MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching

Xufeng Han[†] Thomas Leung[‡] Yangqing Jia[‡] Rahul Sukthankar[‡] Alexander C. Berg[†]
[†]University of North Carolina at Chapel Hill [‡]Google Research
xufeng@cs.unc.edu {leungt,jiayq,sukthankar}@google.com aberg@cs.unc.edu

Abstract

Motivated by recent successes on learning feature representations and on learning feature comparison functions, we propose a unified approach to combining both for training a patch matching system. Our system, dubbed MatchNet, consists of a deep convolutional network that extracts features from patches and a network of three fully connected layers that computes a similarity between the extracted features. To ensure experimental repeatability, we train MatchNet on standard datasets and employ an input sampler to augment the training set with synthetic exemplar pairs that reduce overfitting. Once trained, we achieve better computational efficiency during matching by disassembling MatchNet and separately applying the feature computation and similarity networks in two sequential stages. We perform a comprehensive set of experiments on standard datasets to carefully study the contributions of each aspect of MatchNet, with direct comparisons to established methods. Our results confirm that our unified approach improves accuracy over previous state-of-the-art results on patch matching datasets, while reducing the storage requirement for descriptors. We make pre-trained MatchNet publicly available.[1]

1. Introduction

Patch-based image matching is used extensively in computer vision. Finding accurate correspondences between patches is instrumental in a broad variety of applications including wide-baseline stereo (*e.g.*, [14]), object instance recognition (*e.g.*, [13], fine-grained classification (*e.g.*, [36]), multi-view reconstruction (*e.g.* [20]), image stitching (*e.g.* [4]), and structure from motion (*e.g.* [17]).

Since 1999, the advent of the influential SIFT descriptor [13], research on patch-based matching has attempted to improve both accuracy and speed. Early efforts focused on identifying better affine region detectors [16], engineering more robust local descriptors [7, 15], and exploring improvements in descriptor matching using alternate distance metrics [8, 9].

Early efforts at unsupervised data-driven learning of local descriptors (*e.g.*, [11]) were typically outperformed by modern engineered descriptors, such as SURF [1], ORB [18]. However, the greater availability of labeled training data and increased computational resources has recently reversed this trend, leading to a new generation of learned descriptors [3, 22, 27, 28] and comparison metrics [9]. These approaches typically train a nonlinear model discriminatively using large datasets of patches with known ground truth matches and serve as motivation for our work.

Concurrently, approaches based on deep convolutional neural networks have recently made dramatic progress on a range of difficult computer vision problems, including image classification [12], object detection [6], human pose estimation [26], and action recognition in video [10, 23]. This line of research highlights the benefits of jointly learning a feature representation and a classifier (or distance metric), which to our knowledge has not been adequately explored in patch-based matching.

In this paper, we propose a unified approach that jointly learns a deep network for patch representation as well as a network for robust feature comparison. In our system, dubbed MatchNet, each patch passes through a convolutional network to generate a fixed-dimensional representation reminiscent of SIFT. However, unlike in SIFT, where two descriptors are compared in feature space using the Euclidean distance, in MatchNet, the representations are compared using a learned distance metric, implemented as a set of fully connected layers.

Our contributions include: 1) A new state-of-the-art system for patch-based matching using deep convolutional networks that significantly improves on the previous results. 2) Improved performance over the previous state of the art [22] using smaller descriptors (with fewer bits). 3) A careful set of experiments using standard datasets to study the relative contributions of different parts of the system, showing that MatchNet improves over both hand-crafted and learned descriptors plus comparison functions. 4) Finally we provide a public release of MatchNet trained using our own large

[1] http://www.cs.unc.edu/~xufeng/matchnet

978-1-4673-6964-0/15/$31.00 ©2015 IEEE

collection of patches.

The remainder of this paper is organized as follows. Section 2 discusses related work, focusing on learned descriptors and metric learning. Section 3 details the network architecture behind MatchNet. Section 4 explains how the joint training and the two-stage evaluation pipeline are performed. Section 5 presents the experimental methodology and results on a suite of standard datasets. We conclude with a summary and ideas for future work.

2. Related work

Much previous work considers improving some components in the detector-descriptor-similarity pipeline for matching patches. Here we address the most related work that considers learning descriptors or similarities, organized by goal and the types of non-linearity used.

