An Improved Deep Learning Architecture for Person Re-Identification

Ejaz Ahmed
University of Maryland
3364 A.V. Williams, College Park, MD 20740
ejaz@umd.edu

Michael Jones and Tim K. Marks
Mitsubishi Electric Research Labs
201 Broadway, Cambridge, MA 02139
{mjones, tmarks}@merl.com

Abstract

In this work, we propose a method for simultaneously learning features and a corresponding similarity metric for person re-identification. We present a deep convolutional architecture with layers specially designed to address the problem of re-identification. Given a pair of images as input, our network outputs a similarity value indicating whether the two input images depict the same person. Novel elements of our architecture include a layer that computes cross-input neighborhood differences, which capture local relationships between the two input images based on mid-level features from each input image. A high-level summary of the outputs of this layer is computed by a layer of patch summary features, which are then spatially integrated in subsequent layers. Our method significantly outperforms the state of the art on both a large data set (CUHK03) and a medium-sized data set (CUHK01), and is resistant to overfitting. We also demonstrate that by initially training on an unrelated large data set before fine-tuning on a small target data set, our network can achieve results comparable to the state of the art even on a small data set (VIPeR).

1. Introduction

Person re-identification is the problem of identifying people across images that have been taken using different cameras, or across time using a single camera. Re-identification is an important capability for surveillance systems as well as human-computer interaction systems. It is an especially difficult problem, because large variations in viewpoint and lighting across different views can cause two images of the same person to look quite different and can cause images of different people to look very similar. See Figure 1 for some difficult examples. The problem of re-identification is usually formulated in a similar way to face recognition. A typical re-identification system takes as input two images, each of which usually contains a person's full body, and outputs either a similarity score between the two images or a classification of the pair of images as *same* (if the two images depict the same person) or *different* (if the images are of different people). In this paper, we follow this approach and use a novel deep learning network to assign similarity scores to pairs of images of human bodies. Our network architecture includes two novel layers: a neighborhood difference layer that compares convolutional image features in each patch of one input image to the same features computed on nearby patches in the other input image, and a subsequent layer whose features summarize each patch's neighborhood differences. These novel aspects of our network lead to large improvements over the previous state of the art on the CUHK03 and CUHK01 data sets. We also show that our method is less prone to overfitting on small training sets. Results on CUHK01 and VIPeR demonstrate its effectiveness on smaller data sets.

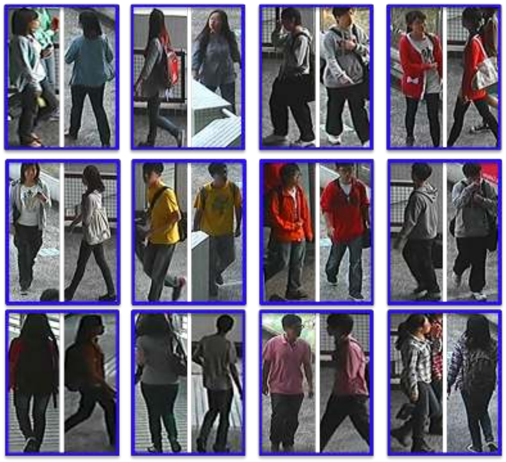


Figure 1: Examples of true positives (first row), false positives (second row), and true negatives (bottom row) for our trained network on CUHK03. More results can be found in the supplementary material.

2. Related Work

2.1. Overview of Previous Re-Identification Work

Typically, methods for re-identification include two components: a method for extracting features from input images, and a metric for comparing those features across images. Research on re-identification usually focuses either on finding an improved set of features ([26, 28, 31]), finding an improved similarity metric for comparing features ([12, 14, 20, 17, 29]), or a combination of both ([25, 11, 16]). The basic idea behind the search for better features is to find features that are at least partially invariant to lighting, pose, and viewpoint changes. Features that have been used include variations on color histograms [25, 11, 12, 20, 14, 29], local binary patterns [25, 11, 12, 20, 14], Gabor features [14], color names [26], and local patches [28]. The basic idea behind metric learning approaches is to find a mapping from feature space to a new space in which feature vectors from *same* image pairs are closer than feature vectors from *different* image pairs. Metric learning approaches that have been applied to re-identification include Mahalanobis metric learning [12], Locally Adaptive Decision Functions [17], saliency weighted distances [20], Local Fisher Discriminant Analysis [25], Marginal Fisher Analysis [25], and attribute consistent matching [11]. Our approach is to learn a deep network that simultaneously finds an effective set of features and a corresponding similarity function.

2.2. Deep Learning for Re-Identification

To our knowledge, there have been two previous papers that also used a deep learning approach for re-identification: Yi et al. [27] and Li et al. [16]. In [27], a "siamese" convolutional network is presented for metric learning. Their network architecture consists of three independent convolutional networks that act on three overlapping parts of the two input images. Each part-specific network consists of two convolutional layers with max pooling, followed by a fully connected layer. The fully connected layer produces an output vector for each input image, and the two output vectors are compared using a cosine function. The cosine outputs for each of the three parts are then fused to get a final similarity score.

Li et al. [16] use a different network architecture that begins with a single convolutional layer with max pooling, followed by a patch-matching layer that multiplies convolutional feature responses from the two inputs at a variety of horizontal offsets. (The response to each patch in one input image is multiplied separately by the response to every other patch sampled from the same horizontal strip in the other input image.) This is followed by a max-out grouping layer that keeps the largest response from each horizontal strip of patch-match responses, followed by another convolutional layer with max pooling, and finally a fully connected layer and softmax output.

