NCC-Net: Normalized Cross Correlation Based Deep Matcher with Robustness to Illumination Variations

Arulkumar Subramaniam[*], Prashanth Balasubramanian[*], Anurag Mittal
Indian Institute of Technology Madras
Chennai, India 600036
{aruls, bprash, amittal}@cse.iitm.ac.in

Abstract

*The task of matching image patches is a fundamental problem in computer vision. When sufficiently textured patches are normalized up to similarity transformation, a simple Normalized Cross Correlation (*NCC*) of corresponding patches will give a high value. In practice, using it on patches per se may not perform well due to the noisy variations of pixel intensities. A more prudent approach will be to apply it to the abstract features extracted by a deep convolutional network. We study the applicability of an* NCC *based convolutional network for the task of Patch Matching. Further, there may be cases where the network may fail due to insufficient textures. In those cases, a simple pixel difference based method will be beneficial.*

To this end, we propose to improve the two basic architectures, Siamese networks and Central-Surround stream networks, using robust matching layers for learning the similarities of patches, assisted by a simple cross-entropy loss function. We empirically verify the performance of the proposed models on the challenging UBC Patches *dataset and show that they are close to the state-of-the-art. Further, we evaluate their resilience to large illumination changes in two experimental scenarios: 1) by manually varying the patches of* UBC Patches *by an affine model 2) by using the publicly available* Webcam *dataset. We demonstrate that our models are indeed very resilient to illumination variations; they reduce the false positive rate to nearly* 10%*, and improve over the popular methods by nearly* 5%*. Further, we demonstrate the generalisability of the proposed* NCC *based matching layer by applying it to Face Recognition and show that it improves the performances of well known networks on a real-world, surveillance dataset.*

1. Introduction and Related Work

Patch Matching is one of the most fundamental tasks of Computer Vision. It aims to find correspondences of localized, textured regions which are usually centered at distinctive keypoints. The matching has to be robust across many geometric and photometric changes as it is used as an indispensable subroutine in many problems such as Image Registration and Mosaicking[11], Stereo Matching[45], 3-D Reconstruction[1], and Object Tracking[44], to name a few.

There is a large body of work on Patch Matching, and it will be beyond the scope of this paper to survey all of them. However, they can be broadly classified into 3 categories, by virtue of their techniques: 1) those using hand-crafted descriptors, 2) those predicting the correspondences based on non-deep machine learning methods, and 3) those predicting based on deep neural networks.

Hand-crafted techniques : The aim, here, is to hand-design descriptors that are robust to photometric and geometric challenges. A seminal work in this direction is Lowe's *SIFT*[24], following which other methods have been proposed that improve either its computational efficiency[13, 40, 8, 31, 20] or its performance by alternative strategies [22, 31, 29, 4, 37, 17, 38].

Non-deep Machine Learning techniques : Though impressive, hand-crafted descriptors can be limited in their applicability when the challenges become generic and unpredictable, especially with the availability of large sets of images taken in differing conditions. Learning to match the patches is an attractive and a natural alternative to it. A notable attempt was made by Brown et al.[10] who learned robust descriptors that performed much better than *SIFT*. A large dataset of patches[12] was also made available by them. *Trzcinski* et al.[38] learned discriminative descriptors based on boosting, while *Simonyan* et al.[33] proposed convex learning techniques to build robust descriptors.

Deep-Learning techniques : An appreciable attempt to use *CNNs* for Patch Matching was made by *Zagoruyko and Komodakis*[43] who studied various architectures to directly learn the underlying matching function from the pixels. *Serra* et al.[32] proposed the *MatchNet*, a Siamese network with sparse connections between the layers such that the

[*]equal contribution

mean number of connections of an input layer is nearly constant. *Zbontar* and *LeCun*[45] designed a CNN-based stereo matcher for small baselines and obtained the best results on the KITTI dataset. *Balntas* et al.[5, 7] proposed a CNN using a triplet-based loss function. They aim to increase the speed of descriptor computation and decrease its memory footprint. *Kumar* et al.[21] also propose a global loss that aims to decrease the intra-class variances and increase the inter-class distance; here, the similar and dissimilar pairs constitute the binary classes.

In our work, we make use of normalized cross-correlation(*NCC*) which is a statistical technique to measure how two signals vary together. When the patches are scale and orientation normalized, the features extracted after successive convolutions and pooling may have little noise in them; hence, it will be prudent to apply *NCC* on these feature maps as a measure of similarity between the input images. An earlier work on Person Re-identification[35] has shown state-of-the-art results using *NCC* based networks. Further, in cases where it may fail (viz. an image with insufficient textures), an additional matching layer[2] that is based on simple pixel differences is used.

We propose two deep neural network architectures for the task of Patch Matching, one based on Siamese networks and the other on Central-Surround stream networks. In these models, the two aforesaid matching layers are combined and are trained with a simple, no-frills cross-entropy loss function. We verify their performance on the challenging *UBC Patches* dataset and show that they remain close to the performance of the current state-of-the-art, L2$-$Net[36], while improving over the others [21, 32, 42, 43]. Further, we evaluate their resilience to large illumination changes by empirically demonstrating on two experimental setups: $1)$ manual variation of the pixel intensities of the patches in the *UBC Patches* dataset. $2)$ natural illumination changes from the real-world, publicly available *Webcam* dataset. Our models achieve an error rate as low as 10%, improve over the performances of popular patch matchers[43, 32, 21] by as much as 5% and achieves near-state-of-the-art performance.