Feature learning methods such as [3], [28] and [22] encode non-linearity into the procedure for mapping intensity patches to descriptors. Their goal is to learn descriptors whose similarity with respect to a chosen distance metric match the ground truth. For [3] and [22], the procedure includes multiple parameterized blocks of gradient computation, spatial pooling, feature normalization and dimension reduction. [28] uses boosting with weak learners consisting of a family of functions parameterized by gradient orientations and spatial location. Each weak learner represents the result of feature normalization, orientation pooling and thresholding in its $+1/-1$ output. Weighting and combining multiple weak learners builds a highly non-linear mapping from gradients to robust descriptors. Different types of learning algorithms are proposed to find the optimal parameters: Powell minimization, boosting and convex optimization for [3], [28] and [22], respectively. In [3] and [22] the similarity functions are simply the Euclidean distance. [28] uses a Mahalanobis distance and jointly learns the descriptors and the metric. In comparison, our proposed feature extraction uses a deep convolutional network with multiple convolutional and spatial pooling layers plus an optional bottle neck layer to obtain feature vectors, followed by a similarity measure also based on neural nets.

Metric learning methods such as [8] and [9] learn a similarity function between descriptors that approximates a ground truth notion of which patches should be similar, and achieve results that improve on simple similarity functions, most often the Euclidean distance. Jain *et al*. [8] introduces non-linearity with a predefined kernel on patches. A Mahalanobis metric is learned on top of that similarity. Jia *et al*. [9] uses a parametric distance based on a heavy-tailed Gamma-Compound-Laplace distribution, which approximates the empirical distribution of elements in the difference of matching SIFT descriptors. The parameters for this distance are estimated using the training data. In comparison, we use a two-layer fully connected neural network to learn the pairwise similarity, which has the potential to embrace more complex similarity functions beyond distance metrics such as Euclidean.

Semantic hashing or embedding learning methods learn non-linear mappings to generate low dimensional representations, whose similarity in some easy-to-compute distance metric (e.g., Hamming distance) correlates with the semantic similarity. This can be done using neural networks, e.g., [5] and [19] with a two-tower structure and recently [33] that samples triplets for training. Spectral hashing [34] or boosting [21, 27] can also be used to learn the mapping. This approach can be applied to raw image input [19] as well as to local feature descriptors [25]. In comparison, although we do not map input to an intermediate embedding space explicitly, the representation provided by our feature extraction network naturally serves the purpose, and the dimensionality can be controlled depending on the accuracy vs. storage and computation tradeoff. We explore and analyze such effects in Section 5.

Our network structure is similar to the recent preprint [37] for stereo matching, with a notable difference that we use pooling layers to learn compact representations from patches. Our approach, MatchNet, is designed for general wide-baseline viewpoint invariant matching, a significantly different problem than the more local matching problem in stereo. As one example, for wide-baseline matching, scale estimation from the key point descriptor may not be accurate. The pooling layers increase the robustness of the network robust to such variation. MatchNet has several other architectural differences, an additional convolutional layer, two fewer fully connected layers, and various differences in filter supports and layer complexity compared to [37]. We evaluate some architectural variations in Section 5.

3. Network architecture

MatchNet is a deep-network architecture (Fig. 1 C) for jointly learning a feature network that maps a patch to a feature representation (Fig. 1 A) and a metric network that maps pairs of features to a similarity (Fig. 1 B). It consists of several types of layers commonly used in deep-networks for computer vision. We show details of these layer in Table 1, and discuss some of the high-level architectural choices in this section.

The feature network: The feature network is influenced by AlexNet [12], which achieved good object recognition performance. We use many fewer parameters and do not use Local Response Normalization or Dropout. We use Rectfied Linear Units (ReLU) as non-linearity for the convolution layers.

The metric network: We model the similarity between features using three fully-connected layers with ReLU non-linearity. FC3 also uses Softmax. Input to the network is the concatenation of a pair of features. We output two values in

