Recurrent Neural Networks for Person Re-identification Revisited

Jean-Baptiste Boin
Stanford University
Stanford, CA, U.S.A.
jbboin@stanford.edu

André Araujo
Google AI
Mountain View, CA, U.S.A.
andrearaujo@google.com

Bernd Girod
Stanford University
Stanford, CA, U.S.A.
bgirod@stanford.edu

Abstract

The task of person re-identification has recently received rising attention due to the high performance achieved by new methods based on deep learning. In particular, in the context of video-based re-identification, many state-of-the-art works have explored the use of Recurrent Neural Networks (RNNs) to process input sequences. In this work, we revisit this tool by deriving an approximation which reveals the small effect of recurrent connections, leading to a much simpler feed-forward architecture. Using the same parameters as the recurrent version, our proposed feed-forward architecture obtains very similar accuracy. More importantly, our model can be combined with a new training process to significantly improve re-identification performance. Our experiments demonstrate that the proposed models converge substantially faster than recurrent ones, with accuracy improvements by up to 5% on two datasets. The performance achieved is better or on par with other RNN-based person re-identification techniques.

1. Introduction

Person re-identification consists of associating different tracks of a person as they are captured across a scene by different cameras, which is useful for video surveillance or crowd dynamics understanding. The challenges inherent to this task are the variations in background, body pose, illumination and viewpoint. It is important to represent a person using a descriptor that is as robust as possible to these variations, while still being discriminative enough to be characteristic of a single person's identity. A sub-class of this problem is *video-based* re-identification, where the goal is to match a video of a person against a gallery of videos captured by different cameras, by opposition to image-based (or single-shot) re-identification.

Person re-identification has recently received rising attention due to the much improved performance achieved by methods based on deep learning. For video-based re-identification, it has been shown that representing videos by aggregating visual information across the temporal dimension was particularly effective. Recurrent Neural Networks (RNNs) have shown promising results for performing this aggregation in multiple independent works [12, 18, 15, 25, 20, 17, 2]. In this paper, we analyze one type of architecture that uses RNNs for video representation. Our contributions are the following. We show that the recurrent network architecture can be replaced with a simpler non-recurrent architecture, without sacrificing the performance. Not only does this lower the complexity of the forward pass through the network, making the feature extraction easier to parallelize, but we also show that this model can be trained with an improved process that boosts the final performance while converging substantially faster. Finally, we obtain results that are on par or better than other published work based on RNNs, but with a much simpler technique.

2. Related work

The majority of the traditional approaches to image-based person re-identification follow a two-step strategy. The first step is feature representation, which aims at representing an input in a way that is as robust as possible to variations in illumination, pose and viewpoint. Common techniques include: SIFT used in [21], SILTP used in [10], Local Binary Patterns used in [16, 18], color histograms used in [10, 16, 21, 18]. This step is followed by metric learning, which transforms the features in a way that maximizes intra-class similarities while minimizing inter-class similarities. Some metric learning algorithms specifically introduced for person re-identification are XQDA [10], LFDA [13], and its kernelized version k-LFDA [16]. See [24] for a more detailed survey of these techniques.

More recently, with the successes of deep learning in computer vision as well as the release of larger datasets for re-identification (VIPeR [5], CUHK03 [9], Market-1501 [23]), this field has shifted more and more towards neural networks. In particular, the Siamese network architecture provides a straightforward way to simultaneously tackle the tasks of feature extraction and metric learning into a unified end-to-end system. This architecture was introduced to the field of person re-identification by the pioneering works of [19] and [9]. This powerful tool can learn an embedding where inputs corresponding to the same class (or identity) are closer to each other than inputs corresponding to different classes. Unlike classification approaches it has the added benefit that it can be used even if a low number of images is available per class (such as a single pair of images). Variants of the Siamese network have been used for re-identification. [1] achieved very good results by complementing the Siamese architecture with a layer that computes neighborhood differences across the inputs. Instead of using pairs of images as inputs, [3] uses triplets of images whose representations are optimized by using triplet loss.