Additionally, to demonstrate the generalization ability of the proposed *NCC* based matching layer, we test it in the task of Face Recognition. We observe an increase in performance when the matching layer is used on the feature maps from well-known Face Description networks[3, 30] on a real-world, surveillance dataset[16].

The paper is organized as follows: Section 2 describes the two proposed architectures that use robust matching layers to estimate the similarities of patches. Section 3 outlines the three experimental setups, the datasets and their results, followed by detailed analyses. Section 4 describes the applicability of the proposed NCC based matching layer to the task of face recognition which extends the scope of its utility. Section 5 concludes the paper by summarizing it and providing possible future directions to it.

2. Methodology

In this section, we propose and discuss two *CNN*-based architectures that learn to predict the similarity of a given pair of image patches. The task is formulated as an instance of a $2-$way classification problem, the classes being similar and dissimilar pairs.

2.1. Brief description of *NCC*

Normalized Cross-Correlation(*NCC*) is a statistical measure of the tendency of two signals to vary linearly with each other. It is given by

$$NCC(X,Y) = \frac{1}{N-1}\sum_{i=1}^{N}\frac{(X_i-\mu_X)(Y_i-\mu_Y)}{\sigma_X\sigma_Y} \qquad (1)$$

where N is the number of samples, (μ_X,μ_Y) and (σ_X,σ_Y) denote the means and the unbiased standard deviations of signals X and Y.[1] It ranges in between $[-1,1]$ such that when X and Y are linearly correlated its absolute value is large, and small when they are uncorrelated. Thus, when the pixels of two input patches vary in a similar manner, their *NCC* is large. As the patches under consideration are usually small, typically 64×64 pixels, their pixel variations can be taken to be uniform, and hence *NCC* is used as a measure of similarity.

While *NCC* can be applied directly on the pixel values, it can also be used on the features extracted from them. The latter is preferred as it remains robust to the pixel-level artifacts that occur, say, image noise. So, when included as a layer in a deep network, it operates on its feature map inputs, computes their *NCC* coefficients, and passes them to the neurons further downstream in the architecture. In such a case, the *NCC* layer acts as a “natural matcher", implicitly measuring the similarity of the input patches. Further, it is differentiable with respect to its inputs[35] and hence can back-propagate the gradients.

2.2. Robust Matching layers

***NCC* for Patch Matching :** Given two feature maps, *NCC* can be used to measure their similarity. Nevertheless, it will be more beneficial to study the similarities of the various smaller regions(termed as *support regions*) that constitute the feature maps. Such an approach helps the network to learn from the correlations between the finer features that reside in the support regions. For example, the finer features may be corners, junction points, strong edges or ramps which usually serve as distinguishing characteristics.

Let P, Q denote the input feature maps of the *NCC* layer. A support region defines a spatial extent whose pixel-values

[1]In practice, a small $\epsilon = 0.01$ is added to the standard deviations for numerical stability.

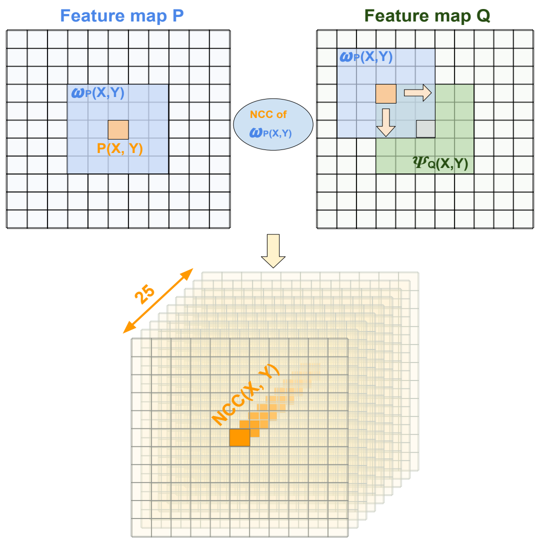

Figure 1: Illustrations of support regions. Using *NCC*, a support region $\omega_{P(x,y)}$ on the left is compared with multiple, neighboring support regions $\omega_{Q(.,.)}$ on the right. Each $\omega_{Q(.,.)}$, on the right, is centered at a green pixel. Comparing a support region with multiple neighboring regions, as shown here, helps the network to learn the patterns from a large neighborhood. The *NCC* between P and Q results in an *NCC* feature map, shown in the bottom row.

are considered for the computation of an *NCC* score. Thus, the support region, $\omega_P(x,y)$, centered at (x,y) in P is defined as

$$\begin{aligned}\omega_P(x,y) = \{&x-2,\ldots,x+2\} \\ &\times\{y-2,\ldots,y+2\}\end{aligned} \tag{2}$$

where $x \in \{1,\ldots,\mathrm{W}\}$ and $y \in \{1,\ldots,\mathrm{H}\}$ vary over the width W and height H of feature map P ($0-$ padding inherently assumed).

Further, it may be unnecessary to compare, exhaustively, all the support regions of a feature map with those of the other. Apart from its computational expense, such a comparison is also redundant as distant region-pairs need not be corresponding to each other. Thus, a support region from the first feature map, P, is compared only with its neighboring support regions from the second feature map, Q.

