Detecting Wearable Objects via Transfer Learning

Svatopluk Kraus, Pavel Krsek, Kristína Malinovská
Czech Institute of Informatics, Robotics, and Cybernetics
Czech Technical University
Prague, Czech Republic
Email: {svatopluk.kraus,pavel.krsek,kristina.malinovska}@cvut.cz

Matúš Tuna
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava
Bratislava, Slovak Republic
Email: tuna1@uniba.sk

***Abstract*—Transfer learning is a well known technique to circumvent the problem of small datasets in deep machine learning. It has been successfully used in the field of camera surveillance image processing which suffers from poor data quality and quantity. We focused on the task of wearable object detection, namely distinguishing if a person is or is not wearing a backpack. We created new annotations for the DukeMTMC-attribute dataset to overcome the discrepancies among the attributes. We explored transfer learning with a frozen feature extractor as well as the model fine-tuning, which turned out to perform much better. In both setups we found that the Densenet161 is the best from tested architectures. Our best model achieved about 92% balanced accuracy on the testing set.**

***Index Terms*—Image classification, Camera surveillance, Deep learning, Transfer learning**

I. Introduction

Our project, called the Smart Camera, is widely focused on an improvement of monitoring centers. The aim is to improve mainly the automatic evaluation of video from the large numbers of surveillance cameras. Observing and recognition of persons and situations in video from many different cameras is both exhausting and difficult task for operators. Considering information presented to an operator, we would like to reduce it in quantity to make things less exhausting and to add quality to make decision making easier.

Both on-line video processing and off-line forensic exploration of video records are in our focus. An important task is to describe the observed persons by visual attributes (the kind and the color of clothes, carried baggage, ..) and be able to search for them quickly. Our approach is to define appropriate attributes classes, to prepare corresponding automated classifiers, and to establish a process of training the classifiers to be able to generalize over new, unseen data.

Detecting the objects carried by people is a difficult task even in the good quality images. The surveillance systems are often heterogenous. They use cameras with different resolution and positioning. The cameras usually have a quite low resolution and lighting conditions are changing during the recording. Thus the image quality may vary significantly. Further, the field of view recorded by the cameras is not continuous in general.

Our motivation is to overcome these difficulties and be able to process videos from individual cameras and detect the characteristic wearable objects and clothes. Resulting high-level information may be used for finding correspondences in the multiple-camera images. Having a system detecting such attributes might be utilized in forensic search.

In this paper, we present a research on the path towards automated camera surveillance. Due to the legal issues with the actual surveillance data, we restrict our research to freely available datasets and focus on the methodology. As many other state-of-the art computer vision systems, we focus on the deep neural networks (DNN). The main benefit of DNN over other solutions is their generalization capability. Here, we present our approach using existing well-known deep architectures and the methodology of transfer learning in the task of detecting people wearing backpacks from images.

II. Related work

Our work is based on the transfer learning paradigm which is currently very popular in the field of deep machine learning. Supervised learning of deep neural networks requires large number of labeled examples. However for some tasks, a large number of labeled examples is not easily obtainable or simply not available at all. The transfer learning [1] makes use of an existing model, usually trained to perform a more general task, and a freely available big dataset, such as the ImageNet [2]. The general model with trained weights is adapted and tuned up to perform another task on smaller amount of more specific data. Transfer learning has successfully been applied to many tasks including categorization of images, semantic segmentation, but also on the person re-identification task.

A. Transfer learning

Transfer learning can be defined as a process of adapting the source predictive function (neural classifier) to the target domain using the target domain data and labels [3]. The neccessary assumption is that the target and source domain share common low level structure that enables us to reuse certain parts of the source predictive function. In practice, this is usually achieved by first training a source domain predictor (classifier) on the data from the source domain. Subsequently, a target domain predictor is trained using limited data from target domain using the parameters from source domain predictor as an initial parameters. The source domain predictor is either trained as a whole or only certain parts of source domain predictor are trained, usually the fully connected layers responsible for the final prediction, while making use of low-level features extracted from the source

domain that are common for both the source or target domains. The best example are convolutional neural networks trained on large dataset of natural images (such as ImageNet) that develop features such as edge detectors that are useful for all natural image classification tasks. Parameters of networks trained on the ImageNet dataset are used nowadays as a starting point for training of models for many natural image classification tasks.