Our architecture differs substantially from these previous approaches. Our network begins with two layers of convolution and max pooling to learn a set of features for comparing the two input images. We then use a novel layer that computes cross-input neighborhood difference features, which compare the features from one input image with the features computed in neighboring locations of the other image. This is followed by a subsequent novel layer that distills these local differences into a smaller patch summary feature. Next, we use another convolutional layer with max pooling, followed by two fully connected layers with softmax output. Along with our new layers which have learnable parameters in them, our network has three convolutional layers as compared to just two in [16] and [27], making our network deeper than previously presented networks for re-identification in the literature. In addition, our network introduces a more powerful way to compare the features learned in the early layers.

Our deep network's re-identification performance exceeds that of all previous approaches on both the large CUHK03 [16] data set and the smaller CUHK01 [15] data set. In addition, even though small data sets can make effective training of large networks difficult or impossible [16], our network performs comparably with the state of the art on the much smaller VIPeR data set.

3. Our Architecture

In this paper, we propose a deep neural network architecture that formulates the problem of person re-identification as binary classification. Given an input pair of images, the task is to determine whether or not the two images represent the same person. Figure 2 illustrates our network's architecture. As briefly described in the previous section, our network consists of the following distinct layers: two layers of tied convolution with max pooling, cross-input neighborhood differences, patch summary features, across-patch features, higher-order relationships, and finally a softmax function to yield the final estimate of whether the input images are of the same person or not. Each of these layers is explained in the following subsections.

3.1. Tied Convolution

To determine whether two input images are of the same person, we need to find relationships between the two views. In the deep learning literature, convolutional features have proven to provide representations that are useful for a variety of classification tasks. The first two layers of our network are convolution layers, which we use to compute higher-order features on each input image separately. In order for the features to be comparable across the two images in later layers, our first two layers perform tied convolution, in which weights are shared across the two views, to ensure that both views use the same filters to compute features. As shown in Figure 2, in the first convolution layer we pass input pairs of RGB images of size $60 \times 160 \times 3$ through 20 learned filters of size $5 \times 5 \times 3$. The resulting feature maps are passed through a max-pooling kernel that halves

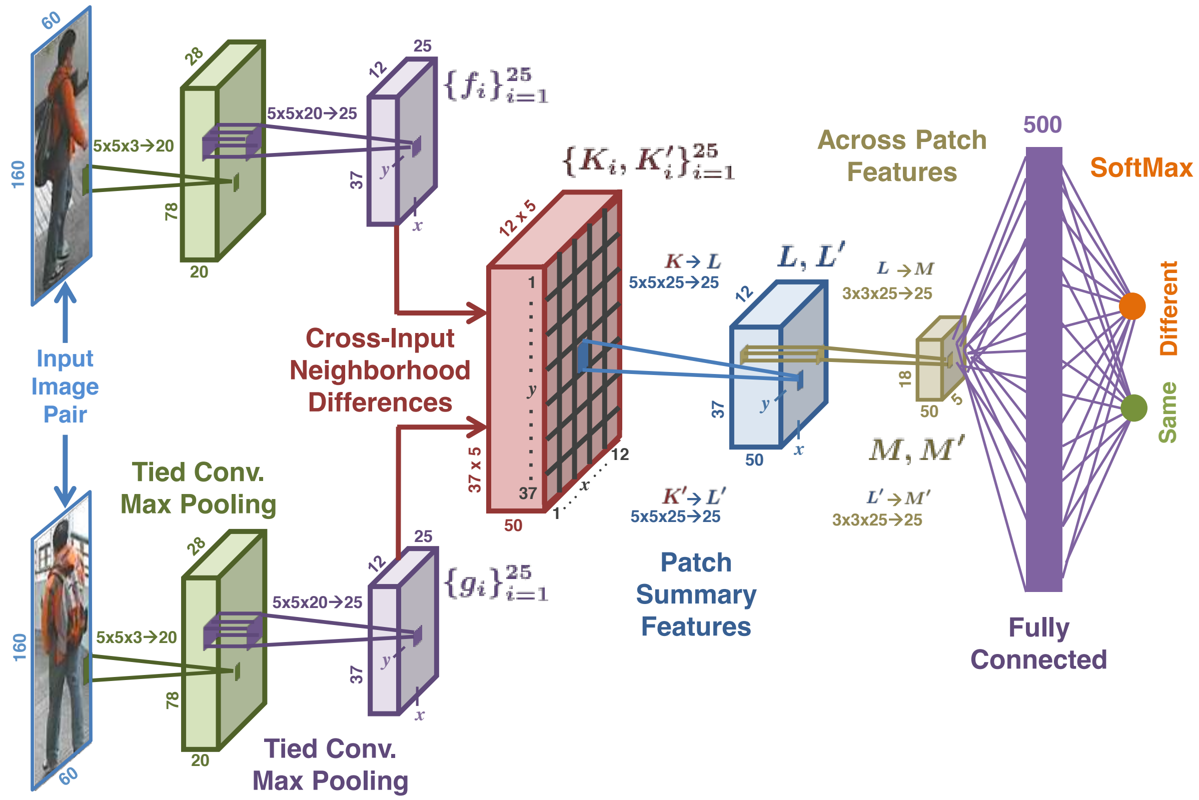


Figure 2: Proposed Architecture. Paired images are passed through the network. While initial layers extract features in the two views individually, higher layers compute relationships between them. The number and size of convolutional filters that must be learned are shown. For example, in the first tied convolution layer, $5 \times 5 \times 3 \to 20$ indicates that there are 20 convolutional features in the layer, each with a kernel size of $5 \times 5 \times 3$. There are 2,308,147 learnable parameters in the whole network. Refer to Section 3 for more details. [Note that all of the figures in this paper are best viewed in color.]

the width and height of features. These features are passed through another tied convolution layer that uses 25 learned filters of size $5 \times 5 \times 20$, followed by a max-pooling layer that again decreases the width and height of the feature map by a factor of 2. At the end of these two feature computation layers, each input image is represented by 25 feature maps of size 12×37.

