Active Descriptor Learning for Feature Matching

Aziz Koçanaoğulları[1] and Esra Ataer-Cansızoğlu[2](✉)

[1] Northeastern University, Boston, MA, USA
akocanaogullari@ece.neu.edu
[2] Mitsubishi Electric Research Labs (MERL), Cambridge, MA, USA
cansizoglu@ieee.org

Abstract. Feature descriptor extraction lies at the core of many computer vision tasks including image retrieval and registration. In this paper, we present an active learning method for extracting efficient features to be used in matching image patches. We train a Siamese deep neural network by optimizing a triplet loss function. We develop a more efficient and faster training procedure compared to the state-of-the-art methods by increasing difficulty during batch training. We achieve this by adjusting the margin in the loss and picking harder samples over time. The experiments are carried out on Photo Tourism dataset. The results show a significant improvement on matching performance and faster convergence in training.

Keywords: Feature matching · Active learning · Curriculum learning

1 Introduction

Extraction of feature descriptors lies at the core of many computer vision tasks. There exists a tremendous amount of work to represent image patches and match them across images. Initial studies included hand-crafted features such as SIFT [8], SURF [3] and ORB [9]. Recently, with the rise in deep neural network methods, learned features are developed and introduced as a more efficient alternative to the hand-crafted features.

Recent work on learned feature representation focus on formulating the loss function or designing neural network architectures. However, training for feature learning is challenging since it heavily depends on the selected matching/non-matching pairs and the initialization. In this paper, we present an active learning procedure that increases the difficulty of batch training over time. We achieve this by (i) increasing the margin between the similarities of matched and non-matched pairs in the loss function and (ii) picking harder sample pairs over time. Similar to a kid learning pattern matching starting from easier primitive shapes, we start batch training by feeding samples with low loss values that are easily detected as matching or non-matching with our current model. Gradually we increase the difficulty of patterns presented while expecting to see a better separation between examples. Thus, over time we add harder samples with higher loss values, while increasing the margin between distances of matching and non-matching pairs in our loss function. We demonstrate the use of our technique for matching image patches in Photo Tourism dataset [16]. The experiments show a significant improvement in performance and a faster convergence rate compared to the state-of-the-art learning-based features. Our method provides more robust features with a significant speed up in training.

1.1 Existing Work

In deep feature learning, final performance of the feature matching is highly dependent on sample selection. In order to increase performance of the model, learning should be supervised. As an obvious example, a feature representation that is learned by training with matching samples only ends up having a constant function, in contrast a representation coming from a training with non-matching samples only has a scattered range.

The importance of selecting a $1:1$ match and non-match ratio during training was emphasized in [7,17]. Zeng et al. [19] stated that they accumulate training samples in a reservoir and generate batches online without violating $1:1$ ratio constraint. Similarly, Balntas et al. [2] presented triplet-loss as a way of enabling a totally randomized batch sampling. Since triplets consist of two matching samples and a non-matching sample, it already satisfies $1:1$ ratio between matching and non matching pairs. However, these approaches suffer from incorrect gradient estimates due to batches being dominated by samples close to zero loss. In order to tackle this problem, authors in [10,14] proposed to discard training samples with zero loss. This approach suffers from low sample noise assumption. Discarding such samples causes overestimated gradients, making the training sensitive to outliers. Hence it slows down the training and prevents convergence to a local minimum. In order to solve this issue, instead of discarding training samples directly, it is more convenient to select samples actively to ensure convergence. Simo et al. [10] used more matching pairs in the beginning of training and increase the number of non-matches in each batch as the training progress. Therefore, their method learns boundaries after determining cluster centers. Increasing difficulty was structured further by authors in [1]. Their technique involved selecting samples based on their discriminative scores on a state-of-the-art model. Although these methods aim to increase difficulty in training by actively selecting samples, they ignore the effect of the loss function in learning the feature space. The non-match distance, namely margin, for both pairwise and triplet loss in all these scenarios is a constant that is cross-validated across random trials. In this approach, after certain number of iterations learned features do not change since most of the training samples already satisfy the condition. Hence, a better separation between clusters is omitted. In this work we propose to use an active objective that adjusts the margin (correlated with the intra-class distance) gradually to ensure convergence to better local optimum.