B. Feature extractors

In the case of image processing, transfer is usually made from a very big domain, such as the ImageNet classification [2], to a smaller domain, such as a classification of more specific features or objects. The ImageNet benchmark is one of the biggest databases of assorted images and due to its diversity, it is very useful for low-level feature extraction in the bottom-most convolutional layer of the embedding architectures. The most prominent architectures in current state of the art of image processing are the VGG, the ResNet, and the DenseNet.

VGG network architecture was introduced by Simonyan and Zisserman in 2014 [4]. Their main idea was to replace one convolutional layer with large receptive field with several convolutional layers that use smaller receptive field. This enabled a significant increase of the number of convolutional layers (depth) in the network. Authors showed that this increase of depth leads to major increase of performance on image classification problems. This architecture, namely its variant with 16 convolutional layers is still considered a benchmark in image classification.

Residual network (ResNet) by He et al. from 2015 [5] is another key architectural innovation that enabled the training of much deeper neural network models. Resnet architecture addresses the key problem of constructing very deep architectures, namely the vanishing gradients, by introducing the so called residual blocks. A residual block contains a number of convolutional layers and a skip connection which copies the input to the block and adds it to its output. A skip connection enables the gradient from the layer above to bypass the layers in the residual block during backpropagation. This mechanism leads to better propagation of gradients through the network allowing much deeper network architectures. The authors empirically demonstrated that this increased depth leads to a superior performance across various image classification tasks.

DenseNet from 2017 proposed by Huang et al. [6] expands upon the idea of residual connections by introducing blocks of convolutional layers connected by dense connections. Dense connections are used to make an input to the next block from the outputs of all preceding blocks. Unlike ResNet, where the output of preceding block is added to the output of next block using summation, in DenseNet, the outputs of preceding blocks are simply concatenated and used as an input to the next block. These blocks encapsulating multiple layers connected by dense connections are called dense blocks. Such blocks can contain tens of convolutional layers. DenseNet alleviates the vanishing gradient problem as well as it enables the individual convolutional layers to exploit information from different parts of the model, not just from the input of preceding layer, which increases the parameter efficiency of the model. On several image classification tasks, DenseNets exhibit accuracy similar to ResNets, but using a fraction of parameters.

C. Re-identification and attribute recognition

The transfer learning paradigm has been successfully applied in the domain of human re-identification. Pioneers in this work, Li et al. [7], proposed a metric-transfer approach and outperformed former methods based on the visual features extraction and matching. In the following work, a multi-task paradigm has been employed. Geng et al. [8] enhanced the performance of the transferred model by employing person-identification and person-classification (distinguishing one person from others). Chen et al. [9] proposed joint training of the identification subnet with an additional network performing a ranking task.

Lin et al. [10] came with an idea to include feature recognition to enhance the performance of the neural model in the re-identification task. Within their work, they re-annotated two re-identification benchmarks: DukeMTMC-reID and Market-1501. Their results show that learning through optimizing a composite loss of both re-identification and attribute classification leads to better performance in re-identification task. A further step in this line was made by Wang et al. [11], who made use of the attribute annotations in case of semi-supervised learning problem, where labels are sparse. Their assumption was that attributes might serve for as defining features for learning person identities in the unlabeled data.

Yu et al. [12] showed that if using a deep architecture with special detector layers, weakly-supervised learning suffices to classify pedestrian attributes without the need of bounding box annotations. This approach, which does not use any form of transfer learning, would indeed require a large dataset, which is usually a problem for real-world camera surveillance applications. Latest work by Xiang et al. [13] shows how incremental few-shot learning can overcome the problems with the amount of labeled data and rigidly predefined feature sets needed for majority of the classification systems. An interesting example of deep learning in attribute recognition task is the deep hierarchical contexts model by Li et al. [14]. It is a precisely engineered architecture which builds up pose-based deep representations of humans, compares them, and provides a scene-level feature description, from which it derives attributes. For example it detects the activity people are doing (such as skiining) and estimates the wearable attributes semantically related to the activity (sunglasses) in all people from the group. Thus it can detect attributes that might, in some cases, not be detectable by a human observer. Another very novel model was proposed by Ji et al. [15], who combined the standard CNN approach with recurrent neural networks for representing attributes as some kinds of sentences with recurrent context capturing the relations among the attributes.

III. Dataset

Our primary motivation is to build a system for feature recognition for automatic camera surveillance. Such a system must work robustly with different camera types, resolutions, view-points, etc. Since there is a limited number of suitable datasets for our task, we finally chose to work with the available DukeMTMC-attribute dataset [10], which adds additional annotations to the DukeMTMC-reID dataset [16].