3.2. Cross-Input Neighborhood Differences

The two tied convolution layers provide a set of 25 feature maps for each input image, from which we can learn relationships between the two views. Let f_i and g_i, respectively, represent the ith feature map ($1 \le i \le 25$) from the first and second views. A *cross-input neighborhood differences* layer computes differences in feature values across the two views around a neighborhood of each feature location, producing a set of 25 neighborhood difference maps K_i. Since $f_i, g_i \in \mathbb{R}^{12 \times 37}$, $K_i \in \mathbb{R}^{12 \times 37 \times 5 \times 5}$, where 5×5 is the size of the square neighborhood. Each K_i is a 12×37 grid of 5×5 blocks, in which the block indexed by (x, y) is denoted $K_i(x, y) \in \mathbb{R}^{5 \times 5}$, where x, y are integers ($1 \le x \le 12$ and $1 \le y \le 37$). More precisely,

$$K_i(x, y) = f_i(x, y)\mathbb{1}(5, 5) - \mathcal{N}[g_i(x, y)] \qquad (1)$$

where

$$\mathbb{1}(5,5) \in \mathbb{R}^{5\times5} \text{ is a } 5 \times 5 \text{ matrix of 1s,}$$
$$\mathcal{N}[g_i(x,y)] \in \mathbb{R}^{5\times5} \text{ is the } 5 \times 5 \text{ neighborhood of } g_i \text{ centered at } (x,y).$$

In words, the 5×5 matrix $K_i(x, y)$ is the difference of two 5×5 matrices, in the first of which every element is a copy of the scalar $f_i(x, y)$, and the second of which is the 5×5 neighborhood of g_i centered at (x, y). The motivation behind taking differences in a neighborhood is to add robustness to positional differences in corresponding features of the two input images. Since the operation in (1) is asymmetric, we also consider the neighborhood difference map K_i', which is defined just like K_i in (1) except that the roles of f_i and g_i are reversed. This yields 50 neighborhood difference maps, $\{K_i\}_{i=1}^{25}$ and $\{K_i'\}_{i=1}^{25}$, each of which has size $12 \times 37 \times 5 \times 5$. We pass these neighborhood difference maps through a rectified linear unit (ReLu).

3.3. Patch Summary Features

In the previous layer, we have computed a rough relationship among features from the two input images in the form of neighborhood difference maps. A *patch summary* layer summarizes these neighborhood difference maps by producing a holistic representation of the differences in each 5×5 block. This layer performs the mapping from $K \in \mathbb{R}^{12\times37\times5\times5\times25} \rightarrow L \in \mathbb{R}^{12\times37\times25}$. This is accomplished by convolving K with 25 filters of size $5 \times 5 \times 25$, with a stride of 5. By exactly matching the stride to the width of the square blocks, we ensure that the 25-dimensional feature vector at location (x, y) of L is computed only from the 25 blocks $K_i(x, y)$, i.e., from the 5×5 grid square (x, y) of each neighborhood difference map K_i (where $1 \leq i \leq 25$). Since these are in turn computed only from the local neighborhood of (x, y) in the feature maps f_i and g_i, the 25-dimensional patch summary feature vector at location (x, y) of L provides a high-level summary of the cross-input differences in the neighborhood of location (x, y). We also compute patch summary features L' from K' in the same way that we computed L from K. Note that filters for the mapping $K \rightarrow L$ and $K' \rightarrow L'$ are different, not tied as in the first two layers of the network. Both L and L' are then passed through a rectified linear unit (ReLu).

3.4. Across-Patch Features

So far we have obtained a high-level representation of differences within a local neighborhood, by computing neighborhood difference maps and then obtaining a high-level local representation of these neighborhood difference maps. In the next layer, we learn spatial relationships across neighborhood differences. This is done by convolving L with 25 filters of size $3 \times 3 \times 25$ with a stride of 1. The resultant features are passed through a max pooling kernel to reduce the height and width by a factor of 2. This yields 25 feature maps of size 5×18, which we denote $M \in \mathbb{R}^{5\times18\times25}$. We similarly obtain across-patch features M' from L'. Filters for the mapping $L \rightarrow M$ and $L' \rightarrow M'$ are not tied.

3.5. Higher-Order Relationships

We apply a fully connected layer after M and M'. This captures higher-order relationships by a) combining information from patches that are far from each other and b) combining information from M with information from M'. The resultant feature vector of size 500 is passed through a ReLu nonlinearity. These 500 outputs are then passed to another fully connected layer containing 2 softmax units, which represent the probability that the two images in the pair are of the same person or different people.