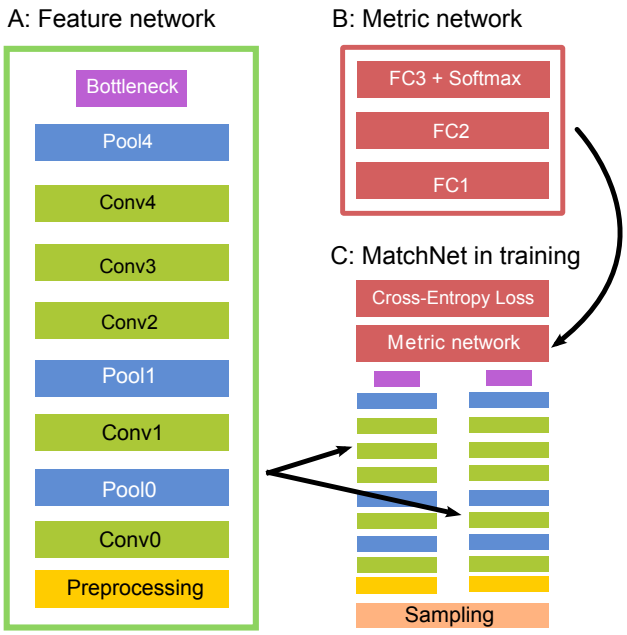

Figure 1. The MatchNet architecture. A: The feature network used for feature encoding, with an optional bottleneck layer to reduce feature dimension. B: The metric network used for feature comparison. C: In training, the feature net is applied as two "towers" on pairs of patches with shared parameters. Output from the two towers are concatenated as the metric network's input. The entire network is jointly trained on labeled patch-pairs generated from the sampler to minimize the cross-entropy loss. In prediction, the two sub-networks (A and B) are conveniently used in a two-stage pipeline (See Section 4.2).

$[0, 1]$ from the two units of FC3, These are non-negative, sum up to one, and can be interpreted as the network's estimate of probability that the two patches match and do not match, respectively.

Two-tower structure with tied parameters: The patch-based matching task usually assumes that patches go through the same feature encoding before computing a similarity. Therefore we need just one feature network. During training, this can be realized by employing two feature networks (or "towers") that connect to a comparison network, with the constraint that the two towers share the same parameters. Updates for either tower will be applied to the shared coefficients.

This approach is related to the Siamese network [2, 5], which also uses two towers, but with carefully designed loss functions instead of a learned metric network. A recent preprint on learning a network for stereo matching has also used the two-tower-plus-fully-connected comparison-network approach [37]. In contrast, MatchNet includes max-pooling layers to deal with scale changes that are not present in stereo reconstruction problems, and it also has more convolutional layers compared to [37].

Table 1. Layer parameters of MatchNet. The output dimension is given by (height $\times$ width $\times$ depth). PS: patch size for convolution and pooling layers; S: stride. Layer types: C: convolution, MP: max-pooling, FC: fully-connected. We always pad the convolution and pooling layers so the output height and width are those of the input divided by the stride. For FC layers, their size B and F are chosen from: $B \in \{64, 128, 256, 512\}$, $F \in \{128, 256, 512, 1024\}$. All convolution and FC layers use ReLU activation except for FC3, whose output is normalized with Softmax (Equation 2).

Name	Type	Output Dim.	PS	S
Conv0	C	$64 \times 64 \times 24$	7×7	1
Pool0	MP	$32 \times 32 \times 24$	3×3	2
Conv1	C	$32 \times 32 \times 64$	5×5	1
Pool1	MP	$16 \times 16 \times 64$	3×3	2
Conv2	C	$16 \times 16 \times 96$	3×3	1
Conv3	C	$16 \times 16 \times 96$	3×3	1
Conv4	C	$16 \times 16 \times 64$	3×3	1
Pool4	MP	$8 \times 8 \times 64$	3×3	2
Bottleneck	FC	B	-	-
FC1	FC	F	-	-
FC2	FC	F	-	-
FC3	FC	2	-	-

In other settings, where similarity is defined over patches from two significantly different domains, the MatchNet framework can be generalized to have two towers that share fewer layers or towers with different structures.

The bottleneck layer: The bottleneck layer can be used to reduce the dimension of the feature representation and to control overfitting of the network. It is a fully-connected layer of size B, between the 4096 ($8 \times 8 \times 64$) nodes in the output of Pool4 and the final output of the feature network. We evaluate how B affects matching performance in Section 5 and plot results in Figure 4.

The preprocessing layer: Following a previous convention, for each pixel in the input grayscale patch we normalize its intensity value x (in $[0, 255]$) to $(x - 128)/160$.

4. Training and prediction

The feature and metric networks are trained jointly in a supervised setting using a two-tower structure illustrated in Figure 1-C. We minimize the cross-entropy error

$$E = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \qquad (1)$$

over a training set of n patch pairs using stochastic gradient descent (SGD) with a batch size of 32. Here y_i is the 0/1 label for input pair x_i. 1 indicates match. $\hat{y}_i$ and $1 - \hat{y}_i$ are the Softmax activations computed on the values of the two