Although less explored, video-based re-identification has followed a similar path since many techniques from image-based re-identification are applicable, ranging from low-level hand-crafted features [11, 14] to deep learning, made possible by the release of large datasets (PRID2011 [7], iLIDS-VID [14], MARS [22]). In order to represent a video sequence, most works consider some form of pooling that aggregates frame features into a single vector representing the video. Some approaches such as [22] do not explicitly make use of the temporal information, but other works have shown promising results when learning spatio-temporal features. In particular, [12, 18, 15] all propose to use Recurrent Neural Networks (RNNs) to aggregate the temporal information across the duration of the video. [25] showed promising results by combining a RNN-based temporal attention model with a spatial attention model.

In this work, we will focus on [12], which directly inspired more recent papers that built upon it: [20] replaces the RNN with a bi-directional RNN; [17] computes the frame-level features with an extra spatial pyramid pooling layer to generate a multi-scale spatial representation, and aggregates these features with a RNN and a more complex attentive temporal pooling algorithm; [2] aggregates the features at the output of the RNN with the frame-level features used as the input to the RNN, and also processes the upper-body, lower-body and full-body sequences separately, with late fusion of the three sequence descriptors. All propose a more complex system compared to [12], but showed improved performance.

3. Proposed framework

3.1. General architecture of the network

In video-based person re-identification, a query video of a person is matched against a gallery of videos, either by using a multi-match strategy where frame descriptors from query and gallery video are compared pairwise, or by a single-match strategy where the information is first pooled across frames to represent each video with a single descriptor. The latter strategies have slowly superseded the former ones and have proved more successful and efficient.

In order to extract a fixed length one-dimensional descriptor from a variable-length sequence of images, we introduce a parametric model called the feature extraction network. The general architecture of that network is shown in Fig. 1: it is made up of three distinct stages namely frame feature extraction, sequence processing and temporal pooling. This multiple-stage architecture is considered because is a good generalization of the systems used in the related works [12] (as well as its extensions [20, 17, 2]), [18] and [15] for video representation.

For each frame of the input video, the first stage (frame feature extraction) independently extracts a descriptor of dimension d_1 that should capture the appearance information to be aggregated across the sequence. The second stage (sequence processing) takes the sequence of frame descriptors and outputs a (different) sequence of fixed-length vectors of dimension d_2 (it is possible that $d_2 \neq d_1$). In general, this stage mixes the information contained in all the frames of the sequence. One example would be to compute the pairwise differences between consecutive frame descriptors, but this stage can perform more complex operations. Last, the temporal pooling stage outputs a single fixed-length vector from the variable-length sequence at the output of the previous stage. Note that in the general formulation, this temporal pooling could depend on the order of the input sequence, but in general much simpler strategies are used: [12] only considers max and average pooling, showing that average pooling generally outperforms max pooling, and [18] and [15] both use average pooling only. Other works explored more complex temporal pooling strategies [17, 25].

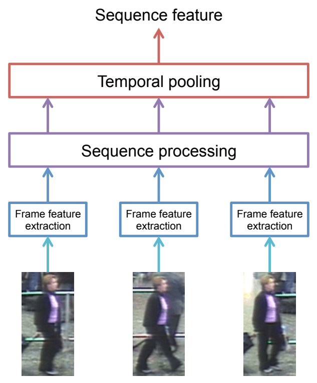

Figure 1. General architecture of the feature extraction network.

Here we will only focus on the specific architecture used in [12], which has shown great success. The frame feature extraction stage is a convolutional neural network (CNN), the sequence processing stage is a recurrent neural network (RNN) and the temporal pooling is performed by average pooling the outputs at each time step. The full network is trained using the Siamese architecture framework.

We call $f^{(t)} \in \mathbf{R}^{d_1}$ ($o^{(t)} \in \mathbf{R}^{d_2}$) the inputs (outputs) of the sequence processing stage, $t = 1, ..., T$. The output at each time step is given by the RNN equations:

$$o^{(t)} = W_i f^{(t)} + W_s r^{(t-1)} \quad (1)$$

$r^{(t-1)} = Tanh(o^{(t-1)})$ is obtained from the previous output.