4. Visualization of Features

Figure 3 gives a visualization of feature responses at each layer (L1–L6) of the network. The left and right sides of the figure display responses to a positive (same) and negative (different) input pair, respectively. The response maps labeled L1 for the positive pair show the response of one of the 20 features after the first tied convolution layer (see Section 3.1). This feature responds strongly to bright white regions of the image, highlighting shirt regions of the person in both views. The maps labeled L1 for the negative pair show the response of a different one of the 20 first-layer features. This feature responds strongly to black regions, highlighting the shirt of the person in view 1 and the pants of the person in view 2. The label L2 indicates feature responses after the second tied convolution layer, which show a pair of feature maps f_i and g_i. The L2 feature shown for the positive pair captures tan and skin-color regions, giving higher responses to the legs, hands, and face of the person. Since this is a positive pair, similar parts of the image are highlighted in the two views. In contrast, the L2 feature for the negative pair activates for different portions of the image across the two views: the legs (pink shorts and pinkish skin) of the person in view 1, versus the torso (pink shirt and pinkish arms) of the person in view 2.

The images labeled L3 are responses of a feature from the cross-input neighborhood differences layer (see Section 3.2). Recall from (1) that this layer computes the differences of feature maps across the two views in a neighborhood. The resultant feature difference map is then passed through a ReLu, which clips all negative responses to zero. For a positive pair, ideally the neighborhood difference map should be close to zero. Nonzero values should be small and relatively uniform across the map, mainly because the two feature maps compared are very similar. This is illustrated in the L3 map on the left in the positive pair (one of the K_i maps), which has small but non-zero values distributed throughout the map. The image just to its right, which is its complement K_i', has values that are all zero or close to zero. For the negative pair, different regions are highlighted by f_i than by g_i, so K_i gives a strong response to legs but zeros elsewhere, whereas K_i' responds only to the person's torso. A similar pattern is observed in the patch summary feature for the negative pair (see Section 3.3), labeled L4. Higher-order relations across summarized neighborhood difference maps are captured in L5 (see Section 3.4). Finally, L6 shows features after the first fully connected layer (section 3.5). Notice that this feature representation of a positive pair is quite different than that of a negative pair. This top-layer feature is discriminative and can be used as input to an off-the-shelf classifier.

Figure 4 shows a visualization of the weights learned by the first tied convolution layer. The weights shown were learned on the CUHK03 data set. In addition to capturing

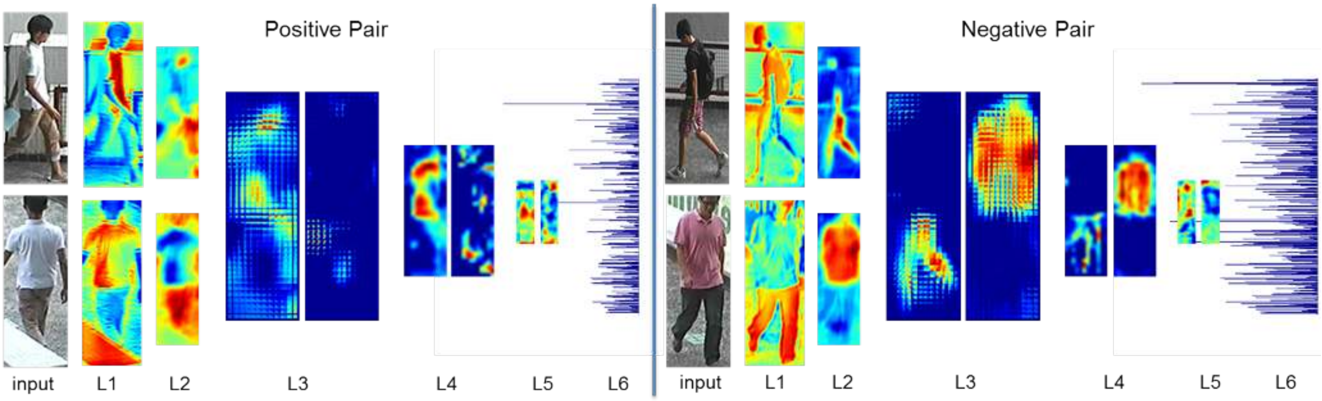


Figure 3: Visualization of features learned by our architecture. Initial layers learn image features that are important to distinguish between a positive and a negative pair. Deeper layers learn relationships across the two views so that classification performance is maximized. For details, see Section 4.

some low-level texture information, several of these learned filters exhibit a strong color specialization.

5. Comparison with Other Deep Architectures

In Figure 5c, we compare our presented network with other variations to gain insights into how much each of our network's novel features contributes to its overall performance. We describe some of these variations here. More details about these architectures can be found in the supplementary material.

Element-wise difference: This architecture illustrates the benefit of comparing with the neighborhood in cross-image comparisons. In this architecture, we perform two layers of tied convolution followed by max pooling. We then compute a cross-input element-wise difference (rather than cross-input neighborhood differences) of the corresponding feature maps. This difference is passed through another layer of convolution followed by a fully connected layer and then softmax.

Disparity-wise convolution: This architecture illustrates the benefit of computing patch summary features. As in our presented network, this architecture performs two tied convolutions followed by max pooling, after which cross-input neighborhood differences are computed. But in this network, the 50 neighborhood difference maps of size $\mathbb{R}^{12\times37\times5\times5}$, are rearranged to give 25 groups of 50 feature maps, where each feature map has size $\mathbb{R}^{12\times37}$. A convolution is then applied to each of these groups. This is then passed through a fully connected layer and then softmax. Rather than explicitly summarizing neighborhood differences, this architecture instead directly learns across-patch relationships.

