Deep Learning Project

Anonymous CVPR submission

Paper ID *****

Abstract

This report describes my solutions in to two problems, which are attributes recognition and person re-identification.

1. Introduction

Generally speaking, I have two research problems: attributes recognition which requires a classification model to distinguish different attributes, and person re-identification (reID) which aims to retrieve images of the same person shown in the query. However, reID problem need features learnt from data to tell the difference from one person to another, semantic attributes learnt from deep neural networks, as an auxilliary feature [3] are helpful to solve reID problem [2]. In this report, I take pretrained resnet50 as my backbone, trained it on Market-1501 dataset in pytorch framework, and try to use the features learnt from attributes recognition model to solve reID task.

2. Classification Task

2.1. Classification Model

The general idea is to handle a multi-label classification problem by dividing it as many binary classification for each attribute. I use pretrained resnet50 from models provided in torchvision, and I tried to add channel attention and spatial attention to improve the original resnet50. There are comparisons between them in classification task and reID task.

2.2. Training Details

In this part, I present implementation details from data preprocessing to tricks when I tuned the network.

- **Data preprocessing.** First thing I have done is preparing data. I read and edited the csv file containing annotations with pandas. I added 2 columns denoting multi-color in up and down clothes, and divided the 'age' column into four columns: 'young', 'teenager', 'adult', 'old', so each attribute becomes binary. I also counted the times of each attribute showing in training data for weighting purpose [1].

- **Validation set obtaining.** In order to validate the model, I split all the labeled data, which contains images of 751 people, into 600 people as training set, 151 people as validation set. First I put images from different people in different folders, then randomly picked 151 folders and get the images of these 151 people in a validation folder, 600 in training folder.

- **Training.** Batch size is 128 for training, the biggest size I can have on my 4GB memory graphics card. My loss function is BCElosswithlogistics, which combines Sigmoid and BCEloss. Moreover, I weighted each attribute according to their showing frequencies, trying to fix the unbalance problem in training data [1]. As for optimizer,at the beginning I used SGD with nesterov momentum, weight-decay and adjustable learning rates. I trained the models for 100 epochs, and the loss drops slowly especially after 30 epochs, then I tried AdamW as my optimizer, the loss drops fast but still remains jumping between 0.002 and 0.001 after 100 epochs. This shows that AdamW yields good convergence, but still may get stuck in local minimum. As a result, I chose AdamW for the first 50 epochs and SGD for the rest 100 epochs, I hoped to get a better result and keep the training loss below 0.001 in 150 epochs.

2.3. Results Analysis

The evaluation of both models is shown in table below. Attention module didn't improve the performance of model, and need more epochs to converge. I think the position I put attention in the model may not be appropriate, and considering I trained all the attributes together, spatial attention, for example, may not find exactly where upper clothes are and didn't work. I also found that attributes with more training data are better recognised, which makes sense. But the unbalance problem is still there, for example, only 1 person wears purple and this happened not to be in my training set, there's no way to predict purple correctly in validation process.

index	resnet50	resnet50 with attention
precision	0.593	0.574
recall	0.549	0.530
f1 score	0.553	0.536

Table 1. Comparison of the two models

2.4. Code Description

In file 'prepare.py', class ImagesData prepare dataset for training and validation, class testData prepare dataset for test. In file 'resnet.py', class resnet is pretrained resnet50 with 32 classes, class res_net is the combination of resnet and attention module. In file 'main.py', function train(), validation() is for train and validate the model, test() is for test purpose and produce 'classification_train.csv'. File 'makeCSV.py' contains functions that build, edit and export CSV file, file 'dataprocess.py' operates images and split train/validation set.

3. Person Re-Identification Task

3.1. ReID Model

I use the resnet50 model in classification task, and fine tuned it for 50 epochs. The idea is to extract features of both query and gallery with the old model, compare the euclidean distance between query and gallery and rank all the images in gallery for each query in ascending order, which means more similar the two image is, the smaller the distance of features.

3.2. Training Details

Here I am talking about all the details in training phase, which is conducted with labeled data so I can judge if the model works or not.

- **Data preparation.** Since I use the same model, data format is the same as that in last task. The ID column is replaced by names of image files, which denote the right correspondence between images in query and gallery. But the other columns contain attributes predictions are what I use for computing distances between images.
- **Validation set obtaining.** Validation set should be totally separated from training set, otherwise the validation results cannot be able to evaluate the model's ability to handle unknown cases. As a result, I picked one image for each person shown in validation set out as a query, and images left as gallery. All images for ReID validation are taken from validation set in classification task.
- **Evaluation metrics.** In ReID task, I use top1 accuracy, top5 accuracy and mean Average Precision as metrics of evaluation. To make it clear, the metrics are mean values of metrics for each query, for example, mAP here is the average of APs of all queries. To improve the performance of the model, I weighted distance of each attributes and then add them together as a distance for two images, instead of directly use the plain euclidean distance. I think I should add more value to attributes that my model can better recognise when measuring distance. The ablation experiment is conducted and the result is in table 2.
- **Test.** The number of images returned for each query is variable, so I tried to find where the true split line is. The main idea is to find a threshold in the ordered distance array, I observed the data and set a little bit higher threshold, because I would rather pick the wrong images than lose any right one.

metrics	weighted distance	unweighted distance
top1 accuracy	0.78	0.787
top5 accuracy	0.94	0.94
mAP	0.523	0.523

Table 2. Comparison of weighted and unweighted distances

3.3. Results Analysis

According to the ablation experiment above, I found that distance weight fails to improve the reID model. The reason I think is that the output of model already considered how reliable the prediction is, if the output has a big absolute value it is confident about the prediction in this attribute. A better way to weight the attributes is build an additional layer after the last full connected layer, to discover the connection between attributes and person ID.

3.4. Code Description

In file 'prepare.py', class testData prepare dataset for model validation here, while it also prepare test dataset in classification task, it returns images and names of images. In file 'evaluate.py', function get_AP is to get average precision of each query, get_CMC is to get cumulative matching characteristic of each query and returns top1 and top5 accuracy. In the 'main' file, function reid(gallerypath, querypath, test = 0) is for reid task, default mode for validation, change the parameter test to other numbers other than 0 will go into test mode, which generates a txt file lists file names.

4. Summary

The comparison between the two models shows that attention does not always work, if it is not proper applied. In fact, I add channel attention before full connected layer, which turns out to be another full connected layer, and it did improve the model, but I cannot tell if it's because attention or another full connection layer. This keeps me thinking: what's the essence of attention, and more general, what's the essence of computer vision in classification problems. I think that no matter what model is employed, CNN or transformer, they are all means to find features and patterns lie in image data and compare the unknown with known to make judgements.
All the experiments are implemented on my laptop: Lenovo R7000, GTX 1650, software: VSCode 2019, Ubuntu 20.04.

References

[1] Dangwei Li, Xiaotang Chen, and Kaiqi Huang. Multi-attribute learning for pedestrian attribute recognition in surveillance scenarios. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 111–115. IEEE, 2015. 1

[2] Yutian Lin, Liang Zheng, Zhedong Zheng, Yu Wu, Zhilan Hu, Chenggang Yan, and Yi Yang. Improving person re-identification by attribute and identity learning. *Pattern Recognition*, 95:151–161, 2019. 1

[3] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven CH Hoi. Deep learning for person re-identification: A survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1