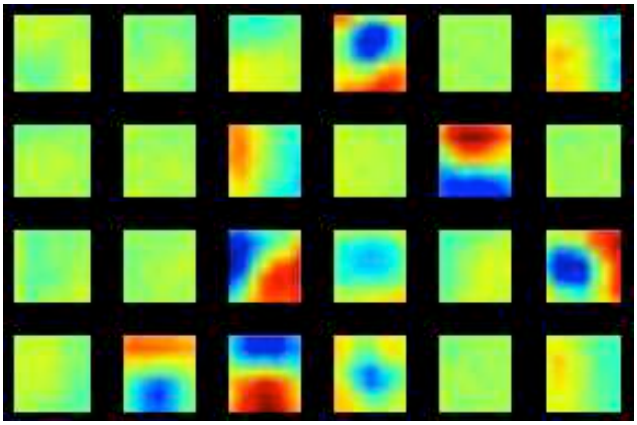

Figure 2. All 24 of the 7×7 filters learned in Conv0 from the liberty dataset. The pseudo-colors represent intensity.

nodes in FC3, $v_0(x_i)$ and $v_1(x_i)$ as follows.

$$\hat{y_i} = \frac{e^{v_1(x_i)}}{e^{v_0(x_i)} + e^{v_1(x_i)}}. \quad (2)$$

$\hat{y_i}$ is used as the probability estimate for label 1 in Equation 1.

We experimented with different learning rates and momentum values and found using plain SGD with 0.01 for the learning rate yields better validation accuracy than using larger learning rates and/or with momentum, even though convergence in the latter settings is faster. Depending on the network architecture, it takes between 18 hours to 1 week to train the full network. Using a learning rate schedule can speed up the training significantly.

Figure 2 visualizes Conv0 filters MatchNet learned on the Liberty dataset. Figure 5 visualizes the network's response to an example patch at different layers in the feature network.

4.1. Sampling in training

Sampling is important in training, as the matching (+) and non-matching (-) pairs are highly unbalanced. We use a sampler to generate equal number of positives and negatives in each mini-batch so that the network will not be overly biased towards negative decisions. The sampler also enforces variety to prevent overfitting to a limited negative set.

Particularly, in our setting, the training set has already been grouped into matching patches; e.g. The UBC patch dataset has an average group size around 3. The learner streams through the training set by reading one group at a time. For positive sampling, we randomly pick two from the group; for negative sampling, we use a reservoir sampler [32] with a buffer size of R patches. At any time T the buffer maintains R patches as if uniformly sampled from the patch stream up to T, allowing a variety of non-matching pairs to be generated efficiently. The buffer size controls the trade-off between memory and negative variety. In our experiments, $R = 128$ was too small and led to severe overfitting; $R = 16384$ has worked consistently. This procedure is detailed in Algorithm 1.

Algorithm 1 Generate a batch of 2S pairs with a sampler.

for $b = 0 \dots S-1$ **do**
  Extract all patches $p_1 \dots p_k$ from the next group;
  Randomly choose p_i and p_j, $i \neq j$, $i, j \in \{1 \dots k\}$;
  Sample$(2b) \leftarrow (1, p_i, p_j)$;
  for $m = 0 \dots k$ **do**
    Consider adding p_m to the reservoir;[1]
  end for
  repeat at most 1000 times
    Randomly draw p_u and p_v from the reservoir;
  until p_u and p_v are from different group;[2]
  if negative sampling is successful **then**
    Sample$(2b+1) \leftarrow (0, p_u, p_v)$;
  else
    Sample$(2b+1) \leftarrow (1, p_i, p_j)$;
  end if
end for
return Sample;

For instance, if the batch size is 32, in each training iteration we feed SGD 16 positives and 16 negatives. The positives are obtained by reading the next 16 groups from the database and randomly picking one pair in each group. Since we go through the whole dataset many times, even though we only pick one positive pair from each group in each pass, the network still gets good positive coverage, especially when the average group size is small. The 16 negatives are obtained by sampling two pairs from different groups from the reservoir buffer that stores previous loaded patches. At the first few iterations, the buffer would be empty or contain only matching patches. In that case we simply fill the slot in the batch with the most recent positive pair.

4.2. A two-stage prediction pipeline

A common scenario for patch-based matching is that there are sets of patches each extracted from two images. The goal is to compute a $N_1 \times N_2$ matrix of pairwise matching scores, where N_1 and N_2 are the number of patches in from image. Pushing each pair through the full network is not efficient because the feature tower would run on the same patch multiple times. Instead, we can use the feature tower and the metric network separately and in two

[1] Following [32], if the sampler's reservoir is not full, the candidate is always added; otherwise for the T-th candidate, with probability R/T it is added and replaces a random element in the reservoir and with probability 1-R/T it gets rejected. R is the reservoir size.

[2] We store meta data along with the patches in the buffer so it is efficient to check whether two patches match or not.