Four-layer convnet: This architecture illustrates the benefit of having a total of four convolutional layers, rather than two as in previous deep approaches to re-identification. We implemented a siamese type network similar to [27], but built the network with 4 layers of convolution rather than 2.

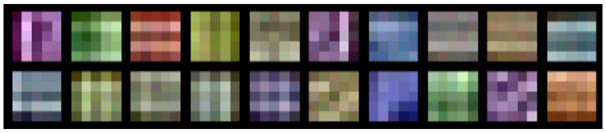


Figure 4: Visualization of the weights learned in the first tied convolution layer. Each filter has size $5 \times 5 \times 3$.

FPNN: We also created our own implementation of FPNN [16] to facilitate comparisons with their results.

6. Training the Network

We pose the re-identification problem as binary classification. Training data consist of image pairs labeled as positive (same) and negative (different). The optimization objective is average loss over all pairs in the data set. As the data set can be quite large, in practice we use a stochastic approximation of this objective. Training data are randomly divided into mini-batches. The model performs forward propagation on the current mini-batch and computes the output and loss. Backpropagation is then used to compute the gradients on this batch, and network weights are updated. We perform stochastic gradient descent [3] to perform weight updates. We start with a base learning rate of $\eta^{(0)} = 0.01$ and gradually decrease it as the training progresses using an inverse policy: $\eta^{(i)} = \eta^{(0)}(1 + \gamma \cdot i)^{-p}$ where $\gamma = 10^{-4}$, $p = 0.75$, and i is the current mini-batch iteration. We use a momentum of $\mu = 0.9$ and weight decay $\lambda = 5 \times 10^{-4}$. With more passes over the training data, the model improves until it converges. We use a validation set to evaluate intermediate models and select the one that has maximum performance. See the supplementary material for performance on the validation set as a function of mini-batch iterations.

6.1. Data Augmentation

There are not nearly as many positive pairs as negative pairs, which can lead to data imbalance and overfitting. To reduce overfitting, we artificially enlarge the data set using label-preserving transformations [13]. We augment the data by performing random 2D translation, as also done in [16]. For an original image of size $W \times H$, we sample 5 images around the image center, with translation drawn from a uniform distribution in the range $[-0.05H, 0.05H] \times [-0.05W, 0.05W]$. For the smallest data set (see Section 7.3), we also horizontally reflect each image.

6.2. Hard Negative Mining

Data augmentation increases the number of positive pairs, but the training data set is still imbalanced with many more negatives than positives. If we trained the network with this imbalanced data set, it would learn to predict every pair as negative. Therefore, we randomly downsample the negative set to get just twice as many negatives as positives (after augmentation), then train the network. The converged model thus obtained is not optimal since it has not seen all possible negatives. We use the current model to classify all of the negative pairs, and identify negatives on which the network performs worst. We retrain the fully connected (top) layer of the network using a set containing as many of these difficult negative pairs as positive pairs[1].

6.3. Fine-tuning

For small data sets that contain too few positives for effective training, we initialize the model by training on a large data set. After hard negative mining on the large set, the parameters of the converged model are then adapted on the new, small data set. For this new network learning, we begin stochastic gradient descent with learning rate $\eta^{(0)} = 0.001$ (which is $1/10$th the initial pre-training rate).

7. Experiments

We implemented our architecture using the Caffe [10] deep learning framework, adapting various layers from the framework and writing our own layers that are specific to our architecture. Network training converges in roughly 12–14 hours on NVIDIA GTX780 and NVIDIA K40 GPUs.

We present a comprehensive evaluation of our approach by comparing it to the state-of-the-art methods on various data sets. The experiments are conducted with five random splits, and all of the Cumulative Matching Characteristics (CMC) curves are single-shot results. We first report results on the largest re-identification data set in the literature, CUHK03 [16]. We then report results on the CUHK01 data set [15], using two distinct settings: a) 100 identities in the test set, as reported in [16], and b) 486 identities in the test set, as reported in most previous work on the CUHK01 data set. We also report results on the VIPeR data set [8]. VIPeR and the 486-identities setting of CUHK01 are small data sets, making it difficult for deep networks to learn their parameters without overfitting. Because of this, [16] does not report results on these two data sets.

7.1. Experiments on CUHK03

The CUHK03 data set contains 13,164 images of 1,360 pedestrians, captured by six surveillance cameras. Each identity is observed by two disjoint camera views. On average, there are 4.8 images per identity in each view. This data set provides both manually labeled pedestrian bounding boxes and bounding boxes automatically obtained by running a pedestrian detector [6]. We report results on both of these versions of the data (labeled and detected).