In order to address these drawbacks we propose an active training procedure for feature learning. We aim to achieve better local minimum with the following contributions (i) by actively increasing difficulty of the training with a sequence of loss functions that converge to a local optimum and (ii) by an active sample batch selection based on descriptive scores to increase difficulty of the training over time. We build a deep descriptor map that finds an embedding in a lower dimensional Euclidean space where clusters are separable. We evaluate the performance of our model by using a publicly available dataset to compare with the literature.

2 Method

We propose a curriculum for the training session and increase difficulty of the training over time. We increase difficulty by adjusting the loss function and picking batches during training. Before discussing our contributions, we first present the problem formulation and notation:

Preliminaries/Notation
Given a set of clusters $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_k\} = \mathcal{D}$, with corresponding distributions $p^{\mathcal{D}_i}$, in feature learning the aim is to learn a mapping f that has a range where each cluster is separable in the range space. Let $f_\theta : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^M$ be the mapping from image domain to the feature domain parameterized by θ with $N^2 >> M$ and d be a distance metric in range. Feature map tries to achieve the following:

$$\begin{aligned} &\mathrm{d}(f_\theta(a), f_\theta(p)) < d(f_\theta(a), f_\theta(n)) \\ &\forall i, j \neq i \text{ and } \forall a, p \sim p^{\mathcal{D}_i}, n \sim p^{\mathcal{D}_j} \end{aligned} \tag{1}$$

We follow a, p, n notation for '*anchor*', '*pair*', '*non-pair*' respectively, which is a conventional naming in the field. In many applications cluster information is not accessible or number of clusters is arbitrarily large, e.g. patch matching, thus maximum likelihood over the indicator defined in (1) is not possible. Hence, the problem is approximately solved with pairwise loss [6] or triplet loss [15] efficiently where only match or non-match information is used. In this paper we focus on the triplet loss, which enforces that the distance between non-matching samples should be at least a margin m larger than the distance between matching pairs. The loss $\mathcal{L}$ is defined as,

$$\mathcal{L}_m(a, p, n, f_\theta) = d(f_\theta(a), f_\theta(p)) - d(f_\theta(a), f_\theta(n)) + m \tag{2}$$

Conventionally, distance metric d is selected as the Euclidean distance to have a Euclidean similarity space. Deep feature maps are learned back-propagating the triplet loss through the network. The parameters are optimized with the following optimization:

$$\hat{\theta} = \arg\min_\theta \sum_{\mathcal{T}} \mathcal{L}_m(a, p, n, f_\theta) \tag{3}$$

Here $\mathcal{T} = \{(a, p, n) | \forall a, p \in \mathcal{D}_i, n \in \mathcal{D}_j, j \neq i\}$ denotes set of sampled triplets. Triplets are usually pre-generated before training and fed through the network as batches to stochastically solve (3), thus the local minimum is biased to the batches at step i of epoch e as $\mathcal{T}_e^i \subset \mathcal{T}$. Additionally based on m, training triplets that satisfy the constraint may yield to incorrect gradient estimate and hence increase training time and may yield bad local minimum.

