i|x_i)\big)$$

Between these two different losses, we thought that the *identity loss* could be more easily integrable within the same network we were training for Task 1 if we considered the ID like an additional "attribute" to predict. It was not immediately clear, though, how to train this network for Task 2 to predict unknown IDs.

2. Proposed solution

Keeping in mind our idea to implement auxiliary feature representation learning by exploiting all data available to us to perform both our tasks, we read several papers proposing architectures and implementations. Among the various papers, we were particularly intrigued by the solution proposed by Lin et al. (2017) [1]. They proposed the *Attribute-Person Recognition* network ("*APR Net*"), a multi-task network which learns a re-ID embedding and at the same time predicts pedestrian attributes.

What we found captivating about this implementation is the fact that they managed to use *one network only* for both tasks simultaneously, instead of using two separate networks. It allows the two tasks to jointly learn from each other, resulting in learning more discriminative feature representations for pedestrians Re-ID.

The architecture they proposed is very simple. Their APR Net starts from a CNN backbone, then contains two prediction parts: one for the *attribute recognition* task and the other for the *identity classification* task.

Given an input sample image, the network first extracts the global feature representation embedding from the last layer of the backbone. It then gives this global embedding vector as input to a set of linear classifiers, to predict the pedestrian attributes. On these predictions, the *attribute recognition loss* is calculated using the predicted and the ground truth labels.

Consistently with the auxiliary features representation learning strategy, they take the predicted attributes as additional cues for identity prediction. In order not to mislead the global identification by possible wrong predictions, they built an *Attribute Reweighting Module* ("*ARM"*): each attribute is re-weighted on the basis of the confidence score of its classifier's prediction, then the reweighted sequence of attributes is concatenated with the global feature embedding vector and fed to a linear ID classifier, on the basis of which a cross-entropy *identity loss* is calculated.

The two losses (*attribute recognition loss* and *identity loss*) are optimized simultaneously and weighted by a hyperparameter, which balances the importance of one loss with respect to the other.

This architecture solution is simple and effective: with a common, joint architecture, the authors demonstrated a possible way in which person re-ID and attribute recognition could benefit each other.

3. Our implementation

We decided, therefore, to reproduce the architecture proposed by Lin et al. (2017) [1] (hereinafter, "our reference paper" or "the paper"), with some customizations and personal choices. Here follows an overview of our implementation. The visual architecture scheme can be found in Figure 1 on the next page.

Architecture

Starting with the backbone, we decided since the beginning of the project to use a ResNet architecture [5]. Compared to a standard AlexNet, a ResNet is specifically designed to benefit from the increased accuracy of a deeper network architecture, but with easier training and optimization. The skip connections within ResNet blocks help, indeed, solve the problem of vanishing gradient in deep networks by allowing alternate shortcut paths for the gradients to flow through. Also, they allow layers to learn