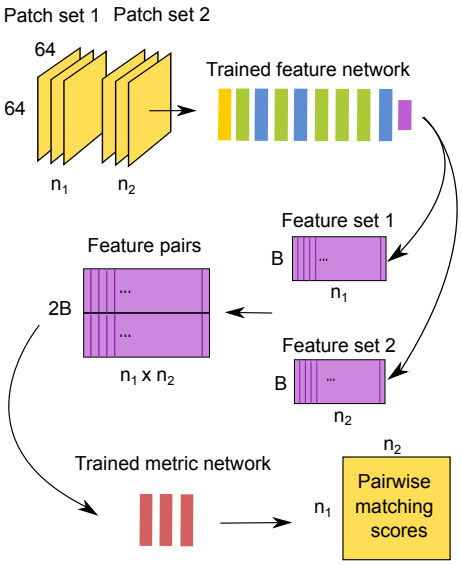

Figure 3. MatchNet is disassembled during prediction. The feature network and the metric network run in a pipeline.

stages (Figure 3). First we generate feature encodings for all patches. Then we pair the features and push them through the metric network to get the scores. In our experiment, on one NVIDIA K40 GPU, after tuning batch size, the feature net without bottleneck runs at 3.56K patch/sec; the metric net (B=128, F=512) runs at 416.6K pair/sec. The computation can be further pipelined and distributed for large-scale applications.

5. Experiments

Dataset. The UBC patch dataset [30] (UBC) was collected by Winder et al. [35] for learning descriptors. The patches were extracted around real interest points from several internet photo collections published in [24]. The dataset includes three subsets with a total of more than 1.5 million patches. It is suitable for discriminative descriptor or metric learning, and has been used as a standard benchmark dataset by many [3, 9, 22, 27, 28]. The dataset comes with patches extracted using either Difference of Gaussian (DoG) interest point detector or multi-scale Harris corner detector. We use the DoG set.

There are three subsets in UBC: Liberty, Notredame and Yosemite. Each comes with pre-generated labeled pairs of 100k, 200k and 500k, all with 50% matches. Each also provides all unique patches and their corresponding 3D point ID. The number of unique patches in each dataset is 450k for Liberty, 468k for Notredame and 634k for Yosemite.

Evaluation protocol. Following the standard protocol established in [3], people train the descriptor on one subset and test on the other two subsets. Although people may use any of the pre-generated pair sets or the grouped patches in the training subset for training and validation, the testing is done on the 100k labeled pairs in the test subset. The commonly used evaluation metric is the false positive rate at 95% recall (Error@95%), the lower the better.

SIFT baselines. We use VLFeat [31]'s `vlsift()` with default parameters and custom frame input to extract SIFT descriptor on patches. The frame center is the center of the patch at (32.5, 32.5). The scale is set to be 16/3, where 3 is the default magnifying coefficient, so that the bin size for the descriptor will be 16. With 4 bins along each side, the descriptor footprint covers the entire patch. In our preliminary experiments we found that normalized SIFT (nSIFT), which is raw SIFT scaled so its L2-norm is 1, gives slightly better performance than SIFT, so nSIFT is used for all our baseline experiments.

For a pair of nSIFT, we compute similarity using L2, linear SVM on 128d element-wise squared difference features (Squared diff.) and a two-layer fully-connected neural networks on 256d nSIFT concatenation (Concat.). For nSIFT Square diff.+ linearSVM, we use Liblinear [29] to train the SVM and search the regularization parameter C among $\{10^{-4}, 10^{-3} \ldots, 10^{4}\}$ using 10% of the training set for validation. For nSIFT Concat.+ NNet, the network has the same structure (with F=512) as the metric network in MatchNet (Figure 1-B) and is trained using plain SGD with learning_rate=0.01, batch_size=128 and iteration=150k.

MatchNet. We train MatchNet using techniques described in Section 4 and evaluate the performance under different (F, B) combinations, where F and B are the dimension of fully-connected layers (F1 and F2) and the bottleneck layer respectively. $F \in \{128, 256, 512, 1024\}$. $B \in \{64, 128, 256, 512\}$. We also evaluate using the feature network without the bottleneck layer.

MatchNet with quantized features. We evaluate the performance of MatchNet with quantized features. The output features of the bottleneck layer in the feature tower (Figure 1-A) are represented as floating point numbers. They are the outputs of ReLu units, thus the values are always non-negative. We quantize these feature values in a simplistic way. For a trained network, we compute the maximum value M for the features across all dimensions on a set of random patches in the training set. Then each element v in the feature is quantized as $q(v) = \min(2^n - 1, \lfloor (2^n - 1)v/M \rfloor)$, where n is the number of bits we quantize the feature to. When the feature is fed to the metric network, v is restored using $q(v)M/(2^n - 1)$. We evaluate the performance using different quantization levels.