2.1 Loss Function Sequence

In feature learning, margin as a hyperparameter is selected based on best convention of the dataset and such convention is learned with multiple trials. We propose to approximate margin empirically based on the trajectory of the error during training. Let $\hat{m}$ denote the true margin for a particular local minimum $\hat{\theta}$. Without loss of generality let us assume $\mathcal{L} \to 0$ then $\mathcal{L}_{\hat{m}+\varepsilon}(\hat{\theta}) > 0 \, \forall \varepsilon > 0$. Let $\mathcal{L}_{m_e}$ be a sequence of functions that converge to the correct estimate of the loss $\mathcal{L}_{\hat{m}}(\hat{\theta})$ and hence allowing us to approximately find solution during training:

$$\lim_{e \to \infty} \mathcal{L}_{m_e}(a, p, n, f_\theta) \to \mathcal{L}_{\hat{m}}(a, p, n, f_{\hat{\theta}}) \;\; \forall a, p, n \tag{4}$$

Analysis of deep learning error trajectory is difficult, hence this is not trivial to find such sequence of margins that satisfy such convergence. Without loss of generality, we assume loss in training is non-increasing epoch-wise $\mathcal{L}_m(\theta_e) \geq \mathcal{L}_m(\theta_{e+1})$ where θ_e denotes model parameters at epoch e. One possible approach would be starting from a margin of $m_0 = 0$ and increasing the margin by a constant when an error bound is reached in the training loss. Thus, we can make the following proposition:

Proposition 1. *Given a separable set, $m_0 = 0$, and let $\bar{\mathcal{L}} = \sum_{(a,p,n)} \mathcal{L}$ and $\mathbb{1}(.)$ be indicator function; $\exists c_e \in \mathbb{R}$ s.t. $m_{e+1} = m_e + c_e \times \mathbb{1}(\bar{\mathcal{L}}_{m_e} \leq \varepsilon)$ satisfies (4) for an arbitrary θ with $\varepsilon \leq \mathcal{L}_{\hat{m}}(\hat{\theta}) \approx 0$.*

Proof. $\mathcal{L}_m$ is a convex function of m for a fixed θ. $\mathcal{L}_m = 0 \; \forall m \in [0, \hat{m}]$ and $\mathcal{L}$ is monotonically increasing $\forall m \in [\hat{m}, \infty]$, hence $\exists \hat{m}$ the unique solution.

By definition for fixed m $\mathcal{L}(\theta_e)$ is non increasing wrt. e and converges to a point. Hence, if $\exists \varepsilon = \mathcal{L}_{\hat{m}}(\hat{\theta})$, the network converges to minimum loss and $\hat{\theta}, \hat{m}$ are found in order.

We can observe that the proposition works in the case of noiseless samples. We propose to pick an empirical ε, due to incorrect gradient estimates and the sample observation noise. Here the selection of ε will affect the final convergence of training as it is indirectly enforced as the minimum loss expected from the learned feature space.

The following corollary states that instead of putting a threshold on the total loss as a condition for increasing margin, we can consider the number of samples with a zero loss and hence instead of a loss threshold we can put a threshold on the size of the set of zero loss samples.

Corollary 1. *$m_{e+1} = m_e + c_e \times \mathbb{1}(|A_e| = 0)$ where $A_e = \{(a, p, n) | \mathcal{L}_{m_e}(a, p, n, \theta_e) > 0 \, \forall (a, p, n) \in \mathcal{T}\}$ satisfies (4).*

The corollary follows the proposition by considering the number of samples in A_e, which contain the samples with nonzero loss in epoch e. However, due to noise in samples, having an empty set of samples with nonzero loss is not achievable. Considering noise and outliers in the dataset, we propose to update m_e with a single scalar if the number of samples in A_e is smaller than a threshold. This threshold can be selected based on a proportion of samples in the training set. The details of the margin update method can be seen in Algorithm 1. At the first line, margin m, constant margin update factor c and proportion k of samples to be used as a threshold for number of nonzero loss samples are initialized. Each epoch starts with an empty set A_e (Line 3). Lines 5 and 6 receive the training batch and updates model parameters respectively. In line 7, the number of samples with loss values close to 0 are added to the set A_e. Finally, in line 8 we test whether the number of samples in A_e is greater than a certain proportion of total training samples. If so, the margin is updated by c.