The last output of the RNN technically contains information from all the time steps of the input sequence so it could be used to represent that sequence (the temporal pooling stage would then just ignore $o^{(t)}$ for $t \leq T-1$ and directly output $o^{(T)}$). However, in practice we observe that the contribution of a given input to each of the time steps of an output of a RNN decreases over time, as the information of the earlier time steps gets diluted and the later time steps dominate. This means that the value of $o^{(T)}$ is much more strongly dependent on $f^{(T)}$ than on $f^{(1)}$. This is not suitable when designing a sequence descriptor that should be as representative of the start of the sequence as of its end. In order to prevent this phenomenon, [12] shows that a good choice is to use average pooling as a temporal pooling stage. A sequence can thus be represented as:

$$v_s = \frac{1}{T} \sum_{t=1}^{T} o^{(t)} \quad (2)$$

3.2. Proposed feed-forward approximation

This limitation of RNNs regarding long-term dependencies has been widely studied and is related to the problem of vanishing gradients. Motivated by this phenomenon, we propose to approximate the RNN to consider only a one step dependency. Under this approximation, the effect of inputs $f^{(1)}, ..., f^{(t-2)}$ on $o^{(t)}$ is in general negligible for a trained RNN. In other words, $o^{(t)}$ may only depend on the previous and the current time steps: $f^{(t-1)}$ and $f^{(t)}$. Hence, (1) can then be rewritten as:

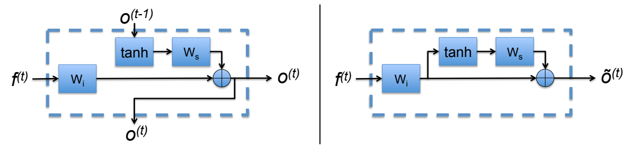

Figure 2. Architecture of the sequence processing stage. Left: Recurrent architecture used as baseline (RNN). Right: Our non-recurrent architecture (FNN). The blocks labeled W_i and W_s correspond to fully connected layers.

$$\begin{aligned} o^{(t)} &= W_i f^{(t)} + W_s Tanh\left(o^{(t-1)}\right) \\ &= W_i f^{(t)} + W_s Tanh\left(W_i f^{(t-1)} + W_s r^{(t-2)}\right) \\ &\approx W_i f^{(t)} + W_s Tanh\left(W_i f^{(t-1)}\right) \end{aligned}$$

The appearance descriptor of the sequence from (2) can now be approximated as

$$\begin{aligned} v_s &\approx \frac{1}{T}\sum_{t=1}^{T}\left(W_i f^{(t)} + W_s Tanh\left(W_i f^{(t-1)}\right)\right) \\ &= \frac{1}{T}\sum_{t=1}^{T} W_i f^{(t)} + \frac{1}{T}\sum_{t=0}^{T-1} W_s Tanh\left(W_i f^{(t)}\right) \\ &= \frac{1}{T}\sum_{t=1}^{T}\left(W_i f^{(t)} + W_s Tanh\left(W_i f^{(t)}\right)\right) - \frac{1}{T} W_s Tanh\left(W_i f^{(T)}\right) \end{aligned}$$

For large enough values of T, the contribution of the last term is small compared to the sum, so it can be neglected. We now introduce $\tilde{o}^{(t)} = W_i f^{(t)} + W_s Tanh(W_i f^{(t)})$, which only depends on $f^{(t)}$. Under the assumptions that were made, we can rewrite

$$v_s \approx \frac{1}{T}\sum_{t=1}^{T} \tilde{o}^{(t)} \tag{3}$$

In other words, the sequence processing stage can be approximated by a simpler non-recurrent network if the RNN is swapped with the feed-forward network shown in Fig. 2. From now on, this architecture of the sequence processing stage will be referred to as FNN (feed-forward neural network). It is important to note that both architectures have the same number of parameters, so their memory footprint is the same.

