The Neural-SIFT Feature Descriptor for Visual Vocabulary Object Recognition

Sybren Jansen, Amirhosein Shantia and Marco A. Wiering
Institute of Artificial Intelligence and Cognitive Engineering
University of Groningen
Groningen, The Netherlands

***Abstract*—Recognizing the semantic content of an image is a challenging problem in computer vision. Many researchers attempt to apply local image descriptors to extract features from an image, but choosing the best type of feature to use is still an open problem. Some of these systems are only trained once using a fixed descriptor, like the Scale Invariant Feature Transform (SIFT). In most cases these algorithms show good performance, but they do not learn from their mistakes once training is completed. In this paper a continuous deep neural network feedback system is proposed which consists of an adaptive neural network feature descriptor, the bag of visual words approach and a neural classifier. Two initialization methods for the neural network feature descriptor were compared, one where it was trained on SIFT descriptor output and one where it was randomly initialized. After initial training, the system propagates the classification error from the neural network classifier through the entire pipeline, updating not only the classifier itself, but also the type of features to extract. Results show that for both initialization methods the feedback system increased accuracy substantially when regular training was not able to increase it any further. The proposed neural-SIFT feature descriptor performs better than the SIFT descriptor itself even with a limited number of training instances. Initializing on an existing feature descriptor is beneficial when not a lot of training samples are available. However, when there are a lot of training samples the system is able to construct a well-performing descriptor, solely based on classifier feedback.**

I. Introduction

One of the most challenging problems in computer vision is to recognize the semantic content of an image. This is especially the case when objects vary in pose, where there is occlusion and where differing light conditions are present. Computer vision is important, for example, in image retrieval tasks, where a search query is given and images containing this query should be reported back. Such systems are useful, for example, in medical diagnosis. In robotics a robot has a hard time navigating to a certain place when it can't localize itself. To aid a robot in localization, scene recognition has shown to be very helpful [1], [2]. Additionally, many tasks for a robot involve manipulating certain objects (e.g., bringing coffee or finding emergency buttons in residential care homes). Without knowing which object is which, a robot has difficulty completing any of these tasks.

A common approach to object recognition in complex and changing environments is to extract local feature descriptions from images and attach object class labels to them [3]. Given the extracted features from a test image, these are then matched against features from each class. When there are enough matching features for an object class in a test image, that specific class is detected.

Finding the best features to use is still an open problem. Some use deep learning methods to learn to create features during training [4], [5], [6]. Others apply feature extraction methods using a fixed algorithm. Such algorithms have shown to give good performance in many applications [7], [8], [9], [10], [11], [12], however, the downside of this type of algorithms is that they are not trainable by design. After training there is no feedback loop from the classifier to the feature extraction stage to update the type of features to extract, while there possibly could be room for improvement.

One of these feature descriptors is the well known Scale Invariant Feature Transform (SIFT) [13]. SIFT extracts potential information-rich keypoints and describes them by constructing a histogram of gradient orientations. In recent studies instead of detecting these keypoints a dense fixed grid of evenly spaced keypoints has been used to describe the whole image. To reduce the size of the feature space which results from using a dense fixed grid the bag of visual words approach has been proposed [7]. Inspired by the bag-of-words approach in text classification (e.g., [14]) this approach builds a vocabulary of visual keywords by clustering extracted feature vectors. It uses this vocabulary to construct a histogram counting the number of keyword occurrences in the image, which can then be given to a final classifier.

This paper proposes a continuous feedback system to improve existing feature descriptors. To create a trainable feature descriptor a fixed algorithm has to be made trainable. For this purpose, trainable feature descriptors like artificial neural networks can be applied which can be initially trained on such algorithms. The popular SIFT algorithm will be used for this purpose. Based on this trainable network (termed 'neural-SIFT') a system is proposed which allows for the classification error to be propagated all the way back to the feature extraction network (termed 'full backpropagation'), which in turn tries to improve its feature extraction capabilities. To see what effect initialization on an existing feature descriptor has on performance, the system is also tested using a randomly initialized feature descriptor network. Finally, the original and improved descriptor will be tested in a different setting than in which it was trained, to see if the improved descriptor is generalizable to other systems.

A support vector machine (SVM) [15] is applied for this setting. Each setting is tested on 10 classes of the Caltech-101 dataset and the full Corel-1k dataset. Results show that the improved neural-SIFT descriptor achieves higher performance than the neural-SIFT descriptor trained on the output of the SIFT descriptor and the SIFT descriptor itself. When using enough training samples starting with a random initialization almost achieves the same level of performance than with a SIFT-initialized descriptor. The improved neural-SIFT descriptor also showed to be superior compared to the original one when using a completely different classifier.

In Section II the SIFT descriptor and previous work using the bag of visual words approach is discussed. Section III provides the details of the neural-SIFT approach with full backpropagation training. The experimental setup, the performance of the system and a discussion regarding the results are presented in Section IV. The main conclusions are summarized in Section V.