Algorithm 1. Margin Update: training procedure given all samples $\mathcal{T}$ consisting of batches $\mathcal{T}_e^i$ for ith batch of epoch e.

```
1: $m \in \mathbb{R}^+, c \in \mathbb{R}^+, k \in \mathbb{N}^+$
2: **procedure** TRAINING($e \rightarrow$ num-epoch)
3:     $A_e \leftarrow \{\}$
4:     **for** $i \rightarrow$ num-batch **do**
5:         $(a, p, n) \leftarrow \mathcal{T}_e^i$
6:         $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}_m(a, p, n, m)$
7:         $A_e \leftarrow A_e \cup \{(a, p, n) | \mathcal{L}_m(a, p, n, f_\theta) = 0, \forall (a, p, n) \in \mathcal{T}_e^i\}$
8:     **if** $|A_e|/|\mathcal{T}| > k$ **then** $m \leftarrow m + c$
9: **return** $\theta$
```

2.2 Active Batch Sampling

Going from easy to hard during training can be satisfied feeding triplets with smaller/larger loss value at a current epoch. For a particular dataset D with k clusters, number of triplets is approximately $\binom{k}{2}|D_i|\binom{|D_i|}{2}$ where D_i is taken as average number of samples in a cluster. Hence determining all triplets might slow down training. Therefore, we propose a stochastic approximate of sample selection for practical reasons. For a fixed batch size b we randomly sample $2 \times b$ triplets from $\mathcal{T}$ and pick a subset of b samples of our interest. Such selection might vary due to the interest and in order to have a curriculum with increasing difficulty we propose two selection methods.

First few epochs of the deep learning is biased to the random initialization of the network parameters. Therefore first few epochs form the baseline of the similarity range by putting samples onto space. In order to have descriptive cluster centers, we propose to use samples that are easy to be separated and hence we pick samples with low triplet loss. However, observe that, if we choose triplets with close to zero loss only, we end up having incorrect gradient estimates. Therefore before selection we limit sampled triplets to the ones without zero loss. Let us denote such subset of triplets with $\bar{\mathcal{T}}_e^i = \{(a,p,n) | \mathcal{L}_m(a,p,n,f_\theta) \neq 0, \forall (a,p,n) \in \mathcal{T}_e^i\}$, the batch for training with easy samples is formed as the following,

$$\hat{\mathcal{T}}_e^i = \arg\min_{\mathcal{T} \subset \bar{\mathcal{T}}_e^i} \sum_{(a,p,n) \in \mathcal{T}} \mathcal{L}_m(a,p,n,f_\theta) \ \text{ s.t. } |\mathcal{T}| = b \tag{5}$$

As training moves forward it is expected that the cluster centers are well structured and hence we propose to fine tune the similarity map by using the hard samples. Since aim in training is to fit the data, in contrast to the initial step we do not discard the triplets with zero loss. Otherwise we will be prune to the dataset noise and outliers. Discarding triplets with zero loss results in an overestimated gradient and cause possible divergence in the long run, conversely keeping samples will saturate the gradient and avoid changes in the parameter domain. Our hard sample selection policy is as the following:

$$\hat{\mathcal{T}}_e^i = \arg\max_{\mathcal{T} \subset \bar{\mathcal{T}}_e^i} \sum_{(a,p,n) \in \mathcal{T}} \mathcal{L}_m(a,p,n,f_\theta) \ \text{ s.t. } |\mathcal{T}| = b \tag{6}$$

The details of our active sampling policy are given in Algorithm 2. Given number of epochs f, the algorithm generate batches by easy sampling until epoch f (Line 4) and by hard sampling after epoch f (Line 5). Lines 6 and 7 receives the batch and updates the model parameters respectively.