In order to keep the equations simple the bias terms were ommitted to improve clarity, but a very similar approximation can be derived when they are taken into consideration using the full equation:

$$o^{(t)} = W_i f^{(t)} + b_i + W_s Tanh\left(o^{(t-1)}\right) + b_s \tag{4}$$

In this case, we still ignore the recurrent part when substituting $o^{(t-1)}$ with its expression

$$o^{(t)} \approx W_i f^{(t)} + b_i + W_s Tanh\left(W_i f^{(t-1)} + b_i\right) + b_s$$

and we then perform the second approximation which again gives us the equation (3) with $\tilde{o}^{(t)} = W_i f^{(t)} + b_i + W_s Tanh\left(W_i f^{(t)} + b_i\right) + b_s$.

It is interesting to notice that our proposed FNN architecture contains a shortcut connection, reminiscent of a ResNet block, as was introduced in [6]. In fact, preliminary experiments showed that this residual connection is critical to the success of the training of this network and removing it causes a considerable drop in the re-identification performance.

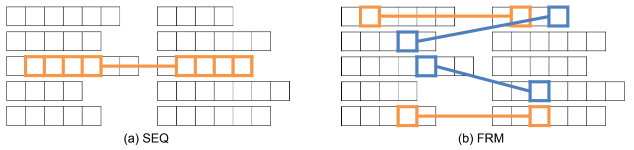

Figure 3. Illustration of the sampling strategy for the two training modes: SEQ (sequence as an input) and FRM (individual frames as an input). The dataset is represented in black: each row represents the two videos available for a given person. We consider a mini-batch with the settings $B = 1$ and $L = 4$. The frames loaded in the mini-batch are the ones with the colored outline. The linked sequences or frames correspond to the pairs used as inputs to the Siamese network. Positive pairs (same identity) are shown in orange; negative pairs (different identity) in blue. SEQ only samples adjacent frames from a maximum of $2B = 2$ distinct sequences while FRM can sample unconstrained frames from up to $2BL = 8$ sequences.

3.3. Improved training pipeline

Compared to the RNN, our proposed FNN architecture keeps the same inputs and outputs, so it can be trained just like the RNN by using a Siamese network. The loss that is used is unchanged (combination of a contrastive loss for a pair of inputs and of an identification loss for each input of the pair).

It is impossible to train an RNN with input sequences of length 1. However, by removing the time dependency, our reformulation of the sequence processing stage as a FNN removes this constraint and allows us to process individual frames as sequences of length 1. Computing the representation of a sequence of length L requires about the same amount of computation time and memory as computing the representation of L individual frames.

We denote B the size of our mini-batch when training the RNN and L the length of the sequences used for training. When training in sequence mode (noted SEQ), within a mini-batch, it is required to load $2BL$ images (the factor 2 is due to the Siamese architecture) from up to $2B$ distinct sequences. If individual frames are used as inputs for training instead (frame mode, noted FRM), it is possible to load and process BL pairs of images with roughly the same memory requirement. This means that a FRM mini-batch is much more diverse since it can now include images from $2BL$ distinct sequences. The two modes are illustrated in Fig. 3.

4. Experiments

4.1. Data and experimental protocol

Datasets. We compare the network architectures and training modes on two datasets: the PRID2011 [7] and the iLIDS-VID [14] datasets. Both datasets were created from videos of pedestrians observed in two non-overlapping camera views, and the matching identities on both views are provided. There are 200 such identities (pairs of video sequences) in the PRID2011 dataset, and 300 in the iLIDS-VID dataset. Note that the PRID2011 dataset contains additional distractor videos of people appearing in only one of the views. We follow the standard testing protocol for this dataset as used in [14] and ignore these distractor videos. Both datasets have comparable video sequence lengths: they range from 5 to 675 with an average of 100 frames

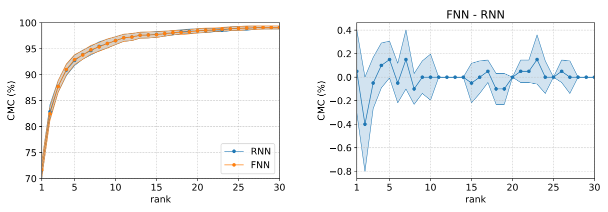

Figure 4. Performance of networks with two different architectures for the sequence processing stage (RNN and FNN) using the same weights. The weights are trained using the RNN architecture. Results are presented on the PRID2011 dataset.

for the PRID2011 dataset, and from 22 to 192 with an average of 71 frames for the iLIDS-VID dataset. We follow the usual testing protocol for these datasets [14]: we randomly split the data into training and test set so that each set contains a distinct half of the identities, and we repeat the training and evaluation for 20 trials.