II. Related work

A. SIFT descriptor

One of the most popular local image descriptors is the Scale Invariant Feature Transform (SIFT) [13]. Typically, SIFT detects stable salient keypoint regions which correspond to parts in the image containing relevant information and creates a description for each of them. The descriptions of these keypoints are constructed in such a way that they are invariant to scale, rotation and partially invariant to affine transformations and illumination changes. Instead of detecting salient keypoints some have used a fixed partitioning scheme to represent the whole image [16], [17], [18]. This approach has been shown to perform similarly well compared to keypoint detection [18].

To describe a keypoint the first step is to assign an orientation to it, which is used in a later step to obtain invariance to rotation. To determine the orientation a histogram is created consisting of 36 bins, each bin covering 10 degrees of a circle. The histogram is formed from the gradient orientations of the neighboring points. For each point in the 16×16 pixel window around the keypoint center the gradient magnitude m and orientation θ are calculated using pixel differences:

$$\begin{aligned} G_x(x,y) &= G(x+1,y) - G(x-1,y) \\ G_y(x,y) &= G(x,y+1) - G(x,y-1) \\ m(x,y) &= \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \\ \theta(x,y) &= \tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y))}\right) \end{aligned}$$

where $G(x,y)$ is the pixel intensity at position (x,y) in the Gaussian smoothed grayscale image, G_x and G_y are the pixel differences in the x and y directions, respectively. Each pixel point is weighted by its gradient magnitude and by a Gaussian-weighted circular window with $\sigma = 1.5$. When the histogram is created the highest peak is detected and used as the keypoint's orientation.

For creating the descriptor again the gradient magnitude and orientations are used, where the keypoint orientation is subtracted from each pixel orientation. Next, the window is divided in 4×4 cells. For each cell a histogram is created consisting of 8 orientation bins (each covering 45 degrees). In a similar way as described above each histogram is filled with the weighted magnitudes of the pixels. The 16 histograms are concatenated to form a 128-dimensional descriptor. In a final step the descriptor is normalized to unit length, values higher than 0.2 are thresholded and given the value 0.2 to overcome some illumination effects. After that, the descriptor is normalized again to unit length.

B. Bag of visual words

The bag of visual words is inspired by the bag-of-words method frequently used in text classification [14], [19]. In text classification word frequency information is gathered and stored in a histogram. Based on this histogram a classifier can determine the semantic context of the text. Apparently there are specific words that have high indicative power for certain contexts. The bag of words method has been proposed for the visual domain as well [7], which has shown to work surprisingly well in image classification and categorization [7], [8], [9], [10], [11], [12].

For the bag of visual words approach the idea is to cluster the extracted features from training images to obtain visual keywords. As a result, these visual keywords represent similar features. The extracted features from a given test image are then matched to the visual keywords and the frequency of matches per cluster is stored in a histogram. This histogram is then used as input for a classifier. The counting of matches per cluster is called the hard bag-of-features approach [7], as it is employing a hard assignment scheme. Other approaches using a soft assignment approach to construct a histogram have been introduced. These approaches give weights to multiple clusters which are close to a feature. Weights can be given by ranking nearest neighbors by distance [20] or using the distances itself [21], [22].

Histograms have the advantage of simplicity and computational efficiency, but ignore any spatial information in the image, information which could be of potential use. It is then also highly surprising that this method shows such great results, even under challenging real-world conditions including intra-class variations and background clutter [9], [23]. This method has even showed its strength in object and scene retrieval in videos [7].

The lack of spatial information has been addressed by using so-called spatial pyramids, first introduced by [24] and later adopted to use with the bag of visual words approach by [11]. The idea behind this is to divide the image into multiple regions and create a histogram for each of them. Spatial information can be captured by combining these histograms to form a set of histograms (e.g., by concatenation). This approach can be applied at different resolutions to create an even richer representation of an image. It is shown that this method can often outperform the single histogram approach [24], [11], [16], [25].

Multiple approaches to clustering the visual keywords have been proposed, some have used k-means [7], [9], others have used Gaussian mixture models (GMMs) [10], [26]. k-means clustering [27] provides a hard assignment scheme, while with GMMs a soft assignment scheme is used where a single feature can belong to multiple clusters. Multiple soft assignment schemes were compared to the traditional hard assignment scheme and the soft assignment approaches have been shown to perform significantly better [8].

III. NEURAL-SIFT

One approach of improving upon fixed feature descriptors, like SIFT, is to make these feature descriptors trainable. This section first proposes the neural-SIFT descriptor, a neural network which mimics the output of the SIFT descriptor. The next part describes the bag of visual words approach which constructs a histogram of visual keyword occurrences. This is done by applying a sliding window over the image, extracting feature vectors from each window and comparing them to the established keyword vocabulary. This histogram is used as input for the neural network classifier and is described in Section III-C. The final part presents the full backpropagation training scheme, where the error of the neural classifier is propagated all the way back to the neural-SIFT descriptor.

A. Neural-SIFT feature extraction

Before extracting features from an image the same preprocessing steps are applied as are used by SIFT (i.e., conversion to gray scale and applying Gaussian smoothing).