The support regions from the second feature map Q which are considered for computing the *NCC*s with the values at $\omega_P(x,y)$ are formed from a small $t_x \times t_y$ neighborhood of (x,y). Let $\Psi_Q(x,y)$ denote such support regions in Q. Then,

$$\Psi_Q(x,y) = \{\omega_Q(x+\delta_x, y+\delta_y)\} \tag{3}$$

where $\delta_x \in \{-t_x, -t_x+1,\ldots,t_x\}$, $\delta_y \in \{-t_y, -t_y+1,\ldots,t_y\}$. t_x and t_y are hyper-parameters. Thus $\Psi_Q(x,y)$ defines the set of support regions from map Q, each of which will be used to compute an *NCC* score with $\omega_P(x,y)$ from map P. Figure 1 pictorially depicts $\omega_P(x,y)$ and $\Psi_Q(x,y)$. The values of t_x, t_y have to be chosen with care. Essentially, $\Psi_Q(x,y)$ can be viewed as the "search space" to which the features resident in $\omega_P(x,y)$ could have moved under a variation. Too small a choice for t_x, t_y, the features may be missed, and too wide a choice, the search can become computationally burdensome. In our experiments, we found $t_x, t_y = 2$ to work well.

The choice of considering a square search space (i.e. $t_x = t_y$) is motivated by the problem under consideration. In Patch Matching, it is assumed that a feature may move in any direction, and no preference to any particular direction can be ascribed. This can also be seen in many classical works which consider a square ([24, 8]), or more generally, an isotropic(i.e. a circular)([26, 9]) region to build their descriptors.

With the above settings, every support region $\omega_P(x,y)$ is compared with 25 of their neighboring support regions $\omega_{Q(.,.)}$ and hence yield 25 *NCC* values, see Figure 1. By repeating this for every (x,y) of P, all of the 25 NCC values are concatenated in a specific order to yield 25 "NCC" feature maps of the same width and height as P and Q.

***CIN* for Patch Matching :** Although the patches are expected to have sufficient textures, there may arise some pathological cases which lack them. In such cases, *NCC* is unreliable as σ_X or σ_Y vanishes . As a remedy, it is fused with the Cross-input Neighborhood(*CIN*)[2] layer. *CIN* builds "difference maps" between every pixel of a feature map and a neighborhood window (5×5) of its corresponding feature map. The idea is to help the network learn discriminative features from absolute pixel[2] differences.

2.3. Proposed architectures for Patch Matching

In this section, we present two deep neural network architectures for Patch Matching. Both these architectures include the matching layers discussed above for learning the similarity between the patches. They differ in the way the input patches are handled.

Siamese architecture : Given two image patches each of size $64\times64\times1$, a Siamese architecture seems to be a natural choice to compare them. The proposed Siamese model is shown in the Figure 2. The network consists of two input branches of shared weights which accept the pair of image patches to be matched. The image patches are passed through two convolution layers, C$(32,5,1)$ and C$(96,5,1)$, having 32 and 96 feature maps, each followed by a ReLU activation and a max-pooling layer, M $(2,2)$. For an explanation of the notations and network architecture, refer Figure 2.

The resulting pair of abstract features after M$(2,2)$ of size $13\times13\times96$ are passed through the matching layers. In the *NCC* layer, each of the 96 feature maps of the top branch is compared with its counterpart from the lower branch as described in Section 2.2.

[2]Here, a pixel refers to a location on a feature map.

This yields 2400 feature maps (96 input feature maps $\times$ 25 NCC maps per input feature map), each of size 13×13. In a similar way, CIN comparison yields 4800 feature maps (96 input feature maps $\times$ 25 CIN maps per input feature map $\times$ 2 (being asymmetric)) each of size 13×13. The outputs from the matching layers are passed through two more convolution layers followed by a max-pooling layer for summarization. After summarization, the responses are passed through a fully connected layer of 500 neurons each and followed by a softmax classification layer with 2 nodes which predict whether the given patches are similar or not. We call this model as **Siam-NCC-Net**.

As the inclusion of 2 max-pooling layers reduces the input dimensions by 4, some fine textures present in the feature maps may be lost in this process. To avoid this, we propose a minor variant of **Siam-NCC-Net** in which the second max-pooling layer, $M_2(2,2)$ is discarded. We call this modified network as **Siam-w/oMP$_2$-NCC-Net**. Its architecture diagram is provided in the supplementary material.

2-Stream (or) Central-Surround (CS) stream architecture : A patch which is extracted from an image is usually localized around a keypoint and provides characteristic information about it. Thus, the pixels near the keypoint convey properties of the patch which are more specific and reliable than those from the distant ones. In the classical methods such as SIFT[24], this is modeled by weighting the pixels with an appropriate Gaussian whose mean is at the center of the patch. *Zagoruyko and Komodakis*[43] model this by supplying two kinds of inputs to the network: the first crops the input patch-pairs around their centers to the size 32×32, and the second downsamples them to 32×32. Such a model which explicitly matches the central portion of the patches was shown to improve the baselines.

In a similar spirit, our second proposed architecture contains two parallel Siamese networks; one explicitly matches the central portion of the image patches (of size 32×32), the other matches the downsampled patch-pairs of size 32×32. The network architecture is depicted in the figure 3. We call this network as **CS-NCC-Net**. Further, we obtain a minor variant of **CS-NCC-Net** by discarding $M_2(2,2)$ to preserve finer textures in the feature maps. We term this model as **CS-w/oMP$_2$-NCC-Net** and its network diagram is provided in the supplementary version.