The quantized features give us a very compact representation. The ReLU output of the bottleneck layer is not dense. For example, for the (B=64, F=1024) network, the average density over all the UBC data is 67.9%. Using a naive representation: 1 bit to encode whether the value is

Table 2. UBC matching results. Numbers are Error@95% in percentage. See Section 5 for descriptions of different settings. F and B are dimensions for fully-connected and bottleneck layers in Table 1. **Bold** numbers are the best results across all conditions. Underlined numbers are better than the previous state-of-the-art results with similar feature dimension.

Training		Notredame	Yosemite	Liberty	Yosemite	Liberty	Notredame
Test	Feature Dim.	Liberty		Notredame		Yosemite	
nSIFT + L2 (no training)	128d	29.84		22.53		27.29	
nSIFT squared diff. + linearSVM	128d	26.54	27.07	19.65	19.87	25.12	24.71
nSIFT concat. + NNet (F=512)	256d	20.44	22.23	14.35	14.84	21.41	20.65
Trzcinski et al (2012) [28]	64d	18.05	21.03	14.15	13.73	19.63	15.86
Brown et al (2011) [3] w/ PCA	29d	16.85	18.27	–	11.98	–	13.55
Simonyan et al (2014) [22] PR	<640d	16.56	17.32	9.88	9.49	11.89	11.11
Simonyan et al (2014) discrim. proj.	<80d	12.42	14.58	7.22	6.17	11.18	10.08
Simonyan et al (2014) discrim. proj.	<64d	12.88	14.82	7.52	7.11	11.63	10.54
MatchNet (F=1024, B=64)	64d	9.82	14.27	5.02	9.15	14.15	13.20
MatchNet (F=512, B=128)	128d	9.48	15.40	5.18	8.27	14.40	12.17
MatchNet (F=512, B=512)	512d	8.84	13.02	4.75	7.70	13.58	11.00
MatchNet (F=512, w/o bottleneck)	4096d	**6.90**	**10.77**	**3.87**	**5.67**	**10.88**	**8.39**

zero or not, quantizing the features to 6 bits yields a representation of $64 + 6 \times 64 \times 0.679 = 324.7$ bits on average. As discussed below, this compact representation yields similar performance achieved by state-of-the-art methods with 1024 bits. Of course, employing a more sophisticated encoding mechanism should further improve compactness.

Results. We follow the evaluation protocol and evaluate MatchNet along with several SIFT baselines and other learned float descriptors. Results for SIFT baselines and MatchNet with floating point features are listed in Table 2. Our best model 4096-512x512 (feature dim.-FxF) achieves best performance over all evaluation pairs. It achieves 7.75% average error rate vs. [22]'s <80f at 10.38%. With a bottleneck of 64d, our 64-1024×1024 model achieves 10.94% average error rate vs. [22]'s 10.75% using features with about the same dimension.

One advantage of our approach is that we can easily vary feature dimension and matching complexity and jointly optimize them. The trade off between storage/computation vs. accuracy is plotted in Figure 4. Not suprisingly, increasing F and B leads to lower error rate, but the absolute gain is diminishing exponentially.

The results for MatchNet with quantized features are presented in Table 3. We use the (B=64, F=1024) model and quantize the output of the feature tower into different number of bits. 6-bit quantization achieves an average error of 11.01% and with 3 train/test pairs even better than [22]'s <80f. The performance is almost exactly the same as [22]'s 1024 bit representation while using only 325 bits instead of 1024. For 6-bit quantization, the feature has 324.7 bits (40.6 bytes) per patch on average. Our results also show that going from a 32-bit floating point representation to a 6-bit one yields little degradation in performance. We attribute this to the robustness of the matching network.

Discussion. Our baseline experiments with SIFT features confirms that a better metric can significantly improve performance. For instance, in Yosemite-Liberty, nSIFT concat.+NNet performs better than nSIFT+L2 by 7.61% in absolute error rate, and MatchNet with the same feature dimension (128) and fully-connected layer dimension (512) achieves a further improvement of 6.7% in absolute error rate. The benefit of coupling the descriptor and the metric has been explored in different forms in the past. For instance, [22] learns weights for each pooling regions, so does [28] for each weak gradient map learner to form a Mahalanobis type of similarity. Our proposed unification through neural networks is a simple and powerful alternative.