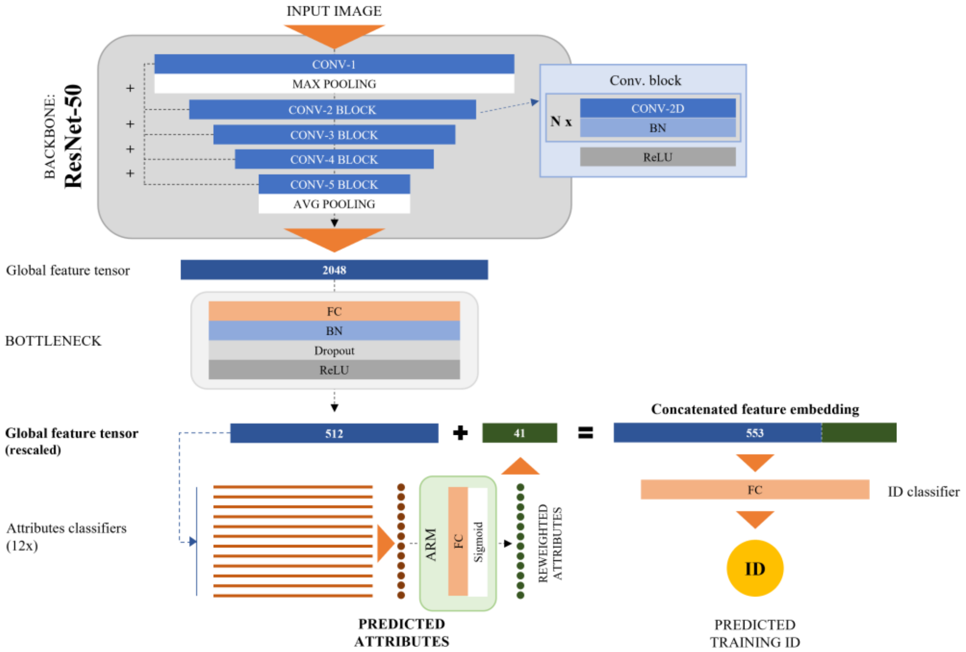

Figure 1. Architecture of our implementation of APR Net, inspired by the work of Lin et al. (2017) [1]. Note that we use the last linear identity classifier only for training purpose, while we use the concatenated feature embedding to evaluate the performance of the network for Task 2.

identity functions, which helps the network not to degrade the performance with the increasing depth. Between the various versions of ResNet (18, 34, 50, 101, 152 layers), we decided to adopt ResNet-50, which, as emerging from He et al. (2015)'s study [5], represents a good balance between depth and performance. In our code, we used PyTorch library's implementation of ResNet-50 [6] and initialized it as pretrained on ImageNet [7], which ensures a faster convergence and better results compared with an initialization on random weights [8].

The standard ResNet, being trained on ImageNet, has a final linear classifier with 1000 classes, which for our specific case was useless. Therefore, we removed this final layer and replaced it with a series of layers and modules replicating our reference paper's architecture. Specifically:

- We added a *linear bottleneck* after the backbone, to reduce the number of features before the final classifiers from 2048 to 512, in order to speed up the convergence. The bottleneck also includes a batch normalization layer and a dropout layer.
- We added a series of *12 linear classifiers*, one per attribute, for Task 1 output predictions. All these classifiers are "parallel", meaning that they are not connected between them but they all draw from the same 512-dim global feature tensor. The number of output classes for each layer depends on the number of labels for its target attribute (see Exhibit 1 in the Appendix).
- Then, we pass the 12 outputs to the *Attribute Re-weighting Module*, created following the indications in the paper: this is a linear layer with a softmax activation function and it is needed to learn the confidence score for each of the 12 predictions. We then weight the 12 predictions on the basis of their confidence score, in order to maximize the weight of the most confident predictions and lower the weight of the most doubtful ones.
- We *concatenate* (i) the tensor with the 12 reweighted attributes to (ii) the feature tensor initially produced by our bottleneck. In this way, we get a new embedding tensor which not only considers the features of the image, but also the features corresponding to the 12 (reweighted) attributes.
- This concatenated embedding tensor is given as input to the *final linear classifier*, which predicts, on the basis of it, the ID for the person. This ID is a 13th prediction that is added to the output list.

The network returns both: (i) the *output* list (12 attributes predictions + ID prediction), and (ii) the concatenated embedding tensor (reweighted attributes + feature tensor). It is important to note that the ID prediction is needed only for training purposes. We explain next why.

Loss function

Following our reference paper, the loss function we adopted is a weighted sum of two losses:

- The *attribute recognition loss* (for Task 1): we sum 12 cross-entropy losses together, one for each predicted attribute.
- (Only for training) The *identity loss*, which, as already explained, is a cross-entropy loss based on the correct classification of the training IDs.

The weight of the two losses is balanced by an *"alpha"* hyperparameter that can be set manually before training. The need for this parameter is represented by the fact that there are 12 losses for Task 1 and only one for Task 2, so we need to manually assign the latter enough importance. In our implementation, *alpha* represents the weight assigned to Task 1's *attribute recognition loss* over the sum of the two losses. Following some tests (see next section), we decided to set *alpha* to 0.7 (70%).