3. Datasets, Experiments and Results

UBC Patches dataset : This dataset[12] has been introduced by *Brown* et al.[10] wherein the keypoint correspondences have been identified with multi-view, stereo constraints. The dataset contains 3 subsets - *Liberty, Notredame, Yosemite*, each of which contains $500,000$ similar and dissimilar pairs of training patches and $100,000$ pairs of testing patches. We followed the standard procedure [10] of training on a $500K$ set and testing on the other two $100K$ sets.

Training : The $500K$ pairs were randomly split in a $90:10$ ratio of training and validation sets. The $50K$ validation data was used for early stopping, to avoid over-fitting. The models were trained using mini-batch Stochastic Gradient Descent(SGD) with a batch size of 128. The standard cross-entropy loss given by

$$L = -\frac{1}{N}\sum_{i=0}^{N}\left(t_i \log p_i + (1-t_i)\log(1-p_i)\right) \quad (4)$$

was used as the objective function. Here, N is the batch size, $t_i \in \{0,1\}$ specifies the target (same=0 or different=1) and p_i denotes the 'same' probability given by softmax,

$$p_i = \frac{e^{a_{i0}}}{e^{a_{i0}} + e^{a_{i1}}} \quad (5)$$

where a_{ij} is the logit of the neuron j in the final layer for example i of the batch. The initial learning rate was set as 0.05, its decay as 10^{-4}, momentum as 0.9 and the weight decay as 5×10^{-4}. Weights were initialized randomly and the models were trained anew. The implementation was done in Torch[14] and is available online[34]. The training was done on NIVIDA Titan GPUs, with the code spawning threads across multiple GPUs to speed up the training phase. The training was conducted for nearly 60 epochs. The best performing model (Siam-NCC-Net) has ~3.3M parameters and takes ~2.9 ms for matching a pair of image patches.

Data augmentation : Each pair underwent one of the selected random transformations during training. The transformations included rotations by 90°, 180°, 270°, flipping horizontally and vertically. This augmentation strategy is also followed in [21, 43].

Evaluation Protocol : We measure the performance of the methods by counting their false positives and true positives, and expressing them on the *ROC* graph(false positive rate(*FPR*) vs. true positive rate(*TPR*)). We consider the probability score given by the "same" softmax neuron in the final layer as the similarity score for the given pairs. An acceptable operating region on this plot is when the *TPR* ≥ 0.95 (at least 95% of the correspondences are successfully matched), and we report the *FPR* when *TPR* $= 0.95$. This is called as the *FPR95* score; lower the score, better the matcher.

3.1. Results on *UBC Patches*

Table 1 shows the comparative performances of various methods on the challenging *UBC Patches* dataset. We observe that our proposed models are close to the state-of-the-art[36] in many of the cases while being better than the popular, oft-used or recent works such as [21, 32, 42, 42]. The learning outcomes are as follows.

Firstly, the Siamese networks in the literature[43, 21] have generally been below par when compared with the CS-stream architectures, probably due to the sub-optimal

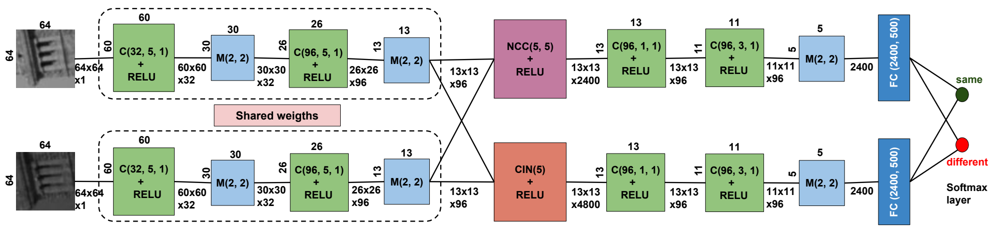

Figure 2: Siam-NCC-Net architecture. Here, C(N,m,s) refers to a Convolutional layer consisting of N filters, each of size $m \times m$ and a stride of s pixels, M(n,s) refers to a max-pooling layer that operates on a window of size $n \times n$ and a stride of s pixels, NCC(n, m) refers to an *NCC* layer that matches a support region of size $n \times n$ centered at any pixel with its $m \times m$ search space, CIN(m) refers to an *CIN* layer with a search space of size $m \times m$, FC(n, m) refers to a fully-connected layer with n inputs and m outputs. The diagram is best viewed in color on a display device.

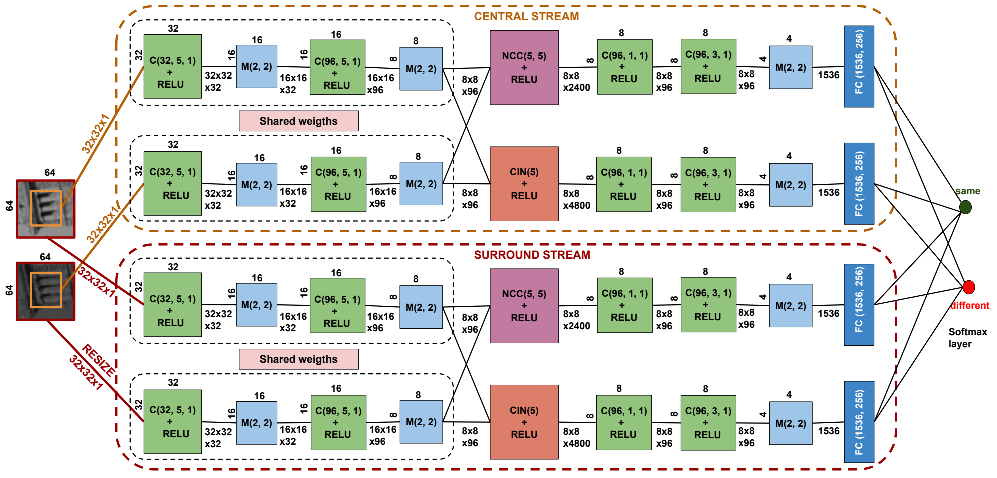

Figure 3: CS-NCC-Net architecture with *NCC* and *CIN* matching layers. The notations are same as those in Figure 2. (Best viewed in color on a display device.)