Raw input data. The data used for all experiments is the color and optical flow data (horizontal and vertical), so in practice each frame in a video is represented as a 5-channel image. The optical flow was extracted as described in [12].

Evaluation metric. As is common practice for these datasets, the reported metric is the Cumulated Matching Characteristics (CMC) curve, averaged over all trials.

Dropout. In the RNN architecture introduced by [12], two dropout layers (with dropout probability of 0.6) were inserted before the input of the first and second fully connected layers W_i and W_s pictured in Fig. 2. In order to keep the FNN architecture as similar as possible, dropout layers with the same probability are inserted at these locations.

4.2. Influence of the recurrent connection

In this section, we investigate the validity of the simplifying assumptions made in Section 3.2, by showing that the approximated descriptors defined in equation (3) yield a similar performance as the original descriptors.

For this experiment, we first train a network with our baseline architecture (where the sequence processing stage is a RNN) and we substitute the RNN to use a FNN architecture instead, without changing the values of the weights. This is possible because there is a natural mapping between the parameters of the two architectures (see Fig. 2).

The performance of the two architectures can be compared by evaluating them on the same test set. We show the results on the PRID2011 dataset across 20 trials in Fig. 4. The first plot shows the average CMC value (along with 95% confidence intervals) for both RNN and FNN architectures while the second plot shows the distribution of the difference between the CMC values for both architectures.

The average CMC curves for both architectures are very similar, and the difference in the CMC values at each trial is usually very close to 0, with 0 being included in the 95% confidence intervals for all rank values. If the same weights are used, there is no value of the rank where one architecture consistently outperforms the other one in a statistically significant manner. This means that, out of the box, *removing the recurrent connection does not have any measurable effect on the performance.*

This is a critical result because it challenges the assumptions that are typically made regarding the reasons why a recurrent architecture gives improved performance. It is commonly assumed that the structure of the RNN allows for non-trivial temporal processing of an ordered input sequence, but what we show here is that the performance can be replicated with a non-recurrent network that processes inputs individually and just aggregates them naively, without taking into account the temporal relationship between frames. Surprisingly, the ordering of the sequence of frame features does not matter.

4.3. Comparison of training modes and architectures

In this section we provide quantitative evidence that training the feature extraction network without the recurrent connection in the sequence processing stage can boost the re-identification performance (accuracy as well as speed).

4.3.1. Training implementation

Our proposed architecture (FNN) is trained using either of the two different training modes as described in section 3.3. The hyperparameters have to be carefully chosen to keep a fair comparison between the modes.

SEQ. For the first mode, called SEQ, we use the values of hyperparameters reported in [12]. The feature dimension is set to 128 and the margin in the contrastive loss function to 2. The inputs are extracted from the full sequences by selecting random subsequences of $L = 16$ consecutive frames. At training time, the inputs are loaded in a mini-batch of size $B = 1$. We alternate between positive and negative pairs of sequences. The network is trained using stochastic gradient descent with a learning rate of $1\text{e}{-3}$ for 1000 epochs. In this case, an epoch is defined as the number of iterations it takes to show N positive examples (N being the number of identities, or pairs of sequences, in the training set). This ensures that all identities were shown as positive pairs exactly once per epoch. The negative pairs are chosen randomly.