Algorithm 2. Active sampling: shows the steps of active sampling during batch training given number of epochs f. The algorithm selects the batches with easy sampling until epoch f and switches to hard sampling afterwards.

```
1: $f \in \mathbb{N}^+$
2: **procedure** TRAINING($e \rightarrow$ num-epoch)
3:     **for** $i \rightarrow$ num-batch **do**
4:         **if** $e < f$ **then** $\mathcal{T}_e^i \leftarrow$ (5)
5:         **else** $\mathcal{T}_e^i \leftarrow$ (6)
6:         $(a,p,n) \leftarrow \mathcal{T}_e^i$
7:         $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}_m(a,p,n,m)$
8: **return** $\theta$
```

In the next section we propose the implementation details of active similarity learning and how we put loss updates and sampling together.

3 Implementation Details

Margin update and active sampling procedures are decoupled in training. Namely, sampling is called at each batch state and margin update is called after each epoch. Therefore we simply use both methods without any further

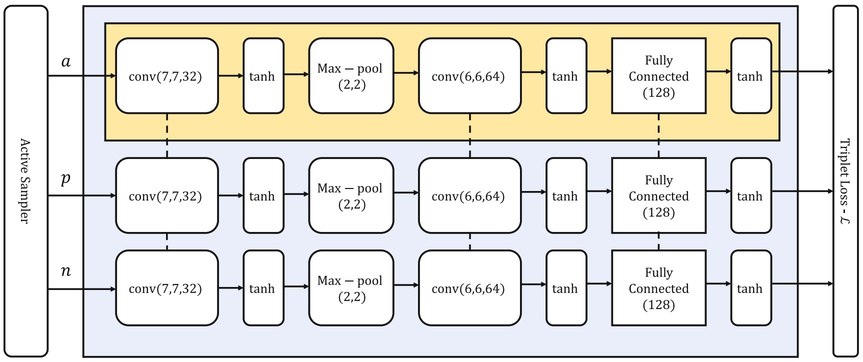

Fig. 1. Network architecture used for descriptor learning. Active sampler provides a, p, n as anchors, positives and negatives respectively for training. As denoted with dashed lines, convolutional and fully connected layers share parameters. During inference (after training) only one of the Siamese networks is used.

adjustments. Initial margin value is set as $m = 1$ for most of the experiments and c is selected from the range $[0, 1]$ empirically in margin update algorithm. The proportion of samples is set as $k = 0.7$ which is used to decide whether to do a margin update or not. Our batch size is $b = 128$, thus we performed sampling from 256 triplets. For active sampling algorithm, we switched from easy to hard sampling after $f = 2$ epochs.

In order to better evaluate the affect of the proposed active learning method, we use the same shallow network architecture defined in [2]. The architecture consists of Conv(7, 7)-Tanh-MaxPool(2, 2)-Conv(6, 6)-Tanh-FullyConnected(128) as seen in Fig. 1. The system is implemented in Tensorflow. During training we use stochastic gradient descent with momentum [12] with a fixed learning rate of 10^{-4} and a momentum of 0.9.

4 Experiments

The goal of this work is to improve the performance of feature matching by following an active learning procedure. In addition to sample selection during batch training, we increase the difficulty of objective function. We carry out two sets of experiments. First we demonstrate the use of margin update for feature learning on MNIST dataset. Second we carry out experiments on the image patch benchmarks of Photo Tourism dataset in order to demonstrate the performance of our technique in local descriptor matching problem.

Feature Learning on MNIST Dataset
In order to show how active adjustment of objective affects feature learning, we used MNIST hand written digits dataset and apply sequence of loss function idea on training. MNIST is a good example to better observe descriptor difference,