learning of the underlying matching function (compare *Siam vs. Siam CS-Stream* in Table 1). From the results in Table 1, we observe that **Siam-NCC-Net** improves over *Siam* and *Siamese GLoss* by about 7% and 3% in the mean error respectively. Further, it performs better than the CS stream counterparts(*2ch-CS stream GLoss, Siam CS-stream*) in many cases and on par with them in the others. This reveals that robust matching techniques in conjunction with the Siamese networks empower them to perform better than their vanilla versions.

Secondly, the said observation may suggest that the matching layers can also be combined with the CS-stream architectures to improve their performance. Nonetheless, our experimental results suggest that this may not be so straightforward. A direct adaptation of **Siam-NCC-Net** as a CS stream (**CS-NCC-Net**) did not yield a competitive performance. In fact, it increased the mean error by nearly 1.2% (see **CS-NCC-Net** in Table 1). This could be due to the low-resolution (8×8) feature maps yielded by $M_2(2, 2)$ which may not contain sufficient textures for the matching layers to cue upon. To overcome this problem, **CS-w/oMP$_2$-NCC-Net** outputs larger maps (16×16) by eliminating $M_2(2, 2)$. This reduces the mean error by about 1% (see in Table 1). This asserts that it is important to retain sufficiently large maps for the matching layers to learn from. In a similar spirit, we eliminated $M_2(2, 2)$ from **Siam-NCC-Net** to increase the feature map resolutions. We observed that its performance remains similar to that of **Siam-NCC-Net**, although it helps significantly in overcoming illumination variations, refer Section 3.3.

3.2. Performance on HPatches dataset

We compare the performance of our best performing model(**Siam-w/oMP$_2$-NCC-Net**) with the recent deep-learning based works in the literature using the HPatches dataset[6]. The quantitative results for the dataset can be found in the supplementary material and our source code repository[34].

3.3. Robustness to illumination changes

Illumination changes are a common occurrence and yet remain challenging for a patch matcher. Common techniques to counter this include mean subtraction as seen in some deep network based methods[43]. Such a simple scheme can, nevertheless, be improved with *NCC*. In order to study the effect of illumination changes, we carried out experiments specifi-

Train dataset	Liberty		Notredame		Yosemite		mean
Test dataset	Notredame	Yosemite	Liberty	Yosemite	Liberty	Notredame	
Siam-NCC-Net(ours)	1.25	**2.03**	3.87	**1.86**	5.16	1.8	**2.66**
Siam-w/oMP$_2$-NCC-Net(ours)	1.14	2.30	4.02	2.34	**4.71**	1.81	2.72
CS-NCC-Net(ours)	1.24	3.09	5.99	4.22	6.54	2.06	3.86
CS-w/oMP$_2$-NCC-Net(ours)	1.17	2.19	4.28	2.30	4.81	1.7	2.74
L2-Net [36]	**0.56**	**2.07**	**1.71**	**1.76**	**3.87**	**1.09**	**1.84**
DeepCD [42]	2.59	7.03	5.85	6.69	7.82	2.95	5.49
2ch-CS stream GLoss [21]	**0.77**	3.09	**3.69**	2.67	4.91	**1.14**	2.71
2ch-CS stream[43]	1.9	4.75	4.55	4.1	7.2	2.11	4.10
Siamese GLoss[21]	1.84	6.61	6.39	5.57	8.43	2.83	5.28
TFeat [7]	3.12	7.82	7.22	7.08	9.79	3.85	6.48
PNNet[5]	3.71	8.99	8.13	7.1	9.65	4.23	6.97
Siam CS-stream[43]	3.05	9.02	6.45	10.45	11.51	5.29	7.63
MatchNet[32]	4.75	13.58	8.84	11.00	13.02	7.7	9.81
Siam[43]	4.33	14.89	8.77	13.23	13.48	5.75	10.07
VGG-Convex[33]	7.52	11.63	12.88	10.54	14.82	7.11	10.75

Table 1: *FPR95* scores of the proposed models and the baselines. These being False Positive Rates (FPR), lower their values, better is their performance. Testbed: *UBC Patches* dataset[10]. Color coding : **best**, **second best**.

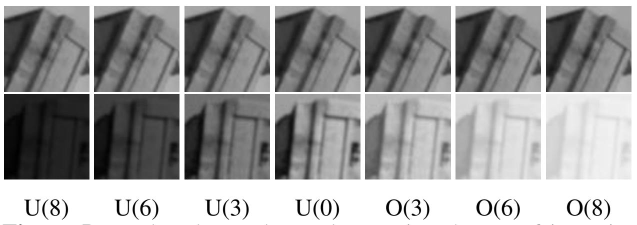

Figure 5: Each column shows the varying degree of intensity change introduced to a corresponding pair from *UBC Patches* dataset[12]. Notations as in Figure6.

cally to validate the models when such changes occur. We devise two experiments, one by manually varying the intensities of the images from the *UBC Patches* dataset[10] by an affine model; the other by using the *Webcam* dataset[39, 19] which captures various outdoor scenes at different times of day and seasons, thus exhibiting illumination variations in a natural way.