As already introduced, we exclude the *identity loss* during validation and testing: in these phases, we only take into account Task 1 loss. This is due to the specific configuration of the dataset and the requirements of our two tasks. The dataset is indeed structured to consider separate pedestrian identities for training and test, which means that *none* of the 751 training identities is present in the test set. We replicated this same condition while doing the split between training and validation sets. It is therefore adamant that the final *id classification* layer of our network cannot be trained to predict any validation and test identities, being the latter different people from the ones used for training, so the predicted IDs in inference scenarios will clearly be wrong.

For this reason, Task 2 loss is needed only during the training phase: the network must be able to make use of both its predicted attributes to classify the correct training IDs, and its ID classification errors to improve the predictions of the pedestrians' attributes. The ablation studies performed in our reference paper showed that to learn jointly both tasks boosts the accuracy of the attributes predictions, which means that the final ranking of ID predictions on unknown inference identities will be made on the basis of much more confident data.

Therefore, Task 2 will be evaluated during validation on the basis of the concatenated feature embedding using *mean average precision* (mAP). Defining *precision* as the ratio between the correct positive predictions and the total positive predictions returned by the model, and *recall* as the ratio between the correct positive predictions and the total positive ground truths, mAP represents the area under the precision-recall curve, averaged over all queries.

With the joint implementation represented by the APR Net, we are also confident that the network is able not to match any query identity with distractors or junk images present in the gallery: it is indeed highly improbable that the network predicts many of the 12 attributes and an ID with high confidence on pictures not of humans, so we expect actual human pictures to prevail in the generated Re-ID ranking retrieval. It could be also possible to set up some threshold hyperparameters to explicitly flag an image as "junk" in the case too many attributes (or the ID itself) are predicted with a low score.

Code structure

We developed, trained and tested the architecture on Google Colab. Being the architecture one for both tasks, we prepared one Jupyter notebook only with the entire code. The software details and instructions on how to run it are available in the *ReadMe* file of our GitHub repo (link). The structure of the notebook follows in logical order the various steps we followed in developing the solution. For better readability, we put the detailed explanations on the steps performed directly in the *markdown* sections above each code cell.

4. Results

Many parameter configurations were tested to find which one returned the best results.

Initially, we were using as training-validation dataset split a 75%-25% allocation. Following the peculiar needs of our architecture and in order to allow the network to train on a bigger set of identities, we decided to set our training-validation dataset split to 80%-20% instead. We trained our network on 601 pedestrian IDs and validated it on 150 different IDs.

We set as number of training epochs 50, with an early stop callback based on validation loss monitoring and 3 epochs as patience. We also save the weights after each epoch of improvement as ".pt" file and reload the ones from the last best epoch after early stopping. To improve generalization of our network, we used a series of transformations in the Dataset class, like random rotation, random flip, a low pass filter with a 3x3 mask to reduce the noise and made sharpness adjustments.

We noticed that our implementation of the network converges really fast. We think this is due mainly to two factors: (i) we are using a pretrained ResNet backbone, so the training starts with the model already able to extract the key image features; (ii) the joint optimization of the *attribute recognition loss* and the *identity loss* represents a powerful guide for the model during the training.

In our trials, we performed training runs using either the SGD (with Nesterov momentum) optimizer or the Adam optimizer. We can see that with both optimizers we achieve great results: the network is always able to converge to scores ranging, for Task 1, between 80.3-83.3% for validation accuracy and 8.7-10.5% for validation loss, and, for Task 2, between 58.0-64.8% for mAP. For comparison, our reference paper, in the runs performed using Market-1501 dataset and a ResNet-50 backbone, reached 86.6% validation accuracy (Task 1) and 66.89% mAP (Task 2).

With Adam, we had to use a very low learning rate to slow down an immediate convergence. This is normal, considering the design of the Adam optimizer which considers not only momentum, but also adaptive learning rates to converge faster and more efficiently than SGD [8]. Using Adam with a global learning rate and a weight decay rate both at 0.0001 and an alpha loss balancing at 0.7, we were able to achieve our *best results* with 83.19% val accuracy, 8.75% val loss and 64.79% mAP. The convergence was achieved in 6 epochs only. We tried to achieve a slower convergence using SGD with Nesterov: we got smoother metrics charts, but the network was not able to match the performance obtained with Adam, getting close results for only one task between the two, alternating. We report the best scores obtained with different settings in Exhibit 2 in the Appendix.