In creating the SIFT descriptor a fixed-sized window around a keypoint is used. Typically, the size is chosen to be 16×16 pixels. For each point in the window gradient magnitudes and orientations are calculated using the four direct adjacent points. This means that by providing only the 16×16 pixels for a given window as input for the neural network results in incomplete information. Therefore, the neural network receives a 18×18 window as input.

The target function of the network is the local image descriptor function used in the SIFT algorithm: the SIFT descriptor. The dimensionality of the SIFT descriptor is 128, therefore the output layer of the neural network consists of 128 units as well. One modification to the SIFT algorithm is made: the keypoint's orientation will not be used to make the keypoint rotation invariant. The reason for this is that preliminary experiments showed that SIFT performed better on the datasets without this step than with. This could be due to the nature of the datasets in which not many rotated objects occur. Another consideration for this is that the function to learn as a consequence becomes easier to fit using a neural network.

As a single hidden layer in a network is enough to fit any continuous function [28] only one hidden layer is used. To establish the number of hidden units to use one has to consider that when applying full backpropagation training in a later stage the network will be further trained on the same training data. When the number of hidden units is large overfitting can occur more easily. So, even though a smaller train and validation error for this network can be obtained by using more hidden units, using less units may give better results at a later stage. The number of units as well as the type of activation function to use in the hidden layer are determined empirically, taking these aspects into account. For the output layer the linear activation function is used as the network is dealing with a continuous target function.

Even though the SIFT descriptor is computed using local neighborhood pixels all layers are fully connected. This enables the network to possibly learn more complex functions when the error from the classifier is included. Both the input and the hidden layer have a single bias unit which always has a value of $+1$ as input.

A sliding window over the whole image is applied which results in a large number of patches per image. Considering all images this results in a large number of training samples. Therefore, classic online gradient descent learning is used in conjunction with $L2$-norm regularization for training this network. Preliminary experiments showed that 200 epochs was a sufficient amount of epochs for the network to stabilize. Training is therefore terminated after a fixed amount of 200 epochs.

B. Bag of visual words

After the neural-SIFT network is trained the visual vocabulary is constructed by creating clusters which represent the data as closely as possible. GMMs showed to be too computationally intensive when dealing with 128 dimensional feature vectors. Therefore, the clusters are determined using the k-means algorithm which involves choosing the distance metric and the number of clusters to use. As distance metric the squared Euclidean distance is chosen, the number of clusters to use is determined empirically.

The next step is to create an image histogram which serves as input for the final classifier, of which many hard and soft assignment schemes are available. For this system a soft assignment scheme is applied as they are shown to perform significantly better than the hard methods [8]. One example soft approach transforms the distances from a feature vector to each cluster centroid to a sort of similarity value [21]. Another uses the codeword uncertainty method which calculates the probability of a feature vector belonging to a certain cluster, where the amount of probability mass is normalized [22]. A hybrid approach is used for this system in which similarities are calculated in a similar fashion as in [21] and the amount of probability mass is normalized similarly as in [22]. This method computes the similarity $s_{i,j}$ between feature vector f_i and cluster center c_j using:

$$s_{i,j} = \frac{\exp(-\zeta * d_{i,j})}{\sum_{k=1}^{C} \exp(-\zeta * d_{i,k})} \tag{1}$$

where $d_{i,j}$ is the distance between feature vector f_i and cluster center c_j and ζ is a constant specifying how much to penalize distance. The more the distance is penalized the more this function approximates the actual max function.

When for each patch in the image the vector of similarity values has been computed they are summed and divided by the number of windows (F) to obtain the required image histogram entry g_j for cluster j:

$$g_j = \frac{1}{F} \sum_{i=1}^{F} s_{i,j} \tag{2}$$

To make this histogram more suitable for the classifier input the histogram is normalized to the range $[-1, +1]$ for each input x_i by:

$$x_i = \frac{2 * (g_i - min_i)}{max_i - min_i} - 1 \tag{3}$$

where min_i and max_i are the lowest and highest value of g_i over all the training data, respectively.

C. Neural classifier

The result of the previous stage is a histogram of visual word frequencies. The dimensionality of this histogram equals the number of clusters used, C. The number of input units for the neural network classifier, therefore, equals C as well. The number of output units equals the number of object classes N. As with the neural-SIFT network only one hidden layer is used. Again, the number of hidden units and the activation functions used were set experimentally. The difference with respect to the neural-SIFT network is that the target function in this case is a binary function where each output unit corresponds to a one-versus-all classifier. Therefore, the softmax function is used at the output layer. All layers are fully connected and have an additional bias unit. This time the number of training samples is much lower than when training the neural-SIFT network. Therefore, instead of using online learning iRPROP$^+$ training [29] is applied with $L2$-norm regularization. When not encountering a better validation error within 50 epochs of the current minimum, training is terminated and the stored weights at the lowest point are applied.