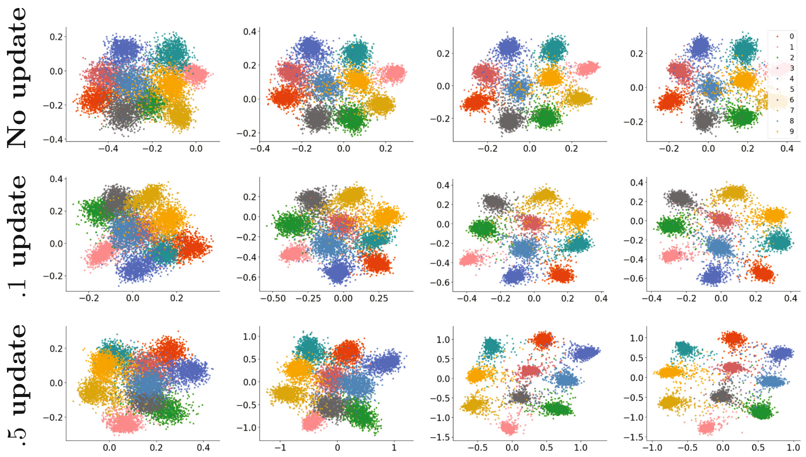

Fig. 2. Descriptor learning on MNIST dataset. Initial margin is selected as .2. Rows represent no-margin update, margin update with .1 and .5 in order. Each column is a snapshot of test images through training in ascending order.

since it has a small number of clusters (i.e. 10 clusters for 10 digits). We set initial margin as $m = 0.2$ and we do not limit output of the network into an interval (e.g. $[0, 1]$) in order to visualize the separation. The learned features on MNIST dataset can be seen in Fig. 2 in various times during training. Rows represent no-margin update, margin update with $c = 0.1$ and margin update with $c = 0.5$ respectively. Each column is a snapshot of learned features in ascending epoch indexes. As can be seen gradual increase of margin during training increases class separation without scattering the descriptors through space. Moreover, a more aggressive margin update step (i.e. $c = 0.5$) can give a larger separation between classes as expected.

Patch Pair Classification
We evaluate the performance of the model in local descriptor learning, where the aim is to determine whether two provided patches are different images of the same point in the world. Performance in patch pair similarity is conventionally reported using receiver operation characteristics (ROC) curve [16]. ROC curve is formed using pairs where matches have label 1 and non matches are 0 and calculation of the curve using a scalar threshold is obvious. We report false positive rate at .95 recall (true positive rate). Hence this measure tells us how likely a model is to put matching pairs together and non-matching pairs apart. In our experiments we use the UBC patches taken from Photo Tourism dataset presented in [16]. This dataset contains three different patch sets extracted using SIFT detector and each set includes pre-defined pairs for the dataset. We compare our method with hand SIFT as a hand crafted feature baseline with other learned descriptors [4,13,18]. We also compare the method with other deep learning techniques [5,10,18]. Specifically; Han et al. [7] propose a network with con-

Table 1. False positive rate at .95 recall for the Photo Tourism dataset. Proposed method is on the bottom of the table in italics. Better results are indicated with bold-font. Models trained using one training set and tested on the test-set. If a row has 3 numbers, that indicates model is trained using both training dataset.

Train-set		Notredame	Liberty	Notredame	Yosemite	Yosemite	Liberty
Test-set		Yosemite		Liberty		Notredame	
Method	Dim						
SIFT	128	27.29		29.84		22.53	
[13]	64	15.86	19.63	18.05	21.03	13.73	14.15
[4]	29	13.55	–	16.85	18.27	11.98	–
[11]	64	10.54	11.63	12.88	14.82	7.11	7.52
[5]	128	30.22		14.26		9.64	
[10]	128	16.19		8.82		4.54	
[18]	256	15.89	19.91	13.24	17.25	8.38	6.01
[7]	128	12.17	14.40	9.48	15.40	8.27	5.18
[7]	256	11.00	13.58	8.84	13.08	5.67	3.87
[2]	128	7.08	7.82	7.22	9.79	**3.85**	**3.12**
active	128	**6.64**	**5.16**	**6.28**	**4.83**	4.44	**3.12**

trastive loss which is specifically optimized for matching. Furthermore, Balantas et al. [2] propose anchor swapping for training a shallow descriptor network with triplet loss. Since Balantas proposes the best performing model, through the experiments we refer this model as "conventional".