Our best model is trained without a bottleneck and it learns a high-dimensional patch representation coupled with a discriminatively trained metric. It improves on the previous state-of-the-art performance across all train-test pairs by up to 3.4% in absolute error rate. Fig. 3 in Simonyan et al. [22] showed increasing feature dimension reduces the error rate. However this effect may not be enough to explain our ∼3% absolute gain for Liberty-Notredame. On one hand, the top curve in [22]'s Fig. 3 shows a diminishing gain. Without discriminative projection, at around 1500d, the error rate is still above 9%, more than twice as much as MatchNet's error rate (3.87%) with 4096d patch representation. On the other, with a 512d bottleneck and quantization, MatchNet still outperforms [22]'s PR (<640d) results in 4 out of 6 train-test pairs with up to 7% improvement in absolute error rate.

Supported by the tradeoff results in Figure 4 and Table 3, we provide the following guidelines to enable users to choose a configuration based on their storage/computation constraints: The 4096-512x512 model should be used if the feature dimension is not a concern, or if accuracy takes priority. This model outperforms others by a large margin on

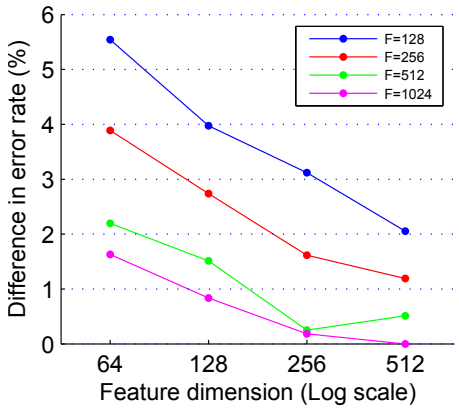

Figure 4. Accuracy vs. dimensionality tradeoff for different fully-connected layer sizes. we plot the average difference in Error@95% between other (F, B) combinations and $(F = 1024, B = 512)$ across all 6 train-test pairs in UBC. Features are unquantized.

Table 3. Accuracy vs. quantization tradeoff for the 64-1024×1024 network. In the first column, the first value is bits per dimension. The second value is average bits per feature vector. It is computed using $64 + 64 \times 0.679 \times b$, where b is the number of bits per dimension, and the average density (non-zeros) of the feature vector is 67.9%. Numbers in the middle are Error@95%. The first row is for the unquantized features.

	Notr.	Yos.	Lib.	Yos.	Lib.	Notr.
# of bits	Lib		Notr.		Yos.	
32 (1456)	9.82	14.27	5.02	9.15	14.15	13.20
8 (411.7)	9.84	14.33	5.06	9.21	14.17	13.21
7 (368.6)	9.82	14.20	5.04	9.23	14.21	13.19
6 (324.7)	9.81	14.22	5.15	9.30	14.27	13.29
5 (281.3)	10.19	14.58	5.33	9.59	14.66	13.39
4 (237.8)	11.37	15.27	6.27	10.93	15.59	14.07

UBC. If extra compression is needed, the 64-1024x1024 one with quantization should be used.

More sophisticated quantizations could enable a better accuracy vs. dimensionality tradeoff. Potential improvements include: (a) Using a per-dimension max value; (b) better zero encoding (currently we use 1 bit per dimension to indicate whether the dimension is 0); (c) better compression algorithm; and (d) Reduce the range from $[0, \text{MaxValue}]$ to $[0, 95\text{th percentile}]$.

6. Conclusion

We propose and evaluate a unified approach for patch-based image matching that jointly learns a deep convolutional neural network for local patch representation as well as a network for robust feature comparison. Our system trains models that achieve state-of-the-art performance on a standard dataset for patch matching. We also evaluate a suite of architectural variations to study the tradeoff between accuracy vs. storage/computation. This work demonstrates that deep convolutional neural networks can be effective for general wide-baseline patch matching. In addition, an important feature of these results is that MatchNet can produce state-of-the-art accuracies while using significantly fewer bits per feature even than very recent work on compact feature representations, even with very simple quantization. This suggests that using deep learning approaches—and more advanced quantization—can make even more significant improvements in the accuracy/feature size trade-off.

Acknowledgments

X. Han thanks NVIDIA for the GPU donation. A.C. Berg's was supported in part by NSF IIS:1452851 and a Google faculty research award.