D. Full backpropagation training

When all parts of the system are individually trained the system as a whole is trained by propagating the error from the classification output all the way back to the neural-SIFT weights, using gradient descent. For this training scheme the chain rule is applied and the whole pipeline can be split up in three parts. First, the classification error is propagated back to the input of the classifier network, which then equals the error at the normalized image histogram. This part can be derived in a similar fashion as one would derive the update equation of the weights going from the input to the hidden layer of a standard feedforward multilayer perceptron (MLP). When deriving this update equation the final part going from the weighted sum a, of the input x and weights w, to a single weight w_{ij} equals:

$$\frac{\partial a_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\sum_{k=1}^{C} x_k w_{kj} \right) = x_i \tag{4}$$

Deriving to the input instead the modification to be made is that the derivative should not be with respect to a weight w_{ij}, but to an input x_i. This partial derivative then simply results in w_{ij}.

Next, the error is further propagated through the steps involved in creating the image histogram:

$$\frac{\partial E}{\partial f_{f,e}} = \sum_{n=1}^{C} \left(\sum_{k=1}^{C} \left[\frac{\partial E}{\partial x_k} \frac{\partial x_k}{\partial g_k} \frac{\partial g_k}{\partial s_{f,k}} \frac{\partial s_{f,k}}{\partial d_{f,n}} \right] \frac{\partial d_{f,n}}{\partial f_{f,e}} \right) \tag{5}$$

where E is the error at the classifier output, $f_{f,e}$ is the e-th element of the feature vector corresponding to image patch f, generated by the neural-SIFT network, and $\frac{\partial E}{\partial x_k}$ is the already derived error at the input of the classifier network.

First, normalization to the range $[-1, +1]$ is derived:

$$\begin{aligned} \frac{\partial x_k}{\partial g_k} &= \frac{\partial}{\partial g_k} \left(\frac{2 * (g_k - min_k)}{max_k - min_k} - 1 \right) \\ &= \frac{2}{max_k - min_k} \end{aligned} \tag{6}$$

The following part corresponds to summing the similarity values and dividing the resulting histogram by the number of patches in the image:

$$\frac{\partial g_k}{\partial s_{f,k}} = \frac{\partial}{\partial s_{f,k}} \left(\frac{1}{F} \sum_{l=1}^{F} s_{l,k} \right) = \frac{1}{F} \tag{7}$$

At this point the error is split up and given to each individual image patch. This means that the subsequent equations need to be applied for each image patch.

The next partial derivative is very similar to deriving a softmax function. In this case there is a function within each exponential function, for which the chain rule can be used. The part that is left to derive is:

$$\frac{\partial}{\partial d_{f,k}} \left(-\zeta d_{f,k} \right) = -\zeta \tag{8}$$

Combining this with the derivative of the softmax function this results in:

$$\frac{\partial s_{f,k}}{\partial d_{f,n}} = -\zeta s_{f,k} (\delta_{k,n} - s_{f,n}) \tag{9}$$

where $\delta_{k,n}$ is the Kronecker delta function.

Finally, the distance computation is derived:

$$\begin{aligned} \frac{\partial d_{f,n}}{\partial f_{f,e}} &= \frac{\partial}{\partial f_{f,e}} \| f_f - c_n \|^2 \\ &= \frac{\partial}{\partial f_{f,e}} \sum_{l=1}^{O} (f_{f,l} - c_{n,l})^2 \\ &= 2(f_{f,e} - c_{n,e}) \end{aligned} \tag{10}$$

When the error is calculated at the output of the neural-SIFT network the final part of backpropagation begins where the error with respect to the weights of the neural-SIFT network are calculated. This follows the same update equations for a standard MLP. The only difference on this part is that the error should be considered with respect to classifier output. iRPROP$^+$ training with $L2$-norm regularization is used to update the weights. As with the neural classifier the settings at the lowest validation error are stored. When no lower validation error is encountered within 50 epochs of the current lowest error training is stopped and the stored settings are applied. This marks one complete iteration of full backpropagation. After one single iteration the system reaches a point where performance cannot be improved anymore by using regular training schemes, but at this point the descriptor has been updated. This translates to a clustering which does not entirely match with the underlying feature vectors. Therefore, the clustering and consequently the neural classifier can be updated. When this retraining is complete a second iteration of full backpropagation can start. This cycle of full backpropagation and retraining can be repeated indefinitely, until no more improvement can be obtained.

IV. Experiments and Results

A. Experimental setup

This section is divided into multiple parts. Section IV-B provides a short overview of the used datasets and Section IV-C describes the training results of the proposed system with full backpropagation training. Results are compared to systems using the SIFT descriptor and the original neural-