Appendix

Attribute name	Shortened name	Type
Age	*Age*	Multiclass (4 labels)
Carrying backpack	*Backpack*	Binary
Carrying bag	*Bag*	Binary
Carrying handbag	*Handbag*	Binary
Type of lower-body clothing	*Clothes*	Binary
Length of lower-body clothing	*Down*	Binary
Sleeve length,	*Up*	Binary
Hair length	*Hair*	Binary
Wearing hat	*Hat*	Binary
Gender	*Gender*	Binary
Color of upper-body clothing	*Upcolor*	Multiclass (9 labels: 8 colors + 1 "multicolor")
Color of lower-body clothing	*Downcolor*	Multiclass (10 labels: 9 colors + 1 "multicolor")

Exhibit 1. Full list of the *Market-1501* dataset attributes, as adapted in our implementation.

Optimizer	Learning Rate (LR)	Weight Decay (WD)	Momentum	Alpha (loss balancing)	Comments	Epochs	T1 val accuracy %	T1 val loss %	T2 mAP %
SGD	**0.001**	**-**	**0.5**	**0.6**	**Best Task 2 with SGD**	**21**	**80.79%**	**10.03%**	**63.46%**
SGD	**0.001**	**-**	**0.5**	**0.7**	**Best Task 1 with SGD**	**19**	**82.66%**	**9.03%**	**61.31%**
SGD	0.001	0.01	0.5	0.6	Tested impact of WD	17	80.63%	10.08%	62.81%
SGD	0.001	-	0.5	0.6	No bottleneck layer after backbone (kept 2048 features, not 512)	18	82.49%	8.97%	59.52%
SGD	0.003	0.001	0.4	0.6	Tested faster LR	11	81.39%	9.92%	63.02%
SGD	0.001	-	0.4	0.7	Tested lowest momentum	23	82.39%	9.09%	58.14%
Adam	**0.0001**	**0.0001**	**-**	**0.7**	**Best run - Adam (code submitted)**	**6**	**83.19%**	**8.75%**	**64.79%**
Adam	0.0001	0.0001	-	0.7	Tested addition of dropout layer at 0.2 after backbone	6	83.30%	9.21%	63.15%
Adam	0.0001	-	-	0.6	Lower alpha, no WD	6	81.74%	9.53%	64.64%
Adam	0.0005	-	-	0.7	Tested higher LR	9	82.31%	10.02%	62.42%
Adam	0.0005	0.0001	-	0.7	Tested impact of WD	7	82.47%	9.97%	60.66%
Adam	0.0005	0.0001	-	0.8	Highest alpha (less weight to Task 2 loss)	6	80.33%	10.46%	58.00%
Adam	0.00005	0.01	-	0.5	Lowest alpha and lowest LR	10	81.55%	9.67%	60.86%

Exhibit 2. Results of our best training runs for our implementation of APR Net. We tested different optimizers, global learning rates, impact of momentum and weight decay rates. We also tested the role of the *alpha* loss balancing parameter in driving the performance of both tasks, as well as the utility of the bottleneck layer (feature reduction) after the backbone. We submitted the run with the best Task 1 and Task 2 performance scores.

References

[1] Y. Lin, L. Zheng, Z. Zheng, Y. Wu, Z. Hu, C. Yan and Y. Yang, "Improving Person Re-identification by Attribute and Identity Learning," *Pattern Recognition,* 2017.

[2] PyTorch, "Documentation homepage," Accessed: 2021. [Online]. Available: https://pytorch.org/docs/stable/index.html.

[3] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang and Q. Tian, "Scalable Person Re-identification: A Benchmark," *IEEE International Conference on Computer Vision (ICCV),* 2015.

[4] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao and S. C. H. Hoi, "Deep Learning for Person Re-identification: A Survey and Outlook," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2020.

[5] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2015.

[6] PyTorch, "ResNet," Accessed: 2021. [Online]. Available: https://pytorch.org/hub/pytorch_vision_resnet/.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2009.

[8] E. Ricci and W. Menapace, "Teaching Materials," *Deep Learning Course - University of Trento,* 2021.