FRM. The FRM training mode, in contrast to SEQ, relies on single frames as an input. This mode uses the same value as SEQ for the feature dimension and margin for the contrastive loss. FRM enables training using much lower memory and computational requirements since the processed input is a pair of images instead of a pair of video sequences. Therefore larger mini-batches can be used. A mini-batch of size BL requires the same resources as a mini-batch of size B for SEQ. So in practice, the comparison between the modes is fair if each FRM mini-batch contains $BL = 16$ pairs of images. These pairs are split in half between positive and negative pairs (8 of each). The positive (resp. positive) pairs are extracted in a similar fashion as in SEQ: one frame from each of the two camera sequences corresponding to matching (resp. non-matching) identities is selected. In SEQ, the number of iterations required per epoch is $2N/B$ ($2N$ pairs need to be shown at a rate of B per mini-batch) but in FRM, this number is only $2N/BL$. So in order to show the same amount of data (same number of mini-batches / iterations for SEQ and FRM) it is necessary to train for L times more epochs in FRM mode. In practice, this means that this network is trained for 16000 epochs.

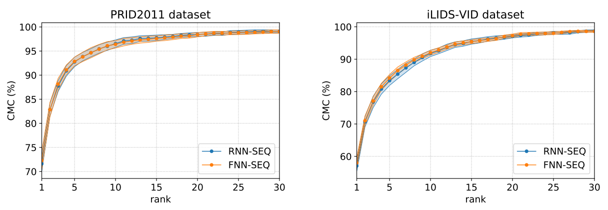

Figure 5. CMC curves obtained on each dataset when training a RNN network and a FNN network in SEQ mode.

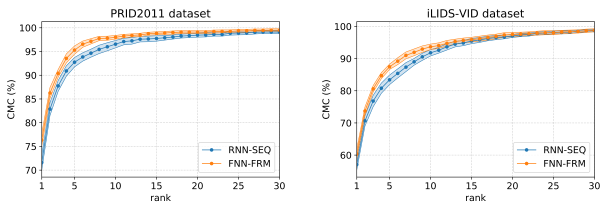

Figure 6. CMC curves obtained on each dataset when training a RNN network in SEQ mode and a FNN network in FRM mode.

It is important to adjust the learning rate in FRM mode. A rule-of-thumb is that when the batch size increases by a factor k, the learning rate should increase by a factor $\sqrt{k}$ [8] or k [8, 4]. When switching to FRM, the batch size is increased by a factor $L = 16$, so it makes sense to increase the learning rate to some value in the range $[4\mathrm{e}{-3}, 16\mathrm{e}{-3}]$. We found that a good value to use was $16\mathrm{e}{-3}$ for the PRID2011 dataset and $8\mathrm{e}{-3}$ for the iLIDS-VID dataset. Finally, for both training conditions, the input data is diversified by performing data augmentation: random cropping and horizontal mirroring are applied to the inputs.

4.3.2. Training with SEQ mode

In this section we compare how the RNN and FNN architectures perform when trained using the SEQ training mode. Results are reported in Fig. 5. The values are very similar for both datasets, and the confidence intervals overlap almost perfectly. This shows that the RNN and FNN architectures are functionally equivalent. The good performance obtained by the feature extraction network does not depend on the presence of the recurrent connections in the sequence processing stage. A feed-forward network that processes frames of a sequence independently and then pools outputs across the sequence works just as well.

4.3.3. Training with more diverse mini-batches

We showed that the performance of the network is the same whether the sub-network in the sequence processing stage is recurrent (RNN) or not (FNN). Without the constraints imposed by the RNN, it makes sense to consider a different training process. Here we examine how the performance of the FNN varies if the training is performed in FRM mode (FNN-FRM), compared to our baseline (RNN trained in SEQ mode). The results for 20 trials are shown in Fig. 6.

For PRID2011, the results are very clear: training using single frames gives a noticeable boost. For all values of the rank k, the average CMC values always exceed (or equate) the baseline values, and the 95% confidence intervals have no overlap for $k \leq 19$ (except for a very small overlap for $k = 13$). The results for iLIDS-VID are also encouraging.

Table 1. Comparison against RNN-based methods. Reported results are the mean CMC values (in %) obtained on the PRID2011 and iLIDS-VID datasets. Best results are shown in blue, second best in cyan. Our method systematically outperforms its recurrent version [12] and performs better or on par with other RNN-based re-identification techniques.