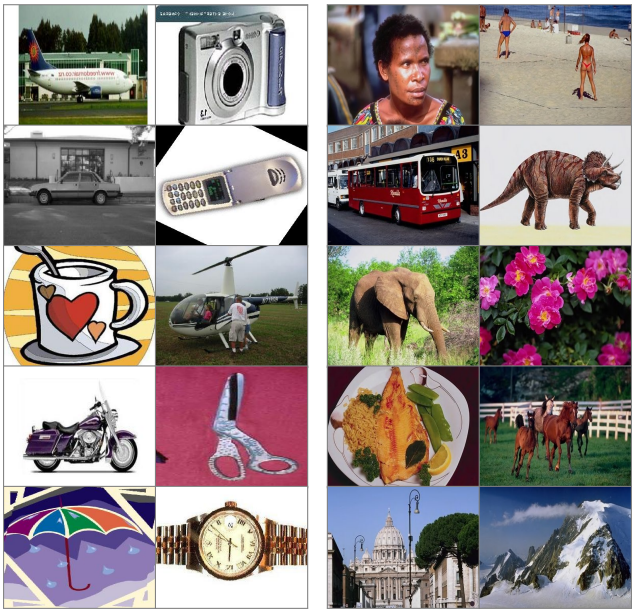

Fig. 1. On the left: example images of the Caltech-101 dataset showing one image of each of the classes: airplanes, cameras, car sides, cellphones, cups, helicopters, motorbikes, scissors, umbrellas and watches. On the right: example images of the Corel-1k dataset showing one image of each of the classes: African people, beaches, buses, dinosaurs, elephants, flowers, food, horses, monuments and mountains.

SIFT descriptor. To investigate the influence of initializing the neural-SIFT network on the SIFT descriptor, Section IV-D compares the proposed system with using a randomly initialized untrained neural network as feature extractor, both using the full backpropagation training step. Finally, in Section IV-E generalizability of the Neural-SIFT descriptor is tested with the use of an SVM classifier.

Preliminary experiments were set up to determine the number of hidden units and type of activation function to use in the hidden layer of the neural-SIFT network, the number of clusters to use and the number of hidden units and type of activation function to use in the neural classifier network. We used 50 units and the logistic function with a steepness of 0.75 in the hidden layer of the neural-SIFT network and the linear activation function with a steepness of 0.25 in the output layer. Two sizes of the visual vocabulary were considered, being 100 and 300 clusters. Using 100 clusters setting ζ to 10 provided the best results. In this setting the neural classifier features 75 hidden units using the hyperbolic tangent activation function with a steepness of 0.10. For the 300 clusters setting ζ was set to 12, using 75 hidden units with the hyperbolic tangent activation function with a steepness of 0.25.

For each of the experiments presented in this paper 10-fold cross-validation is applied to more accurately determine the optimal parameters and to predict the accuracy of the system.

B. Datasets

1) Caltech-101: The Caltech-101 dataset consists of images of 101 object classes. Each class contains about 40-800 images, where most of the classes contain around 50. The resolution of each image varies around 300×200 pixels and can be in both portrait or landscape mode. Of all images

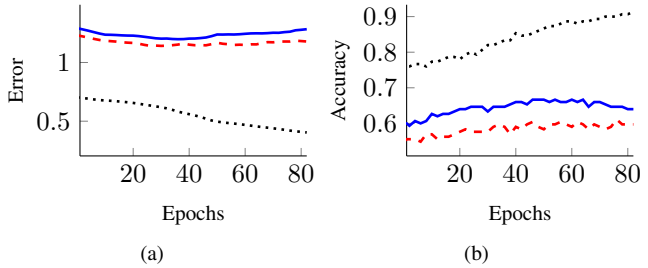

Fig. 2. Example error **(a)** and accuracy **(b)** curve for the first iteration of full backpropagation for a single fold using 100 clusters (Caltech-101 dataset). The black (dotted), red (dashed) and blue (solid) curves represent the training, validation and test set, respectively.

most have little or no clutter, objects tend to be centered in each image and objects are presented in a stereotypical pose. For the experiments 10 classes were selected. These include: airplanes, cameras, car sides, cellphones, cups, helicopters, motorbikes, scissors, umbrellas and watches. Example images are shown in the left column of Fig. 1. For evaluating the different methods 15 training, 15 validation and 15 test images were used for each class.

2) Corel-1k: The Corel-1k dataset consists of 1000 images sorted in 10 classes, with 100 images per class. The classes are: African people, beaches, busses, dinosaurs, elephants, flowers, food, horses, monuments and mountains. Example images are shown in the right column of Fig. 1. The resolution of the images are all 384×256 pixels, either in portrait or in landscape mode. All classes of the Corel-1k dataset were selected for the experiments. From the available 100 images per class 60 were used for training, 20 for validation and 20 for testing.

C. Results on full-backpropagation

After each individual stage of the system is trained the system reaches a point where traditional training cannot push the performance any further. At this point full backpropagation is applied. First, a single iteration of full backpropagation is considered, which can consist of multiple training epochs. An example error curve for the training, validation and test set for a single fold in the 100 clusters setting is shown in Figure 2. At first both validation and test error go down together with the train error, which coincides with an increase in both validation and test accuracy. After around epoch 30 overfitting starts to occur and both the validation and test set error go up.

For all folds, on average the train, validation and test set errors went down and accuracy went up for both datasets. After one single iteration test accuracy went up by 1.50% for the Caltech-101 dataset and 1.74% for the Corel-1k dataset. When using 300 clusters the test accuracy for the Caltech-101 dataset went up from 68.53% to 69.80% (+1.37%). For the Corel-1k dataset accuracy went up from 85.10% to 86.70% (+1.60%).