In order to compare proposed method with previous work we also use the same pre-defined pairs and generate our triplets randomly based on provided information. We generate 1, 28M triplets prior to the training which is a lot smaller than the number of training samples in other methods. Even with less number of training samples, we observed that active learned descriptors perform better in matching. We report the results in Table 1. Compared to the other variants with similar dimensionality (e.g we omitted 4096 dimensional representation presented in [7]) we achieve lower error rates. For implementation we set both $m = 1$ and number of initialization epochs to $f = 1$ by default. However, we observed that, subset Yosemite includes the most number of triplets with 0 loss with $m = 1$ and random initialization. In order to prevent over-fitting we initialized training with $m = 3$ and used $f = 2$ initialization epochs. We cross validated across different margin increments $c \in [0, 1]$ and reported the best. Empirically we observed that $c = 0.5$ improves performance over all scenarios. For our comparison with performance we directly use reported values of each proposed method. For evaluating the gradual performance increase to indicate the impact of the active learning, we use the same baseline (e.g. same training data size, learning rate etc.) for the conventional method and proposed method.

In order to show increase in performance during training we compare proposed method with its components only and conventional triplet based training. We train the same architecture using Notredame dataset and validate on Liberty dataset. We visualize our findings in Fig. 3. Observe that, propose method

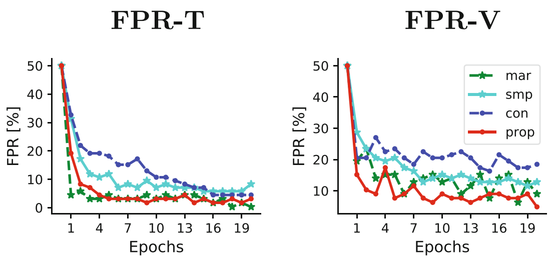

Fig. 3. Training on Notredame, validating on liberty dataset. Figure divided columns of false positive rate (FPR) for [V], [T] which denote validation and training respectively. Compares proposed method (prop) to active sampling only (smp), active margin only (mar) and conventional method (con) in model performance.

outperforms other approaches. Margin increases causes faster convergence in training dataset, unfortunately causes over fitting if samples are not correctly selected. Active sampling results a smoother training and validation error trajectory compared to other methods. We conclude that by stating, proposed method achieves better performance and hence using both margin updates and active sample selection together is crucial.

We also compare our method's performance to the conventional methods for both where training set has 1.28M and 5M training samples. We visualize our findings in Fig. 4. The impact of the proposed active learning procedure is

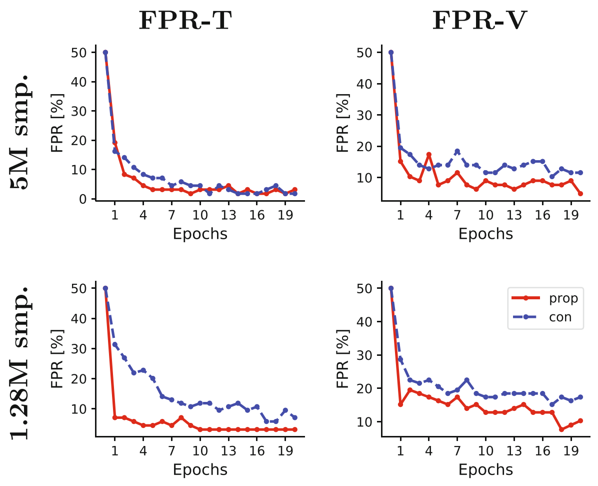

Fig. 4. Training on Liberty, validating on Notredame dataset. Figure divided columns of false positive rate (FPR) for [V], [T] which denote validation and training respectively. First row visualizes results for 5M pre-sampled triplets, second row visualizes results for 1.28M pre-sampled triplets. **act** and **con** denotes active and conventional training respectively.