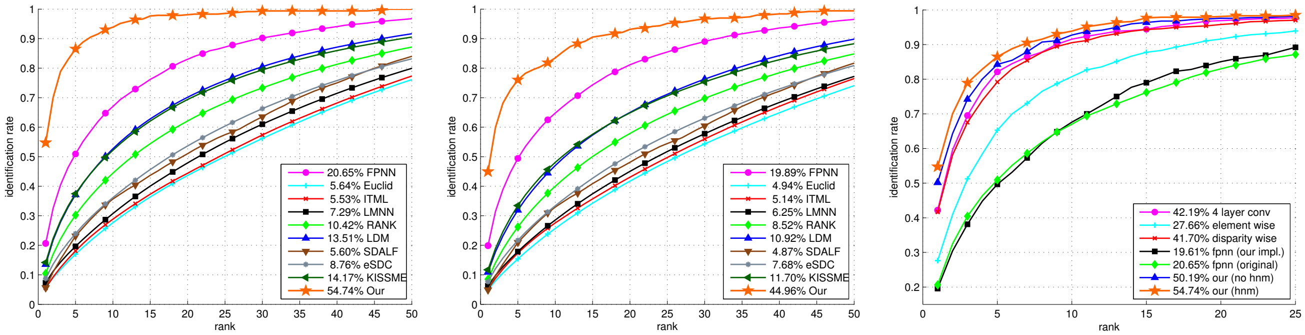
Following the protocol used in [16], we randomly divide 1360 identities into non-overlapping train (1160), test (100), and validation (100) sets. This yields about 26,000 positive pairs before data augmentation. We use a mini-batch size of 150 samples and train the network for 210,000 iterations. We use the validation set to design the network architecture.

We compare our method against KISSME [12], eSDC [30], SDALF [5], ITML [4], logistic distance metric learning (LDM) [9], largest margin nearest neighbor (LMNN) [24], metric learning to rank (RANK) [21], and directly using Euclidean distance to compare features. When using metric learning methods and Euclidean distance, dense color histograms and dense SIFT are used [30]. We also compare against the deep network FPNN [16], which is the current state of the art on this data set.

Figure 5a plots the CMC curves of all these methods on the CUHK03 labeled image data set. We outperform the previous deep learning method, FPNN, by a large margin. Our rank-1 accuracy is more than double that of the previous state of the art (54.74% vs. 20.65%). Figure 5b plots performance on the CUHK03 detected image data set. Although the performance of our method on CUHK03-detected is less than on CUHK03-labeled, mainly due to misalignment caused by the detector, our method still greatly outperforms the state of the art (44.96% vs. 19.89%). Figure 1 shows some true positive, false positive, and true negative example results of our system. More qualitative results can be found in the supplementary material.

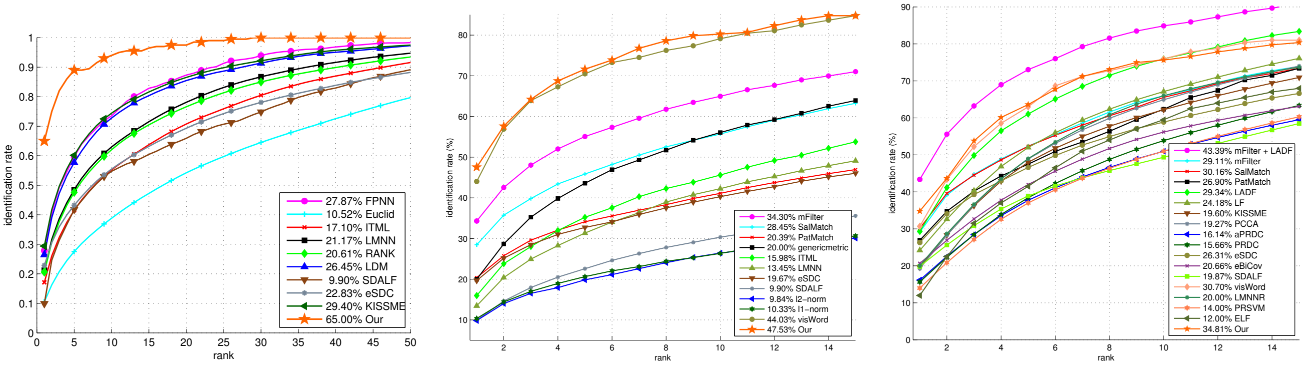
We also implemented a variety of other deep network architectures, explained in Section 5, to illustrate the benefits of various features of our architecture. We compare with these methods in Figure 5c. The top two performing

[1] We also tried retraining the entire network, but retraining just the top layer was more effective.



(a) (b) (c)

Figure 5: CMC curves on CUHK03 data set: (a) and (b) compare our method with previous methods on CUHK03 labeled and detected, respectively. Rank-1 identification rates are shown in the legend next to the method name. Our method beats the state of the art by a large margin. c) Comparison of our method with our own variations of deep architectures on CUHK03 labeled. Out of the shown methods, only FPNN is previously mentioned in the literature. See section 7.1 for details.



(a) (b) (c)

Figure 6: CMC curves on CUHK01 and VIPeR data sets: a) CUHK01 data set with 100 test IDs: Our method outperforms the state of the art by more than a factor of 2. b) CUHK01 data set with 486 test IDs: Our method outperforms all previous methods on this data set with this protocol, as well. c) VIPeR: Our method beats all previous methods individually, although a combination of mFilter + LADF performs better than us. Note that (b) and (c) are especially challenging for deep learning methods since there are very few positive pairs. See Sections 7.2 and 7.3 for more details

methods are our architecture with and without hard negative mining (HNM). Note that other than FPNN, none of the methods in Figure 5c has been previously discussed in the literature.