Dataset	PRID2011				iLIDS-VID			
Rank	1	5	10	20	1	5	10	20
RNN [12]	70	90	95	97	58	84	91	96
– (reproduced)	71.6	92.8	96.6	98.5	57.1	83.4	91.8	97.1
RFA-Net [18]	58.2	85.8	93.4	97.9	49.3	76.8	85.3	90.0
Deep RCN [15]	69.0	88.4	93.2	96.4	46.1	76.8	89.7	95.6
Zhou *et al.* [25]	79.4	94.4	-	99.3	55.2	86.5	-	97.0
BRNN [20]	72.8	92.0	95.1	97.6	55.3	85.0	91.7	95.1
ASTPN [17]	77	95	99	99	62	86	94	98
Chen *et al.* [2]	77	93	95	98	61	85	94	97
FNN-FRM (ours)	76.4	95.3	98.0	99.1	58.0	87.5	93.7	97.5

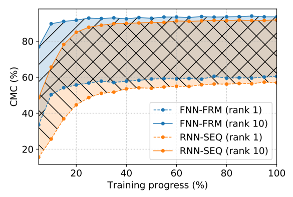

Figure 7. Evolution over the training phase of the CMC values on the iLIDS-VID dataset at rank 1 and rank 10, for both RNN-SEQ and FNN-FRM.

There is an improvement of the mean performance for all values of the rank $k \leq 24$, and the 95% confidence intervals have no overlap for $k \leq 10$. For the few values of k where the baseline performs slightly better than FNN-FRM, the difference never exceeds a statistically insignificant 0.1%.

This experiment shows that switching from SEQ to FRM improves the quality of the training and gives a significant performance boost for some values of k. This illustrates the importance of having more diverse data within a mini-batch.

This raises another limitation of using a sequence to train the network. A subsequence of L consecutive frames from a video of a pedestrian will have a large amount of redundancy so the frame features $f^{(t)}$ will have a high degree of similarity between each other. In our FNN formulation, the output of the sequence processing stage for each time step $o^{(t)}$ only depends on $f^{(t)}$, so these outputs will also be very similar. So it is expected that after average pooling, the feature representation of a sequence (mean of all the $o^{(t)}$) will not be substantially different from the feature representation of a single frame (any $o^{(t)}$). In the extreme case, if a person does not move at all and the inputs are all the same, then all $o^{(t)}$ will be identical and the descriptor for the sequence will also be equal to all $o^{(t)}$. In general, a single frame may yield a descriptor that is slightly more noisy than the descriptor for a sequence, but it should still be sufficient to train our network, while reducing a lot of the wasteful computational cost when computing features for many frames of the same sequence. In fact, we could consider the noise added by computing a descriptor from a single frame compared to a subsequence as some form of data augmentation.

4.3.4. Speeding up training

One last advantage of FRM over SEQ is that each epoch takes fewer iterations for FRM than for SEQ (L times fewer), so the training accuracy improves much faster. See

Fig. 7 for the evolution over time of the CMC values for rank 1 and rank 10. The training progress shown on the x-axis is proportional to the number of iterations, which by design is the same for FRM and SEQ. We can see that after around 25% of the iterations, FRM already converged and the performance does not noticeably improve after that, while SEQ requires the full number of iterations to fully converge. To summarize, not only does FRM converge to a higher performance than SEQ, it also gets there much faster.

4.3.5. Comparison with other RNN-based methods

Table 1 compares our proposed approach with recent RNN-based person re-identification techniques. We note that FNN-FRM always outperforms the baseline (RNN-SEQ). These results also show that our architecture performs better or on par with other methods, that are usually considerably more complex.

5. Conclusion and future work

In this work we revisited the use of RNNs in the context of video person re-identification. After deriving a non-recurrent approximation of a simple RNN architecture, we demonstrated that training our proposed architecture could reach similar performance. Furthermore, we showed that we could significantly improve that performance by training on individual frames instead of sequences, and achieve convergence substantially faster. The obtained performance is better or on par with other RNN-based techniques.