more clear when number of samples are smaller. False positive rates in both training and validation for 1.28M triplet case has a marginal difference than conventional training and hence we conclude that carefully selecting samples increase the performance. We also note that, although loss functions for both cases are different, the loss values during training are close to each other for 5M triplet case and increasing margin did not increase the loss in 1.28M case. Hence, we state that initial margin is underestimated for such scenario which is later compensated by the proposed method. But, it is observable that as number of samples increase, active sampling increases performance less. Therefore we state that active sampling is useful for small training data set applications. In Fig. 4 we only demonstrate first 20 epochs of the training, at further iterations performance margin between the conventional method and the proposed method decreases.

5 Conclusion

We presented an active feature learning method. Our method actively increase the difficulty of training by adjusting th emarging between matching and non-matching pairs and picking harder samples over time during batch training. We demonstrates the use of our algorithm on the problem of feature matching. The experiments were carried out on Photo Tourism dataset. The presented technique outperforms existing methods in matching performance while speeding up training time significantly. We additionally consider using statistical and/or nearest neighborhood based outlier rejection methods during training to further increase the performance of active sampling. Additionally, proposed margin updates can be designed as a gradient update based on the loss function directly and hence we also consider incorporating a generalized margin update with faster convergence guarantees. Future work will focus on the use of geometric information such as depth of the patch and camera view-point in feature learning.

Acknowledgements. We thank Alan Sullivan, Radu Corcodel and Anoop Cherian for their helpful comments. This work was supported by and done at MERL.

References

1. Appalaraju, S., Chaoji, V.: Image similarity using deep CNN and curriculum learning. arXiv preprint arXiv:1709.08761 (2017)
2. Balntas, V., Riba, E., Ponsa, D., Mikolajczyk, K.: Learning local feature descriptors with triplets and shallow convolutional neural networks. In: BMVC, vol. 1, p. 3 (2016)
3. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006). https://doi.org/10.1007/11744023_32
4. Brown, M., Hua, G., Winder, S.: Discriminative learning of local image descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **33**(1), 43–57 (2011)

5. Fischer, P., Dosovitskiy, A., Brox, T.: Descriptor matching with convolutional neural networks: a comparison to SIFT. arXiv preprint arXiv:1405.5769 (2014)
6. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1735–1742. IEEE (2006)
7. Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A.C.: MatchNet: unifying feature and metric learning for patch-based matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3279–3286 (2015)
8. Lowe, D.G.: Object recognition from local scale-invariant features. In: 1999 the Proceedings of the Seventh IEEE International Conference on Computer vision, vol. 2, pp. 1150–1157. IEEE (1999)
9. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 2564–2571. IEEE (2011)
10. Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., Moreno-Noguer, F.: Discriminative learning of deep convolutional feature point descriptors. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 118–126 (2015)
11. Simonyan, K., Vedaldi, A., Zisserman, A.: Learning local feature descriptors using convex optimisation. IEEE Trans. Pattern Anal. Mach. Intell. **36**(8), 1573–1585 (2014)
12. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International Conference on Machine Learning, pp. 1139–1147 (2013)
13. Trzcinski, T., Christoudias, M., Lepetit, V., Fua, P.: Learning image descriptors with the boosting-trick. In: Advances in Neural Information Processing Systems, pp. 269–277 (2012)
14. Wang, J., et al.: Learning fine-grained image similarity with deep ranking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1386–1393 (2014)
15. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. **10**(Feb), 207–244 (2009)
16. Winder, S.A., Brown, M.: Learning local image descriptors. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2007, pp. 1–8. IEEE (2007)
17. Yi, K.M., Trulls, E., Lepetit, V., Fua, P.: LIFT: learned invariant feature transform. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9910, pp. 467–483. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_28
18. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4353–4361 (2015)
19. Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T.: 3Dmatch: learning local geometric descriptors from RGB-D reconstructions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 199–208. IEEE (2017)