DukeMTMC-reID is a subset of DukeMTMC [17], a re-identification dataset created from 85-minute long high-resolution footage from 8 different cameras. Images are captured every 120 frames and they are cropped to bounding boxes containing people. DukeMTMC-attribute consists of 702 training and 1100 testing entities (individual people) from DukeMTMC-reID annotated with 24 additional features related to visible properties, such as the gender (male/female), the properties of outfit (upper-part-black) or whether they are carrying items such as bags or backpacks. For the purposes of our work, we chose just the *backpack* attribute. Note that, despite particular visual similarity among backpacks and bags, we focused only on regular backpacks and consider people wearing (only) bags as negatives.

Our first experiments with original DukeMTMC-attribute dataset showed that the annotations are not fully suitable for our purpose, because the original attributes are assigned based on the entities, rather than visual assessment of the attribute in the individual images. Such annotations are beneficial for enhancing the performance in the re-identification task as shown by the authors [10], but the attribute relevance might be restricted when we use it for recognition of carrying backpack on separate images. There are some mismatches in annotation relevant to our task. Figure 1 shows examples of the mismatches as: occlusion by objects, occlusion by person with different attribute value, visibility of backpack, and few real annotation errors.

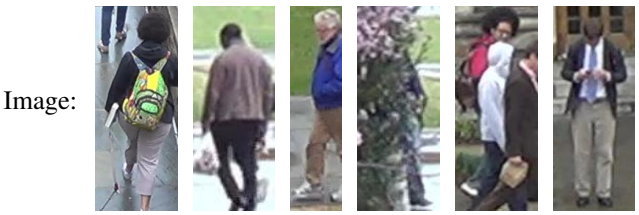


Fig. 1: Example of images and corresponding original annotation for backpack attribute, which are not correct in our task. You can see wrong annotation, bad cropping, object occlusion, multiple occluded persons, and front view which can not be decided.

The mismatches can dramatically influence the process of learning and evaluation of our classifiers. Therefore, we decided to re-annotate the dataset. Based on our observation, we re-annotated the images into three classes: with backpack, without backpack, and uncertain. We put the image to uncertain class either when there were more than one person in an image or when the backpack might not be observed.

We re-annotated each image separately. Our annotation tool was designed to show the images in random order, so the operator could classify samples based only on a single image of an entity, rather than from multiple images of the same entity. From the total of 34183 original annotations: 61.8% were unchanged, 37.0% were marked as uncertain, and 1.2% were changed to the opposite category. It means that 13045 annotations were modified, of which 3.0% was changed into opposite category and 97% was marked as uncertain. We published the resulting annotations with pointers to the DukeMTMC-attribute data items as text files[1]. Examples of images with our annotations for the three categories are displayed in Fig. 2. Note that, the original images in dataset are already cropped out of the camera footage and thus have different sizes and resolutions.

Fig. 2: Example of images from re-annotated dataset divided into three sets according to our manual annotation.

For the purposes of our work, we put the uncertain class aside. We keep the original data split by different entities, not by samples, so all images of one person originally present in the training set remain in the training set and the same applies to the testing set. Even though our re-annotations disqualified some samples, the ratio of the training and testing samples remains similar to the original dataset.

Further on, as our task is a binary classification problem, the positive class means that the person is wearing the backpack and negative class means the opposite. We display the numbers of training and testing images in the dataset after our re-annotation in Table I. The same statistics on the entity level is shown in in Table II. As outlined above, some small portion

[1]https://github.com/pers-attrib/DukeMTMC-backpack

of images of the entities that were attributed as wearing a backpack were transferred to the opposite category (and vice versa). Since the attributes in the original dataset were known and assigned for all images of one entity, not per particular image of the entity, they were indeed labeled as wearing the backpack even if it was not visible in some of them.

TABLE I: Statistics of the dataset classes by image.

	Positive	Negative	All
Train	6540	3593	10133
Test	6911	4491	11402
All	13451	8084	21535

TABLE II: Statistics of the dataset classes by entity.

	Positive	Negative	Both	All
Train	402	180	115	697
Test	737	331	31	1099
All	1139	511	146	1796

In our re-annotated data there is a noticeable imbalance between the positive and the negative examples. In this stage of our research, we decided not to compensate for it neither in terms of data augmentation nor in designing special loss function in our models. However, we take this imbalance into account and use the balanced accuracy (BACC) as a performance measure for our experiments. Mathematically, balanced accuracy can be expressed as:

$$\mathrm{BACC} = (\mathrm{TP}/\mathrm{P} + \mathrm{TN}/\mathrm{N})/2, \tag{1}$$

where TP stands for classified true positives, P for all positive examples, and TN for classified true negatives and N for all negative class samples. Interestingly, our models were quite robust and the imbalance did not influence the classification results as we demonstrate in the following section.