After one single iteration regular training schemes cannot improve performance any further. As described earlier additional iterations can be used to possibly push the system even further. The procedure involves updating the current clustering and the neural classifier based on the improved neural-SIFT network. When these are retrained a second iteration of full backpropagation can begin, this cycle can be

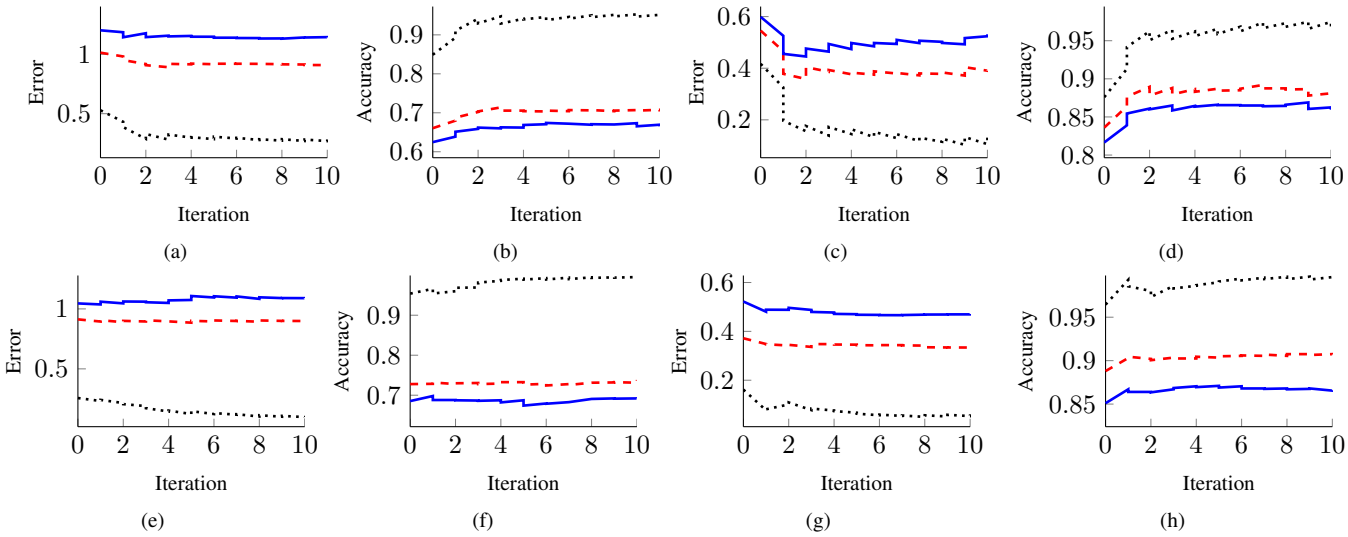

Fig. 3. Full backpropagation training progress averaged over 10 folds. For the $k = 100$ clusters setting **(a)** and **(b)** show the training progress for the Caltech-101 dataset, the progress for the Corel-1k dataset is shown in **(c)** and **(d)**. For $k = 300$ clusters **(e)** and **(f)** show the training progress for the Caltech-101 dataset, **(g)** and **(h)** show the progress for the Corel-1k dataset. Vertical updates indicate the error and accuracy after retraining the clusters and the neural classifier. The black (dotted), red (dashed) and blue (solid) curves represent the training, validation and test set, respectively.

repeated over and over again, until no further improvement is observed. For each experiment involving full backpropagation a maximum of 10 iterations is used. This proved to be a sufficient number of iterations for the error to stabilize or to see overfitting occur.

The development of the errors and accuracies over multiple iterations for both datasets using 100 clusters is shown in Figure 3 (a-d). For the Caltech-101 dataset the validation error went down each iteration, but went up or down during retraining the clusters and classifier (indicated by the vertical updates at each iteration point). On average the lowest validation error was reached at iteration 3, before retraining. From iteration 4 onwards the validation error stabilized and for most folds no further training was possible while not letting the validation error increase. For the Corel-1k dataset on average the lowest validation error was observed at iteration 2, before retraining. After that point retraining most of the time lead to an increase of validation error. The training error was still going down which indicates overfitting on the training set. Figure 3 (e-h) shows the development of the errors and accuracies over multiple iterations in a similar way, but with using 300 clusters instead.

The exact figures for all settings before applying full backpropagation and at the point where the validation error was the lowest are shown in Table I. The performance of using the improved neural-SIFT descriptor is compared with the SIFT descriptor there as well. Before full backpropagation the SIFT descriptor obtained higher accuracy rates for both datasets in the 100 clusters setting and achieved higher accuracy for the Corel-1k dataset for the 300 cluster setting. Only the Caltech-101 dataset with 300 clusters showed a slightly better performing neural-SIFT descriptor. After applying multiple iterations of full backpropagation, however, the improved neural-SIFT descriptor takes over and obtains better performance for all settings.

For the Caltech-101 dataset increases of $+3.33\%$ and $+4.87\%$ were realized for the 100 and 300 clusters setting, respectively. A student t-test was conducted to compare the two descriptors. For the 100 clusters setting there was a significant difference in the scores for the SIFT descriptor ($M = 62.80$, $SEM = 0.95$) and the neural-SIFT descriptor ($M = 66.13$, $SEM = 1.24$) conditions; $t(18) = 2.13$, $p = 0.047$. Also the 300 clusters setting showed a significant difference between the SIFT ($M = 65.33$, $SEM = 1.30$) and the neural-SIFT descriptor ($M = 70.20$, $SEM = 0.57$) conditions; $t(18) = 3.43$, $p = 0.003$.