Manual variation of intensities : To emulate the illumination variations manually, we vary the pixel intensities of one patch in every pair of the $100K$ test set[12] in such a way that each pixel progressively gets either under- or over-saturated. We formulate an illumination model as follows: The modified pixel value, $I_i(x, y)$, at the step i is given by:

$$I_i(x,y) = I(x,y) + \left(i * \frac{(\mathbb{E} - I(x,y))}{N} \right) \tag{6}$$

Here $N(= 10)$ denotes the number of steps towards saturation with $0 \leq i \leq N$, $I(x, y) \in \{0, \ldots, 255\}$ is the input pixel value, $\mathbb{E} \in \{0, 255\}$ is a saturation point for the intensities. Setting $\mathbb{E} = 0$ under-saturates the pixels whereas having $\mathbb{E} = 255$ over-saturates them, see Figure 5. Equation 6 can be viewed as an affine model such that

$$I_i(x,y) = A.I(x,y) + B, \text{ where } \quad A = \frac{N-i}{N}, B = \frac{i\mathbb{E}}{N} \tag{7}$$

In Figure 6, the matching performance of the methods is plotted against the change in pixel's intensities. *FPR95* is used as the metric. We notice that all the methods gradually worsen with an increase in the saturation of the intensities. *SIFT*, being hand-engineered, is uniformly lower than other CNN-based methods but remains fairly invariant to illumination changes; the additive term B is eliminated during its gradient-computation step and the multiplicative term A is compensated for in its unit normalization step.

The impact of the intensity changes seems to be severe on the *2ch-CS stream GLoss* [21] which sharply degrades outside of $[U(5), O(5)]$. The reason could be that the global loss training has unintentionally caused the network to overfit for well-lit patches and so it fails to generalize when the intensities of the pixels are varied. *2ch-CS stream* [43] begins to degrade outside of $[U(4), O(4)]$. Ours and the recently proposed L_2-Net [36] seem to be more resilient than the others throughout the illumination axis. This simple experiment helps to demonstrate the effectiveness of the proposed models when strong illumination variations occur. The plots for other combinations of training and test sets are available in the supplementary material.

Test under natural illumination changes : The *Webcam* dataset has outdoor images collected from web cameras that remain fixed in several locations for different times of day and various seasons[39, 19]. The images are timestamped for

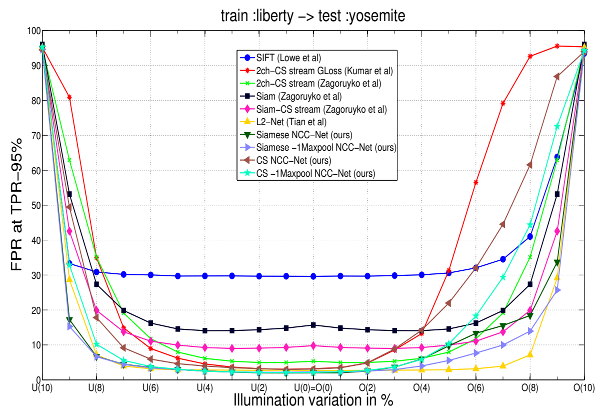

Figure 6: Impact of changing pixel intensities on the matching performance. Here the illumination change is induced by the models, $U(i) : I_i(x,y) = \frac{(N-i)*I(x,y)}{N}, O(i) : I_i(x,y) = \frac{(N-i)*I(x,y)}{N} + \frac{i\mathbb{E}}{N}$. Here, $U =$ under-saturation region, $O =$ over-saturation region. Testbed: *UBC Patches* dataset[10].

a period of 100 days at 6 locations. The dataset exhibits natural illumination changes (morning, evening, sunny, cloudy, snowy etc.,). We extract a total $262,152$ pairs of similar and dissimilar patches (each constitutes 50% of the total). The procedure that we followed to curate the test data is described, in detail, in the supplementary material. The dataset can be obtained from our source code repository[34].

The models trained on the *UBC Patches* dataset [12] have been used to test the *Webcam* patches, without any re-training or fine-tuning. The performance is evaluated in terms of *FPR95* and is shown in Table 2. There are a few observations from these results which we discuss below.

Firstly, the performances of the proposed *CS-stream* networks, as seen in Figure 6 and Table 2, are generally lower than that of their Siamese counterparts. This could be due to the fact that the *CS-stream* networks use images that are either at lower resolution(surround stream) or that lack context(central stream), both of which can get affected during illumination variations.

Secondly, the proposed Siamese models seem to cope up with the challenges fairly well. We observe from Table 2 that **Siam-NCC-Net** improves over *2ch-CS-stream GLoss* and *2ch-CS stream* by about 3% to 4%, while **Siam-w/oMP$_2$NCC-Net** improves by about 5%. It is interesting to note that these models are tolerant even when extreme changes in illumination occur, for example see U(8) and O(8) of Figure 5. At these points on the plots (Figure 6), we notice that *FPR95* of **Siam-w/oMP$_2$NCC-Net** is about 10%, whereas many of the other methods' seem to be $\geq 20\%$. This is due to the ability of the proposed matchers to extract and match the textures even at large illumination conditions. Some example visualizations of images at different illumination levels are shown in the supplementary material. Further, L_2-Net[36], seems to be doing well on this dataset, although there is no consistent winner(between the proposed models and L_2-Net across the 3 datasets. This merits a further exploration.