Note that, our split between training and testing sets is about 50-to-50% ratio, with training set even a bit smaller than testing one. This is not usual, since mostly 80-to-20% or similar splits are used in the deep network training and evaluation. Thus, our task is harder and worse classification accuracy would be expected compared to other similar tasks. Nevertheless, we wanted to keep the closest ratio to the original split done by Lin et al. [10]. Additionally, the particular dataset split per entities and train-to-test ratio makes the generalization more challenging and reveals more of the models' capacity.

IV. Experiments and results

In first experiments with our re-annotated dataset for a binary classification of people wearing backpacks, we worked with three widely-used neural networks architectures for classification of images: the VGG, the ResNet, and the DenseNet. We provided basic information on these architectures in Sec. II. Their implementations are freely available in PyTorch [18] also with pre-trained ImageNet weights. We evaluated two different depths of each architecture. That brings us diverse sizes of feature vectors as well as a possibility to make decision about generalization capacity based on depth of the model. The complete list of the 6 architectures with some basic information is displayed in Table III.

TABLE III: Deep architectures used in our experiments.

Architecture	Layers all	Fully conn.	Parameters	Feature size
VGG16	16	4	134,264,641	4,096
VGG19	19	4	139,574,337	4,096
ResNet34	34	1	21,285,185	512
ResNet50	50	1	23,510,081	2,048
DenseNet121	121	1	6,954,881	1,024
DenseNet161	161	1	26,474,209	2,208

We chose to experimentally evaluate two different paradigms in transfer learning [19]. In the first case, we used a feature extractor with *frozen weights*, where the weights of the convolutional part of the original network trained on the ImageNet dataset are left intact, while the topmost fully connected layer of the model was replaced by a new classifier network tailored for the new task. We gathered the features of the images from our dataset using the feature extractor. Subsequently, we trained a new classifier from the scratch to perform the binary classification. This approach was proven to bring good results in various tasks [20]. We are referring to it as setup A.

In the case of transfer learning with *model fine-tuning*, we also started with a network trained on the ImageNet, but unlike the first setup, the whole model was further trained to perform the new task on the target dataset. The architecture of the feature extractor part of the pre-trained network remained the same and only the topmost layer of the classifier was adapted to fit the new task. In the case of model fine-tuning, we used only a small learning rate to prevent losing the feature structure from the rich ImageNet dataset. Nevertheless, we expected that the network would quite soon converge in the new task. We labeled this learning as setup B.

A. Data analysis and preparation

1) Data split and augmentation: For the requirements of setup A and to speed up the network convergence in setup B, we normalized our dataset according to PyTorch setting used for pre-training of the models. Given the fact that the dimensionality of images in the ImageNet dataset is 224×224 pixels and the dimensionality of images in our dataset varies between approximately 100×240 pixels, we also needed to adjust the size of the images.

We experimentally evaluated several techniques, for example positioning the image in one corner of the 224×224 square and filling the remainder with the mean value, so the network would ignore it. Finally, we decided to use the same method as Lin et al. [10], which was just to stretch the image to the desired size as illustrated in Fig. 3. Albeit the stretched images did not look very natural for a human, there were no significant differences in the network performance.

Fig. 3: Two examples of images before and after resizing. The original images are displayed on the left of the resized ones, the first is 97×245px and the second is 97×231px.

To compensate for the relatively small size of our dataset, we decided to augment our data by flipping each image horizontally as suggested by [21]. We consider horizontal flipping as not harmful to the plausibility of our dataset. We used the same operation on both training and testing sets to keep similar data distribution.

B. Model selection

1) Experimental setup and hyperparameters: In both setups of our experiments, we compared all six architectures from Table III pre-trained on the ImageNet dataset. We used the same data pre-processing pipeline in all experiments. In setup A, we evaluated several classification architectures as described below, while in setup B only the last fully connected layer was replaced with a layer connected to one output neuron with sigmoid activation function.

In setup A, we initialized weights using He Normal initialization [22] which limits the range of the random initialization of weights based on the size of the previous layer, and thus speeds up the convergence of the model. This is useful for us, since hidden layer sizes in our tested classifiers vary. In setup B, we used standard Xavier weight initialization [23].