7.2. Experiments on CUHK01

The CUHK01 data set has 971 identities, with 2 images per person in each view. We report results for two different settings of this data set: 100 test IDs, and 486 test IDs.

a) **100 test IDs:** In this setting, 100 identities are used for testing, with the remaining 871 identities used for training and validation. This protocol is better suited for deep learning because it uses 90% of the data for training. FPNN [16] uses this setting on this data set. Figure 6a compares the performance of our network with previous methods. Our method outperforms the state of the art in this setting by a wide margin, with a rank-1 recognition rate of 65% (vs. 29.40% by the next best method). Notice that the second-best method on this data set is KISSME, and not the deep network FPNN. This can be attributed to a decrease in training data as compared to CUHK03, causing FPNN to overfit. In contrast, our method is able to generalize even with this smaller data set.

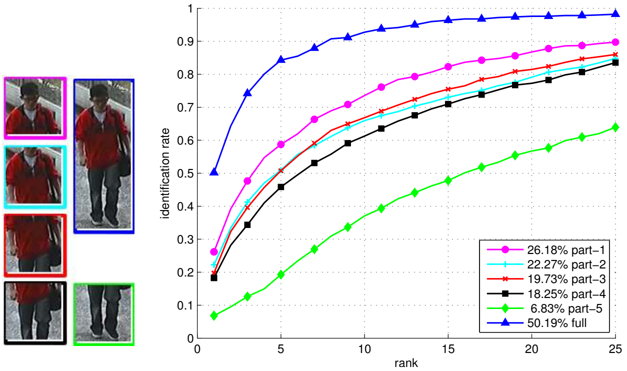
b) **486 test IDs:** Most previous papers report results on the CUHK01 data set by considering 486 identities for testing. We compare our approach against mid-level filters (mFilter) [31], saliency matching (SalMatch) [29], patch matching (PatMatch) [29], generic metric [15], ITML [4], LMNN [24], eSDC [30], SDALF [5], l2-norm, l1-norm [31], and co-occurrence model using visual word (visWord) [28]. With 486 identities in the test set, only 485 identities are left for training. This leaves only 1940 positive samples for training, which makes it practically impossible for a deep architecture of reasonable size not to overfit if trained from scratch on this data. One way to solve this problem is to use a model trained on CUHK03, then test on the 486 identities of CUHK01. This is unlikely to work well since the network does not know the statistics of the test data set, and in fact, our model trained on CUHK03 and tested on CUHK01 gave rank-1 accuracy of around 6%, which is far below the state of the art. Instead, we pre-train a network on CUHK03 and adapt it for CUHK01 by fine-tuning (see Section 6.3) it on CUHK01 with 485 training identities (non-overlapping with the test set). The performance of the network after fine-tuning for $210K$ iterations increases dramatically, to a rank-1 accuracy of 40.5%. Using this model, we search for hard negatives and use them to retrain the top layer of the network (see Section 6.2). After $210K$ iterations, we achieve a rank-1 accuracy of 47.5%, beating the state of the art. See Figure 6b for a comparison with other methods.

7.3. Experiments on VIPeR

The VIPeR data set contains 632 pedestrian pairs in two views, with only one image per person in each view. The testing protocol is to split the data set into half, 316 for training and 316 for testing. In addition to the methods listed in section 7.2 and 7.1, we compare our method against local Fisher discriminant analysis (LF) [23], PCCA [22], aPRDC [18], PRDC [32], eBiCov [19], LMNNR [1], PRSVM [2], and ELF [7]. This data set is extremely challenging for deep network architectures for two reasons: a) there are only 316 identities for training with 1 image per person in each view, giving a total of just 316 positives, and b) the resolution of the images is lower (48 × 128 as compared to 60 × 160 for CUHK01). We train a model using the CUHK03 and CUHK01 data sets, then adapt the trained model to the VIPeR data set by fine-tuning on 316 training identities. Since the number of negatives is small for this data set ($90K$), hard negative mining does not improve results after fine-tuning because most of the negatives were already used during fine-tuning. Figure 6c compares performance of our approach with other methods. Our method obtains 34.81% rank-1 accuracy, beating all other methods individually, although a combination of two approaches (mFilter [31] + LADF [17]) performs better than ours with a rank-1 accuracy of 43.4% as reported in [31]. The deep-metric-learning-based method [27] also reports results on the VIPeR data set, with a lower rank-1 accuracy of 28.2%.

7.4. Analysis of different body parts

To understand the contribution of different body regions to identification, we trained 5 different networks on different body parts, as shown in Figure 7a. The experiment was performed on the CUHK03 labeled data set, and the performance of each part is shown in Figure 7b. The part that performs best is part 1: the upper region of the body including the face. As we move down the body, the performance decreases, with legs capturing the least discriminative information. This experiment suggests a direction for future work in which different models can be trained for different parts of the body, and the scores from different part pairs can then be accumulated to reach a final decision. Such a system may be helpful in handling severe occlusions and to identify people in images that have been taken across time (e.g., sitting in one view and standing in the other).



(a) (b)

Figure 7: Analysis of different body parts: a) Left column shows parts 1 to 4 (from top to bottom). Right column shows full pedestrian image and part 5. b) Shows performance of different parts on the CUHK03 data set. Refer to Section 7.4 for more details.

8. Conclusion

We have presented a novel deep architecture for person re-identification. We have designed two novel layers for capturing relationships between two views: a cross-input neighborhood differences layer, and a subsequent layer that summarizes these differences. We demonstrate the effectiveness of our method by performing a comprehensive evaluation of our approach on various data sets. On the large CUHK03 data set, our method outperforms the state of the art by a huge margin. On the smaller CUHK01 data set (100 test IDs setting), whereas other deep methods overfit [16], our method is able to generalize and produce state-of-the-art results. We also show that models learned by our method on a large data set can be adapted to new, smaller data sets. We demonstrate this by evaluating our method on two small data sets. On CUHK01 (486 test IDs setting), we outperform all previous methods, and on VIPeR, our results are comparable to the state of the art.