The effectiveness of the proposed matching layers is readily apparent when the performances of the vanilla Siamese networks (*Siam, Siam-CS stream*) are taken note of. Although the latter models have performed reasonably well in the *UBC Patches* dataset (refer Table 1), their performances have degraded significantly when strong illumination changes are present. This invites further exploration on improving the generalization capability of these networks. We also observe from the Figure 6 and Table 2 that eliminating $M_2(2,2)$ generally helps in improving the tolerance to illumination changes. During large illumination variations, the pixels of the feature maps could saturate unevenly. Applying max-pool on regions around the saturated pixels will only return saturated values. This could potentially eliminate some finer textures in its vicinity which may be useful in matching. Hence, discarding a max-pool layer helps in preserving them during illumination changes. Overall in our analysis, *Siamese* architectures seem to be doing better than the *CS-stream* architectures, especially when the illuminations changes are significant.

Ablation study: We further tested the performance of the individual matching layers(*NCC* and *CIN* alone) on the *UBC Patches* dataset[10] and summarize the scores in Table3. We notice that *NCC* performs nearly as good as the combination while *CIN*'s individual false alarm rates are much higher.

We further tested if augmenting the training set(apart from geometric augmentation) by explicitly modifying the illumination of the patches would help the networks be more resilient to such changes. One of every pair of input training patches of the *UBC Patches* was randomly over-saturated or under-saturated by the affine model in Equation 6. We retrained the networks proposed by *Zagoruyko and Komodakis*[43] on this augmented set and the performances are noted in Table 4. As expected, many of the scores have improved after being retrained on the augmented set. Nevertheless, such an augmentation may be seen more as an ad-hoc than as a robust approach, especially in a real-world setting wherein the changes due to illumination can tend to be more complex than being an affine modification.

4. Application to Face-Recognition

Face Recognition is the task of finding the right identity of the given probe image from a set of gallery images. Conventionally, discriminative features such as LBP and HOG were used to learn a discriminative metric between identities. Due to recent innovations in Deep Learning models, the usage of deep image descriptors [30, 3] has become prevalent in Face Recognition. Although the deep learning models exhibit a super-human level performance in datasets such as LFW[18], YTF[41], Celeb-A[23], the practical application

Train	**Siam-NCC-Net(ours)**	**Siam-w/oMP$_2$-NCC-Net(ours)**	**CS-NCC-Net(ours)**	**CS-w/oMP$_2$-NCC-Net(ours)**	2ch-CS-stream GLoss [21]	2ch-CS stream[43]	Siam[43]	Siam-CS stream[43]	L_2 Net
L	9.67	**9.45**	11.37	11.35	12.31	12.31	31.45	27.08	9.67
N	16.68	12.56	23.04	19.30	20.01	17.84	28.21	32.17	**9.21**
Y	11.67	**10.56**	15.40	18.28	14.76	19.5	35.21	34.65	16.11

Table 2: Results on the Webcam dataset[39, 19]. *FPR95* scores of the proposed models and the baselines[43, 21]. These being False Positive Rates (FPR), lower their values, better are their performances. Datasets trained from: **L**-Liberty, **N**-Notredame, **Y**-Yosemite. Scores obtained from $262,152$ test patch-pairs.

Train	L		N		Y	
Test	N	Y	L	Y	L	N
NCC	1.23	1.78	4.17	2.24	4.98	2.00
CIN	1.87	5.40	5.27	4.55	7.34	3.11
Both	1.14	2.30	4.02	2.34	4.71	1.81

Table 3: *FPR95* scores when using the individual matching layers in **Siam-w/oMP$_2$-NCC-Net**. These being False Positive Rates (FPR), lower their values, better is their performance. Testbed: *UBC Patches* dataset[10]. L - Liberty, N - Notredame, Y - Yosemite

Train	L		N		Y	
Test	N	Y	L	Y	L	N
2ch-CS stream+	1.82	3.73	2.85	2.56	5.99	1.34
Siam CS-stream+	3.56	9.29	6.46	9.56	11.53	5.47
Siam+	6.59	11.92	7.98	12.07	13.43	8.36

Table 4: *FPR95* scores of *DeepCompare*[43] networks retrained on augmented *UBC Patches* dataset[10]. These being False Positive Rates (FPR), lower their values, better is their performance. L - Liberty, N - Notredame, Y - Yosemite + refers to results after data augmentation.

of these models (for example, in surveillance settings) are in question considering the challenges of illumination changes, pose, appearance changes[25, 15]. The proposed NCC based matching layer in our approach is designed to be inherently tolerant to illumination changes. Hence, the addition of such an explicit matching scheme might help to overcome the effect of illumination challenges.

Face Recognition can be viewed as the matching of local parts (eyes, nose, mouth etc) using which the similarity at a global level is determined. Hence, the combination of the proposed matching layer which identifies similar regions through a search in a local neighborhood and a convolutional framework which learns a global context seems to be aptly applicable for Face Recognition.