For all experiments, we used Binary Cross Entropy (BCE) cost function:

$$BCE = -\sum_{n}^{N} \left[t_k^n \ln y_k^n + (1 - t_k^n) \ln(1 - y_k^n) \right], \qquad (2)$$

where t_k^n stands for the desired value and y_k^n for the network output. We trained all our networks using well established ADAM optimizer algorithm [24] with the same parameters across all experiments. These parameters were $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^8$. We did not use any weight decay.

Due to the class imbalance in our dataset, we chose to use the balanced accuracy (BACC, Eq. 1) as the performance measure. Nevertheless, in order to compare our models with the related work, we computed also the standard accuracy (ACC) of our best models.

We determined the optimal combination of learning rate and minibatch size using grid search. For every combination of selected minibatch sizes and learning rates, we repeated the experiment 10 times in setup A and 5 times in setup B, and averaged the results across runs for the particular combination of hyperparameters. We show our results on the DenseNet161 architecture in setup A in Table IV and for ResNet50 architecture in setup B in Table V.

TABLE IV: Setup A: BACC for hyperparameter search on DenseNet161 and Linear2d classifier.

	Batch size		
Learning rate	**4**	**16**	**64**
0.01	79.12 ± 0.36	79.31 ± 0.31	79.31 ± 0.26
0.001	79.34 ± 0.40	79.54 ± 0.29	79.89 ± 0.31
0.0001	80.04 ± 0.22	80.21 ± 0.22	80.20 ± 0.21
0.00001	80.04 ± 0.15	79.91 ± 0.11	79.71 ± 0.03

TABLE V: Setup B: BACC for hyperparameter search on ResNet50.

	Batch size		
Learning rate	**4**	**16**	**64**
0.001	81.68 ± 0.53	84.98 ± 0.62	87.73 ± 0.76
0.0001	90.25 ± 0.61	91.36 ± 0.22	91.24 ± 0.25
0.00001	91.57 ± 0.25	90.17 ± 0.24	88.45 ± 0.28

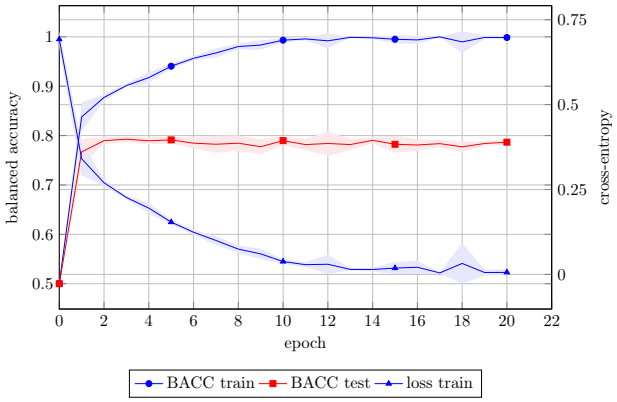
We identified different combinations of learning rates and minibatch sizes that performed the best on the test set. Surprisingly, we were able to find a combination that performs reasonably well in both setup A and setup B, which was learning rate 0.0001 and minibatch size 16. We used these hyperparameters in all subsequent experiments. Even though that this combination was not the best in setup B (the best result was achieved with learning rate 0.00001 and minibatch size 4), we chose minibatch size 16 because of larger minibatches lead to faster training (wall clock time), especially when computing on the GPUs.

2) Classifier architectures: In the A setup, we experimented with the architecture of the trained classifiers. We used only fully connected feedforward networks with different number of hidden layers and their sizes. As well as in setup B, the output of all classifiers is one neuron with sigmoidal activation function, while for hidden layers we used RELU. We experimented with different size of the hidden layer derived from the size of the input layer which varies with particular feature extractor architecture. The names and parameters of tested classifiers are displayed in Table VI, where N denotes the number of neurons in the input layer. We summarize the best results in terms of balanced accuracy for all six feature extractors and all five classifiers in Table VII.