References

[1] S. Bak, E. Corvee, F. Bremond, and M. Thonnat. Multiple-shot human re-identification by mean riemannian covariance grid. In *Proceedings of the 2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance*, AVSS '11, pages 179–184, Washington, DC, USA, 2011. IEEE Computer Society. 8

[2] L. Bazzani, M. Cristani, A. Perina, and V. Murino. Multiple-shot person re-identification by chromatic and epitomic analyses. *Pattern Recognition Letters*, 33(7):898–903, 2012. 8

[3] L. Bottou. Stochastic gradient tricks. In G. Montavon, G. B. Orr, and K.-R. Müller, editors, *Neural Networks, Tricks of the Trade, Reloaded*, Lecture Notes in Computer Science (LNCS 7700), pages 430–445. Springer, 2012. 5

[4] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 209–216, New York, NY, USA, 2007. ACM. 6, 7

[5] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani. Person re-identification by symmetry-driven accumulation of local features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2360 –2367, June 2010. 6, 7

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, Sept. 2010. 6

[7] N. Gheissari, T. B. Sebastian, and R. Hartley. Person reidentification using spatiotemporal appearance. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 1528–1535, Washington, DC, USA, 2006. IEEE Computer Society. 8

[8] D. Gray, S. Brennan, and H. Tao. Evaluating appearance models for recognition, reacquisition, and tracking. In *In IEEE International Workshop on Performance Evaluation for Tracking and Surveillance, Rio de Janeiro*, 2007. 6

[9] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *ICCV*, Kyoto, Japan, Sept. 2009. 6

[10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 6

[11] S. Khamis, C. Kuo, V. Singh, V. Shet, and L. Davis. Joint learning for attribute-consistent person re-identification. In *ECCV Workshop on Visual Surveillance and Re-identification*, 2014. 2

[12] M. Koestinger, M. Hirzer, P. Wohlhart, P. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *CVPR*, 2012. 2, 6

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 6

[14] W. Li and X. Wang. Locally aligned feature transforms across views. In *CVPR*, 2013. 2

[15] W. Li, R. Zhao, and X. Wang. Human re-identification with transferred metric learning. In *ACCV*, 2012. 2, 6, 7

[16] W. Li, R. Zhao, T. Xiao, and X. Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *CVPR*, 2014. 2, 5, 6, 7, 8

[17] Z. Li, S. Chang, F. Liang, T. Huang, L. Cao, and J. Smith. Learning locally-adaptive decision functions for person verification. In *CVPR*, 2013. 2, 8

[18] C. Liu, S. Gong, C. C. Loy, and X. Lin. Person re-identification: What features are important? In A. Fusiello, V. Murino, and R. Cucchiara, editors, *ECCV Workshops (1)*, volume 7583 of *Lecture Notes in Computer Science*, pages 391–401. Springer, 2012. 8

[19] B. Ma, Y. Su, and F. Jurie. Bicov: a novel image representation for person re-identification and face verification. In *BMVC'12*, pages 1–11, 2012. 8

[20] N. Martinel, C. Micheloni, and G. Feresti. Saliency weighted features for person re-identification. In *ECCV Workshop on Visual Surveillance and Re-identification*, 2014. 2

[21] B. Mcfee and G. Lanckriet. Metric learning to rank. In *In Proceedings of the 27th annual International Conference on Machine Learning (ICML)*, 2010. 6

[22] A. Mignon and F. Jurie. Pcca: A new approach for distance learning from sparse pairwise constraints. In *CVPR*, pages 2666–2672. IEEE, 2012. 8

[23] S. Pedagadi, J. Orwell, S. Velastin, and B. Boghossian. Local fisher discriminant analysis for pedestrian re-identification. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:3318–3325, 2013. 8

[24] K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009. 6, 7

[25] F. Xiong, M. Gou, O. Camps, and M. Sznaier. Person re-identification using kernel-based metric learning methods. In *ECCV*, 2014. 2

[26] Y. Yang, J. Yang, J. Yan, S. Liao, D. Yi, and S. Li. Salient color names for person re-identification. In *ECCV*, 2014. 2

[27] D. Yi, Z. Lei, and S. Z. Li. Deep metric learning for practical person re-identification. *ICPR*, 2014. 2, 5, 8

[28] Z. Zhang, Y. Chen, and V. Saligrama. A novel visual word co-occurrence model for person re-identification. In *ECCV Workshop on Visual Surveillance and Re-identification*, 2014. 2, 7

[29] R. Zhao, W. Ouyang, and X. Wang. Person re-identification by salience matching. In *ICCV*, 2013. 2, 7

[30] R. Zhao, W. Ouyang, and X. Wang. Unsupervised salience learning for person re-identification. In *CVPR*, 2013. 6, 7

[31] R. Zhao, W. Ouyang, and X. Wang. Learning mid-level filters for person re-identification. In *CVPR*, 2014. 2, 7, 8

[32] W.-S. Zheng, S. Gong, and T. Xiang. Person re-identification by probabilistic relative distance comparison. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 649–656, Washington, DC, USA, 2011. IEEE Computer Society. 8