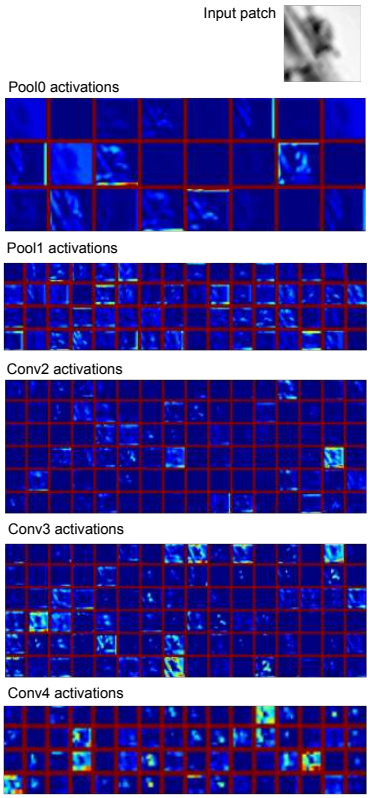

Figure 5. Visualization for the activations in response to an example input patch at different layers in the feature network. The input 64×64 patch is shown at the top. For each layer, we tile its K $H \times W$ activation maps to form a 2D image. H, W and K are the height, width and depth of the 3D activation array respectively. Red margins separates these tiles. Pseudo-colors in the tiles represent response intensity. Border artifacts may occur, but we keep our padding scheme, which retrains half of the information on the original border.

References

[1] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *ECCV*, 2006. 1

[2] J. Bromley, I. Guyon, Y. Lecun, E. Säckinger, and R. Shah. Signature verification using a "Siamese" time delay neural network. In *NIPS*, 1994. 3

[3] M. Brown, G. Hua, and S. A. J. Winder. Discriminative learning of local image descriptors. *IEEE TPAMI*, 33(1):43–57, 2011. 1, 2, 5, 6

[4] M. Brown and D. Lowe. Automatic panoramic image stitching using invariant features. *IJCV*, 74(1), 2007. 1

[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 2, 3

[6] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, 2014. 1

[7] J. Heinly, E. Dunn, and J.-M. Frahm. Comparative evaluation of binary features. In *ECCV*, 2012. 1

[8] P. Jain, B. Kulis, J. V. Davis, and I. S. Dhillon. Metric and kernel learning using a linear transformation. *Journal of Machine Learning Research*, 13:519–547, 2012. 1, 2

[9] Y. Jia and T. Darrell. Heavy-tailed distances for gradient based image descriptors. In *NIPS*, pages 397–405, 2011. 1, 2, 5

[10] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 1

[11] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *CVPR*, 2004. 1

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[13] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 1

[14] J. Matas and O. Chum. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10), 2004. 1

[15] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *TPAMI*, 27(10):1615–1630, 2005. 1

[16] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. J. V. Gool. A comparison of affine region detectors. *IJCV*, 65, 2005. 1

[17] N. Molton, A. Davison, and I. Reid. Locally planar patch features for real-time structure from motion. In *BMVC*, 2004. 1

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2011. 1

[19] R. Salakhutdinov and G. E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, 2007. 2

[20] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, 2006. 1

[21] G. Shakhnarovich. *Learning Task-Specific Similarity*. PhD thesis, MIT, 2006. 2

[22] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *TPAMI*, 2014. 1, 2, 5, 6

[23] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos, 2014. arXiv preprint arXiv:1406.2199. 1

[24] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008. 5

[25] C. Strecha, A. A. Bronstein, M. M. Bronstein, and P. Fua. LDAHash: Improved matching with smaller descriptors. *TPAMI*, 34(1):66–78, 2012. 2

[26] A. Toshev and C. Szegedy. DeepPose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 1

[27] T. Trzcinski, C. M. Christoudias, P. Fua, and V. Lepetit. Boosting binary keypoint descriptors. In *CVPR*, 2013. 1, 2, 5

[28] T. Trzcinski, C. M. Christoudias, V. Lepetit, and P. Fua. Learning image descriptors with the boosting-trick. In *NIPS*, pages 278–286, 2012. 1, 2, 5, 6

[29] http://www.csie.ntu.edu.tw/~cjlin/liblinear/. 5

[30] http://www.cs.ubc.ca/~mbrown/patchdata/patchdata.html. 5

[31] http://www.vlfeat.org/overview/sift.html. 5

[32] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985. 4

[33] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014. 2

[34] Y. Weiss, A. B. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008. 2

[35] S. A. J. Winder, G. Hua, and M. Brown. Picking the best DAISY. In *CVPR*, 2009. 5

[36] B. Yao, G. Bradski, and L. Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *CVPR*, 2012. 1

[37] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. *http://arxiv.org/abs/1409.4326*, 2014. 2, 3