SCface Dataset[16]: The dataset contains face images of 130 subjects that are captured by 5 surveillance cameras placed at different distances. A few sample gallery and probe images are given in supplementary material due to space constraints. Following the protocol from [27, 28], we randomly split the dataset into 50 subjects for fine-tuning the deep learning models and the remaining 80 subjects for testing; hence, there is no overlap between the training and testing subjects. The procedure to combine the proposed *NCC* based matching layer with *VGGFace* and *OpenFace* is described in the supplementary material in detail. The addition of *NCC* layer with the popular methods increases their performances. While the performance of *VGGFace* improves marginally, it improves that of *OpenFace*'s significantly.

Method	Rank-1 (%)
VGGFace+NCC	82.75
OpenFace+NCC	80.5
VGGFace(FC7)[30]	82.5
OpenFace[3]	7.5
DictAlignFR[28]	73.25
LowResFR[27]	69.45

Table 5: *Rank-1* performance of various methods in SCface database

5. Conclusion

In this paper, we have demonstrated how a robust matching technique such as Normalized Cross Correlation can be coupled with *CNN* based architectures to learn a better patch-matcher. Two architectures were proposed which were trained by a simple cross-entropy measure. The proposed models yield good performance on the challenging *UBC Patches* dataset[12]. We also showed how the models are robust to large illumination changes and Face Recognition. Experiments on two illumination-based tests showed that the proposed models perform well even when significant illumination changes occur. It will be interesting to propose a method of extracting descriptors for the given patches by distilling the knowledge contained in the proposed models. We also intend to study the applicability of the proposed patch-matchers in high-level tasks such as 3-D Reconstruction, Tracking, and Recognition.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Commun.*

ACM, 54(10):105–112, Oct. 2011.

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

[3] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. 2016.

[4] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918, June 2012.

[5] V. Balntas, E. Johns, L. Tang, and K. Mikolajczyk. Pn-net: Conjoined triple deep network for learning local image descriptors. *CoRR*, abs/1601.05030, 2016.

[6] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. *arXiv preprint arXiv:1704.05939*, 2017.

[7] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks.

[8] H. Bay, T. Tuytelaars, and L. Van Gool. *SURF: Speeded Up Robust Features*, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[9] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, Apr 2002.

[10] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):43–57, Jan 2011.

[11] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[12] M. Brown, S. Winder, and G. Hua. Multi-view Stereo Correspondence Dataset. `http://matthewalunbrown.com/patchdata/patchdata.html`, 2011.

[13] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. Brief: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, July 2012.

[14] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[15] C. Ferrari, G. Lisanti, S. Berretti, and A. Del Bimbo. Investigating nuisance factors in face recognition with dcnn representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 81–89, 2017.

[16] M. Grgic, K. Delac, and S. Grgic. Scface – surveillance cameras face database. *Multimedia Tools and Applications*, 51(3):863–879, Feb 2011.

[17] R. Gupta, H. Patil, and A. Mittal. Robust order-based methods for feature description. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 334–341, June 2010.

[18] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[19] N. Jacobs, N. Roman, and R. Pless. Consistent temporal variations in many outdoor scenes. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, June 2007.

[20] Y. Ke and R. Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *The IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–506–II–513 Vol.2, June 2004.

[21] V. Kumar B G, G. Carneiro, and I. Reid. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[22] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *The IEEE International Conference on Computer Vision*, pages 2548–2555, Nov 2011.

[23] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[25] M. Mehdipour Ghazi and H. Kemal Ekenel. A comprehensive analysis of deep learning based representation for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 34–41, 2016.

[26] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, Oct 2005.

[27] S. P. Mudunuri and S. Biswas. Low resolution face recognition across variations in pose and illumination. *IEEE transactions on pattern analysis and machine intelligence*, 38(5):1034–1040, 2016.

[28] S. P. Mudunuri and S. Biswas. Dictionary alignment for low-resolution and heterogeneous face recognition. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 1115–1123. IEEE, 2017.

[29] R. Ortiz. Freak: Fast retina keypoint. In *The IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '12, pages 510–517, Washington, DC, USA, 2012. IEEE Computer Society.

[30] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *The IEEE International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[32] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *The IEEE International Conference on Computer Vision*, December 2015.

[33] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1573–1585, Aug 2014.

[34] A. Subramaniam, P. Balasubramanian, and A. Mittal. PatchMatch NormXcorr source repository. `https://github.com/InnovArul/patchmatch_normxcorr`, 2018.

[35] A. Subramaniam, M. Chatterjee, and A. Mittal. Deep neural networks with inexact matching for person re-identification. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2667–2675. Curran Associates, Inc., 2016.

[36] Y. Tian, B. Fan, and F. Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[37] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, May 2010.

[38] T. Trzcinski, M. Christoudias, V. Lepetit, and P. Fua. Learning image descriptors with the boosting-trick. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 269–277. Curran Associates, Inc., 2012.

[39] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit. TILDE: A Temporally Invariant Learned DEtector. In *Proceedings of the Computer Vision and Pattern Recognition*, 2015.

[40] Z. Wang, B. Fan, and F. Wu. Local intensity order pattern for feature description. In *The IEEE International Conference on Computer Vision*, pages 603–610, Nov 2011.

[41] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE, 2011.

[42] T.-Y. Yang, J.-H. Hsu, Y.-Y. Lin, and Y.-Y. Chuang. Deepcd: Learning deep complementary descriptors for patch representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3314–3322, 2017.

[43] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2015.

[44] H. Zhou, Y. Yuan, and C. Shi. Object tracking using {SIFT} features and mean shift. *Computer Vision and Image Understanding*, 113(3):345 – 352, 2009. Special Issue on Video Analysis.

[45] J. Žbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1592–1599, June 2015.