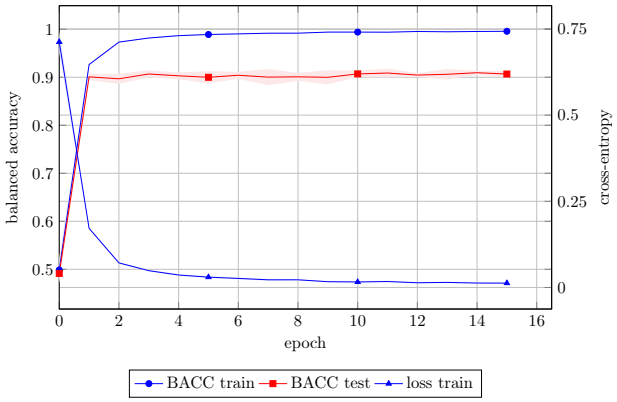
TABLE VI: Setup A: classifiers

Classifier name	**layers**	**hidden**
Linear1	2	-
Linear2a	3	$N/4$
Linear2b	3	$N/2$
Linear2c	3	$3N/4$
Linear2d	3	N

3) Training progress: During the training of our models the networks usually converged to 100% BACC on the training set in less than 20 epochs. Fig. 4 displays the training progress in terms of BACC on the training and testing set and the training



(a) Setup A - DenseNet161 with Linear2d classifier



(b) Setup B - DenseNet161

Fig. 4: Training progress for setup A and setup B with DenseNet161 feature extractor.

TABLE VII: Setup A: BACC for all feature extractors and classifiers.

	Feature extractor					
Classifier	**Res34**	**Res50**	**VGG16**	**VGG19**	**Dense121**	**Dense161**
Linear1	75.58	77.27	75.42	73.55	77.85	79.53
Linear2a	77.42	78.64	74.54	73.05	79.03	80.09
Linear2b	77.66	78.58	74.35	73.12	79.25	80.07
Linear2c	77.73	78.39	74.32	73.15	79.18	80.25
Linear2d	77.71	78.35	74.18	72.91	79.01	80.21

TABLE IX: Setup B: best testing BACC and ACC.

Architecture	**BACC**	**ACC**
ResNet34	90.99 ± 0.31	91.88 ± 0.28
ResNet50	90.43 ± 0.16	91.30 ± 0.27
VGG16	88.80 ± 0.52	89.61 ± 0.39
VGG19	87.52 ± 0.99	88.40 ± 1.16
DenseNet121	91.55 ± 0.31	92.34 ± 0.25
DenseNet161	91.88 ± 0.22	92.65 ± 0.14

loss for setup A (4a) and setup B (4b) with DenseNet161 architecture.

C. Best models and results

In order to find the optimal model, we compared the above mentioned architectures using previously determined learning rate and minibatch size. The highest ACC and BACC on the testing data for every network architecture was averaged across 10 runs in setup A and 5 runs in setup B. We show the results in Table VIII and Table IX. The network architecture with highest accuracy on test data was DenseNet161. This is consistent with the performance of deep neural networks on other image based datasets, where increased depth of neural network architecture generally leads to better generalization.

TABLE VIII: Setup A: best testing BACC and ACC.

Architecture	**Classfier**	**BACC**	**ACC**
ResNet34	Linear2c	77.73 ± 0.19	79.03 ± 0.14
ResNet50	Linear2a	78.64 ± 0.16	80.01 ± 0.16
VGG16	Linear1	75.42 ± 0.08	76.88 ± 0.09
VGG19	Linear1	73.55 ± 0.12	75.29 ± 0.09
DenseNet121	Linear2b	79.25 ± 0.20	80.56 ± 0.15
DenseNet161	Linear2c	80.25 ± 0.22	81.47 ± 0.15

D. Analysis of feature vectors

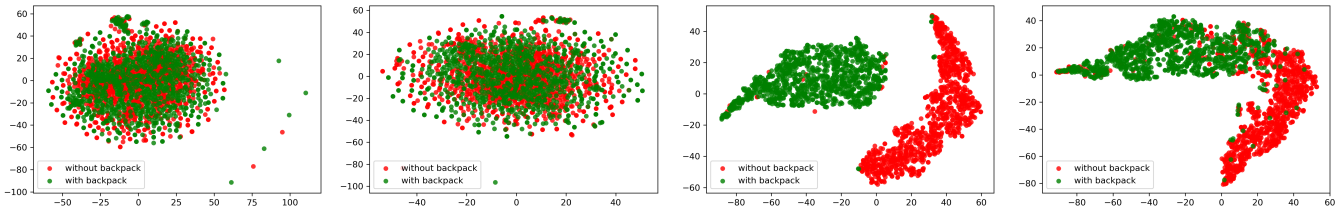
To take a closer look at the extracted features that we use for classification we used the well known t-SNE algortihm [25]. We used freely available t-SNE implementation from the Python-based Scikit-learn library [26] with the default parameters.

We randomly selected 1000 samples from each class, i.e. images of people with backpacks and without backpacks separately from the testing set. Note that, our choice to use a fixed number of random samples from each class does not respect the real distribution of the classes in the dataset which is actually slightly imbalanced.