For the Corel-1k dataset increases of $+3.00\%$ and $+0.60\%$ were realized. The student t-test could not be used here because we observed that the outcomes do not follow a normal distribution, so the Binomial test was used instead. The results indicated that for both the 100 and 300 clusters setting there were 8 wins, 2 losses. The cumulative Binomial test then gives $P(X \geq 8 \mid N = 10) = 0.054$, which is almost significant at the $p = 0.05$ level.

D. Results on random initialization

All settings used for this system (termed neural-RANDOM) are identical to the neural-SIFT based system. The only aspect that changed is the initialization of the neural-SIFT network: now it is not pretrained on SIFT descriptor output. The weights of this network were randomly initialized in a broad range, namely $[-0.5, +0.5]$.

The results are summarized in Table I. Before full backpropagation training the neural-RANDOM based system performed less well than the SIFT and original neural-SIFT based systems. After applying full backpropagation training performance increased for each setting. In three out of four settings the neural-RANDOM descriptor still performed less well than the SIFT descriptor. Only for the Corel-1k dataset using 100 clusters the neural-RANDOM descriptor showed a higher recognition accuracy ($+0.45\%$). Compared to the improved neural-SIFT descriptor the neural-RANDOM descriptor falls behind. The difference is much larger for the Caltech-101 dataset than for the Corel-1k dataset. Adding more training images seems to have a positive effect on

TABLE I. AVERAGE CLASSIFICATION ACCURACY OF THE TEST SET OVER 10 FOLDS USING THE NEURAL CLASSIFIER WITH EITHER THE SIFT DESCRIPTOR, THE ORIGINAL OR IMPROVED NEURAL-SIFT NETWORK, THE ORIGINAL OR IMPROVED RANDOMLY INITIALIZED NEURAL-RANDOM NETWORK OR THE SVM CLASSIFIER WITH THE ORIGINAL OR IMPROVED NEURAL-SIFT NETWORK. THE STANDARD ERROR OF THE MEAN (SEM) IS SHOWN FOR THE SIFT AND IMPROVED NEURAL-SIFT DESCRIPTOR WITH THE NEURAL CLASSIFIER. FOR THE NEURAL CLASSIFIER THE NUMBERS IN PARENTHESES BELOW THE PERCENTAGES SHOW THE DIFFERENCE IN PERFORMANCE WHEN COMPARED TO THE SIFT DESCRIPTOR. FOR THE SVM THIS IS SHOWN IN COMPARISON TO USING THE ORIGINAL NEURAL-SIFT FEATURE DESCRIPTOR.

100 clusters							
	SIFT	**Neural-SIFT**		**Neural-RANDOM**		**SVM classifier + neural-SIFT**	
	descriptor (SEM)	**Original**	**Improved (SEM)**	**Original**	**Improved**	**Original**	**Improved**
Caltech-101	62.80% (0.95)	62.47%	66.13% (1.24)*	50.33%	58.73%	75.90%	77.08%
		(−0.33%)	(+3.33%)	(−12.47%)	(−4.07%)		(+1.18%)
Corel-1k	83.15% (2.61)	81.65%	86.15% (3.18)	78.10%	83.60%	90.10%	90.55%
		(−1.50%)	(+3.00%)	(−5.05%)	(+0.45%)		(+0.45%)
300 clusters							
	SIFT	**Neural-SIFT**		**Neural-RANDOM**		**SVM classifier + neural-SIFT**	
	descriptor (SEM)	**Original**	**Improved (SEM)**	**Original**	**Improved**	**Original**	**Improved**
Caltech-101	65.33% (1.30)	68.53%	70.20% (0.57)**	54.59%	61.80%	79.17%	79.72%
		(+3.20%)	(+4.87%)	(−10.74%)	(−3.53%)		(+0.55%)
Corel-1k	86.25% (2.61)	85.10%	86.85% (2.98)	76.90%	84.70%	90.25%	91.40%
		(−1.15%)	(+0.60%)	(−9.35%)	(−1.55%)		(+1.15%)

* The mean difference is significant at the 0.05 level

** The mean difference is significant at the 0.005 level

the learning capabilities of the system through full backpropagation. Although a properly initialized local image descriptor neural network gives a head start performance-wise, if there are enough train images available the system can recover quite well using full backpropagation. In other words, it can come up with its own representation of what a good image feature looks like. For the Corel-1k dataset with 100 clusters, for example, it even outperformed the SIFT descriptor without any prior knowledge on how to describe local parts of an image.

E. Results on generalization

In order to test generalizability of the improved neural-SIFT descriptor, performance is measured with a different classifier: a support vector machine (SVM). This classifier is trained using the original neural-SIFT network and the corresponding clusters in the first setting and using the improved neural-SIFT network and the corresponding clusters in the second. This experiment is applied for both 100 and 300 clusters.