In future work we plan to explore and experiment on other video-level recognition problems, such as action recognition or video-based object recognition. Indeed, these tasks can also be framed as a high-dimensional embedding of a time-dependent image sequence, which is conceptually not different from what is performed in this paper. Some works in action recognition showed some improvements from previous state-of-the-art methods by using recurrent neural networks, and it would be interesting to explore whether simple FNNs can replace RNNs for these applications as well.

References

[1] E. Ahmed, M. Jones, and T. K. Marks. An improved deep learning architecture for person re-identification. In *Computer Vision and Pattern Recognition*, 2015.

[2] L. Chen, H. Yang, J. Zhu, Q. Zhou, S. Wu, and Z. Gao. Deep spatial-temporal fusion network for video-based person re-identification. In *Computer Vision and Pattern Recognition Workshops*, 2017.

[3] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *Computer Vision and Pattern Recognition*, 2016.

[4] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[5] D. Gray and H. Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *European Conference on Computer Vision*, 2008.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, 2016.

[7] M. Hirzer, C. Beleznai, P. M. Roth, and H. Bischof. Person re-identification by descriptive and discriminative classification. In *Scandinavian Conference on Image Analysis*. Springer, 2011.

[8] A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[9] W. Li, R. Zhao, T. Xiao, and X. Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Computer Vision and Pattern Recognition*, 2014.

[10] S. Liao, Y. Hu, X. Zhu, and S. Z. Li. Person re-identification by local maximal occurrence representation and metric learning. In *Computer Vision and Pattern Recognition*, 2015.

[11] K. Liu, B. Ma, W. Zhang, and R. Huang. A spatio-temporal appearance representation for video-based pedestrian re-identification. In *International Conference on Computer Vision*, 2015.

[12] N. McLaughlin, J. M. del Rincon, and P. Miller. Recurrent convolutional network for video-based person re-identification. In *Computer Vision and Pattern Recognition*, 2016.

[13] S. Pedagadi, J. Orwell, S. Velastin, and B. Boghossian. Local fisher discriminant analysis for pedestrian re-identification. In *Computer Vision and Pattern Recognition*, 2013.

[14] T. Wang, S. Gong, X. Zhu, and S. Wang. Person re-identification by video ranking. In *European Conference on Computer Vision*, 2014.

[15] L. Wu, C. Shen, and A. van den Hengel. Deep recurrent convolutional networks for video-based person re-identification: An end-to-end approach. *arXiv preprint arXiv:1606.01609*, 2016.

[16] F. Xiong, M. Gou, O. Camps, and M. Sznaier. Person re-identification using kernel-based metric learning methods. In *European conference on computer vision*, 2014.

[17] S. Xu, Y. Cheng, K. Gu, Y. Yang, S. Chang, and P. Zhou. Jointly attentive spatial-temporal pooling networks for video-based person re-identification. In *International Conference on Computer Vision*, 2017.

[18] Y. Yan, B. Ni, Z. Song, C. Ma, Y. Yan, and X. Yang. Person re-identification via recurrent feature aggregation. In *European Conference on Computer Vision*, 2016.

[19] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Deep metric learning for person re-identification. In *International Conference on Pattern Recognition*, 2014.

[20] W. Zhang, X. Yu, and X. He. Learning bidirectional temporal cues for video-based person re-identification. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[21] R. Zhao, W. Ouyang, and X. Wang. Unsupervised salience learning for person re-identification. In *Computer Vision and Pattern Recognition*, 2013.

[22] L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, and Q. Tian. Mars: A video benchmark for large-scale person re-identification. In *European Conference on Computer Vision*, 2016.

[23] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In *International Conference on Computer Vision*, 2015.

[24] L. Zheng, Y. Yang, and A. G. Hauptmann. Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984*, 2016.

[25] Z. Zhou, Y. Huang, W. Wang, L. Wang, and T. Tan. See the forest for the trees: Joint spatial and temporal recurrent neural networks for video-based person re-identification. In *Computer Vision and Pattern Recognition*, 2017.