Fig. 5 illustrates the t-SNE for the DenseNet161 feature vectors from setup A (5a–5b) and setup B (5c–5d). In the figures, it is clearly visible that the features from the feature extractor with frozen weights are very different from the fine-tuned ones and that it is obviously a harder task in setup A, which explains the noticeably worse performance of the models.

V. Discussion and Future Work

Our chosen deep image classification architectures are tracing the evolution of CNNs and how the state of the art can be improved by models with smaller size and still bigger depth. While the VGG16 has 16 layers and more than 130 milion parameters, the Densenet161 has 161 layers and about 5-times less parameters (Table III). In setup A, we were testing how well this deep architectures generalize using features extracted only from the ImageNet. In setup B, we tuned up the features with the target data, which turned out to be crucial for successful transfer learning in the pedestrian attribute recognition domain.



(a) Setup A - training set (b) Setup A - testing set (c) Setup B - training set (d) Setup B - testing set

Fig. 5: t-SNE for 1000 randomly chosen DenseNet161 features with frozen weights (setup A) and fine-tuning (setup B).

For each of the tested architectures, we evaluated two variants with different depths, which got us quite different feature vector sizes for tested models. The VGG models got the worst results in both setups among others, even though their feature vectors are the biggest ones. Such results could be expected as it is the oldest architecture from the tested ones and batch normalization layers are not included in the design. There are 4 fully connected layers used as a classifier at the top of the model which could transform the general convolutional features into specific ones more than in other architectures. We replaced only the last fully connected layer by our classifiers to be in line with other architectures. As shown in Table VIII, the best classifier is Linear1 for both variant the VGG16 and the VGG19. We assume that our more complex classifiers tend to overfit more in this case when there are some original fully connected layers left in the tested model.

The batch normalization layers in the ResNet led to better generalization, while the best classifiers were Linear2c for the ResNet34 and Linear2a for the Resnet50. It looks that the model with smaller depth needs a classifier with bigger capacity as the features are not as good as in the case of the deeper variant. The DenseNet models got the best results among others. It might be expected as both normalization layers are included in the design and the dense connections bring features from bottom layers to the top ones. The best classifiers were Linear2b for the DensetNet121 and Linear2c for the DensetNet161.

The differences in performance among the three architectures are consistent within both setups. Apparently, the newer and deeper architectures perform better, which is in line with the findings from other image recognition studies. As we intuitively expected, setup B with fine-tuning achieved much better performance (cca 10% increase in accuracy) on the testing dataset compared to setup A.

It is most likely due to the fact that the distribution of the data in our dataset is substantially different from the distribution of data in the ImageNet dataset, that was used for the pre-training of the tested feature extractors. Apart from the actual content, images in our dataset have different aspect ratio, lower resolution, and lower visual quality. This is evident from the differences in the BACC as well as from the feature visualizations using the t-SNE method (Fig. 5). Obviously, features from the fixed feature extractor are very difficult to separate, which explains the performance gap between the setups.

Since our re-annotated dataset is unique, we cannot make a full comparison with the related work. There was a small portion (1.2%) of images for which the annotation actually changed from positive to negative and vice versa. Additionally, the majority of the altered annotations belonged to the uncertain category which we did not use in our experiments. Nevertheless, we kept the same split to training and testing data as was in the original DukeMTMC-attribute dataset which left us with slightly bigger size of the testing set compared to the training set. We can say that this way the omission of the uncertain samples made both our task easier in some respects (data validation) and harder in other (bigger testing set). Since no entity present on testing images was presented to the models during the training, we can assume our models really learned to generalize well.

The accuracy for the backpack attribute reported by Lin et al. [10] for DukeMTMC-attribute was 77.28%. Our best model in setup B reached much higher accuracy, namely 92.65%, which is 15.37% more. Even our best model from setup A with frozen weights outperformed it by 4.19%. Li et al. [27] report about 77% of accuracy in backpack recognition on the RAP benchmark dataset. Deng et al. [28] were able to reach about 70% maximum accuracy on their PETA benchmark dataset. All of the above-mentioned results were obtained via multi-attribute classification. The apparent difference in the single-class versus multi-class models leads us to think that some kind of ensembling might be considered for reaching a better performance in the task of pedestrian attribute recognition.

The bag attribute was one of the other attributes annotated in original image dataset. There are many images with such attribute in the dataset. From some viewpoints, it's hard to distinguish between the backpack and bag attribute even for human. It seems that our models deal with that quite well.

We did not use any technique to deal with the issue of the unbalance between positive and negative classes. Instead, we used balanced accuracy to evaluate performance of the models. This suggests, that there is still room to improve results, for instance through the cost function. Additionally, insight from setup A suggests that setup B may benefit from more fully connected layers.

In the future work we would like to expand our dataset and apply our method to other attributes besides the backpack. We would also like to explore the methods of semi-supervised

learning that would enable us to improve the performance of our models using a large unlabeled dataset in conjunction with our existing re-labeled dataset.

VI. Conclusion

We explored the deep transfer learning paradigm in the task of pedestrian attribute recognition, which is one of the hallmarks of automated camera surveillance. In our current work we focused solely on the backpack attribute, i.e. classifying people in the images based on whether they were carrying a backpack. For our purposes we re-annotated and slightly pruned the DukeMTMC-attribute dataset.

We used transfer learning to train three image classification architectures, the VGG, the ResNet and the DenseNet to perform our binary classification task. We explored the frozen feature weight approach (setup A) and the model fine-tuning (setup B). DenseNet161 architecture reached the best performance in both tested setups, namely about 80% BACC in setup A and about 92% BACC in setup B.

The results from both setups reached quite good performance taking into account the apparent differences among the source and the target dataset. The ImageNet dataset used for pre-training is neither destined for attributes recognition, nor similar in terms of image quality and feature distribution. Images in our dataset were taken from various viewpoints, in various distances, and by different cameras and are mostly narrow rectangle-shaped croppings from surveillance videos. Moreover, keeping the original data split by DukeMTMC-attribute rendered the size of our training set rather small and testing size rather large compared to the usual deep learning setups making the task harder, but proved that our best models generalize well because the actual people observed in the testing set were never present in the training set.

In conclusion, our results suggest that transfer learning is a good approach to tackle the lack of quality and quantity of data in the single-attribute recognition task in the domain of video surveillance.

Acknowledgment

This research was supported by the Ministry of the Interior of the Czech Republic under Project VI20172019082 - Smart Camera. The work of Matúš Tuna was supported by Slovak Grant Agency for Science (VEGA), project 1/0798/18.

References

[1] L. Y. Pratt, "Discriminability-based transfer between neural networks," in *Advances in neural information processing systems*, 1993, pp. 204–211.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[3] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, p. 9, 2016.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[7] W. Li, R. Zhao, and X. Wang, "Human reidentification with transferred metric learning," in *Asian conference on computer vision*. Springer, 2012, pp. 31–44.

[8] M. Geng, Y. Wang, T. Xiang, and Y. Tian, "Deep transfer learning for person re-identification," *arXiv preprint arXiv:1611.05244*, 2016.

[9] W. Chen, X. Chen, J. Zhang, and K. Huang, "A multi-task deep network for person re-identification," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[10] Y. lin, L. Zheng, Z. Zheng, Y. Wu, and Y. Yang, "Improving person re-identification by attribute and identity learning," *arXiv preprint arXiv:1703.07220*, 2017.

[11] J. Wang, X. Zhu, S. Gong, and W. Li, "Transferable joint attribute-identity deep learning for unsupervised person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2275–2284.

[12] K. Yu, B. Leng, Z. Zhang, D. Li, and K. Huang, "Weakly-supervised learning of mid-level features for pedestrian attribute recognition and localization," *arXiv preprint arXiv:1611.05603*, 2016.

[13] L. Xiang, X. Jin, G. Ding, J. Han, and L. Li, "Incremental few-shot learning for pedestrian attribute recognition," *arXiv preprint arXiv:1906.00330*, 2019.

[14] Y. Li, C. Huang, C. C. Loy, and X. Tang, "Human attribute recognition by deep hierarchical contexts," in *European Conference on Computer Vision*. Springer, 2016, pp. 684–700.

[15] Z. Ji, W. Zheng, and Y. Pang, "Deep pedestrian attribute recognition based on lstm," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 151–155.

[16] Z. Zheng, L. Zheng, and Y. Yang, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," *arXiv preprint arXiv:1701.07717*, 2017.

[17] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[20] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[27] D. Li, Z. Zhang, X. Chen, H. Ling, and K. Huang, "A richly annotated dataset for pedestrian attribute recognition," *arXiv preprint arXiv:1603.07054*, 2016.

[28] Y. Deng, P. Luo, C. C. Loy, and X. Tang, "Learning to recognize pedestrian attribute," *arXiv preprint arXiv:1501.00901*, 2015.