The radial basis function (RBF) is used as kernel function for the SVM. A grid-search algorithm is applied to fine-tune the C and γ parameters. First a coarse grid search is used with C-values of $2^{-5}, 2^{-4}, ..., 2^{15}$ and γ-values of $2^{-15}, 2^{-14}, ..., 2^{3}$. The best performing parameters, C^* and γ^*, are used as the starting point for a fine grid-search. Here C ranges from $0.5C^*$ to $2.0C^*$ and γ ranges from $0.5\gamma^*$ to $2.0\gamma^*$ where each range is divided in 20 equal steps. The best parameters of the fine grid-search are used to test the performance of the system.

The results for this experiment are shown in Table I. When using the original neural-SIFT descriptor the SVM performed better than the neural classifier with the original neural-SIFT network for each number of clusters and type of dataset. Especially for the Caltech-101 dataset the SVM performed a lot better: $+13.43\%$ for the 100 clusters setting and $+10.64\%$ for using 300 clusters. When training the SVM with the improved neural-SIFT network instead the performance for each setting improved further. The improved neural-SIFT descriptor not only led to an improvement for the neural classifier by which it was trained, but it also led to an improvement when using a completely different classifier. A classifier which has little or no structural resemblance to a neural network. Although the performance increases are small, this shows that the full backpropagation system has the capability to improve a feature descriptor which can then be used in other settings than the one in which it was trained.

V. CONCLUSIONS

In this paper a deep neural network training framework was introduced which propagates the classification error through the entire pipeline all the way back to the feature extraction stage to learn to extract 'better' features than the initial local image descriptor. A neural network was trained on the output of the SIFT descriptor and by applying the proposed full backpropagation training scheme the system was able to achieve higher recognition accuracy in each single setting. Performance increases between 0.60% and 4.87% were realized when compared to the SIFT descriptor.

The influence of initializing the feature extracting neural network on an existing local image descriptor was examined by using a randomly initialized network instead. The system showed a much lower performance initially, but was able to narrow the performance-gap quite a bit after applying full backpropagation training. This gap was a lot smaller for the Corel-1k dataset where a lot more training images were available. This shows that when a system has many training samples available it can figure out by itself, without any prior knowledge on the subject, how a distinctive image descriptor should look like. Still, the results also showed that pretraining on SIFT outputs led to better final recognition accuracies than initializing the feature extraction neural networks completely randomly.

The image descriptor was improved in one specific

context, with the bag of visual words approach and a neural network classifier. An SVM classifier was used to account for context and in each setting the performance increased when using the improved neural-SIFT descriptor instead of the original. However, these improvements were small and were only achieved in one other context. Other contexts should be tested to get a better view on the generalizability of the improved descriptor.

As future work, other feature descriptors, or combinations of them, can be applied. In fact, any type of feature descriptor which takes a range of pixels as input can be used in this framework and it would be interesting to see what performance increases are possible using other feature descriptors. One point of improvement lies in the clustering. During retraining of the clusters and the classifier the validation errors regularly went up. This could be due to the fact that the clustering is performed by the k-means algorithm, which applies a hard assignment scheme. The error propagation through the clustering makes use of a soft assignment scheme. This mismatch can be solved by using a different clustering approach which also applies a soft assignment scheme. One example of such an approach is Gaussian mixture models, although this approach is more computationally expensive. Finally, possible performance increases can be achieved by implementing spatial pyramids or by applying other classification systems.

REFERENCES

[1] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1403–1410.

[2] T. Botterill, S. Mills, and R. Green, "Speeded-up bag-of-words algorithm for robot localisation through scene recognition," in *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*. IEEE, 2008, pp. 1–6.

[3] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: a survey," *Foundations and Trends® in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2008.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. Association for Computing Machinery, 2009, pp. 609–616.

[6] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[7] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1470–1477.

[8] A. Abdullah, R. Veltkamp, and M. Wiering, "Ensembles of novel visual keywords descriptors for image categorization," in *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference*. IEEE, 2010, pp. 1206–1211.

[9] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, European Conference on Computer Vision*, vol. 1, 2004, p. 22.

[10] J. Farquhar, S. Szedmak, H. Meng, and J. Shawe-Taylor, "Improving "bag-of-keypoints" image categorisation: Generative models and pdf-kernels," University of Southampton, Tech. Rep., 2005.

[11] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference*, vol. 2. IEEE, 2006, pp. 2169–2178.

[12] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[13] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] T. Joachims, "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization." DTIC Document, Tech. Rep., 1996.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[16] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *Proceedings of the 6th ACM international conference on Image and video retrieval*. Association for Computing Machinery, 2007, pp. 401–408.

[17] A. Abdullah, R. C. Veltkamp, and M. A. Wiering, "Spatial pyramids and two-layer stacking svm classifiers for image categorization: A comparative study," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. IEEE, 2009, pp. 5–12.

[18] A. Abdullah, R. Veltkamp, and M. Wiering, "Fixed partitioning and salient points with MPEG-7 cluster correlograms for image categorization," *Pattern Recognition*, vol. 43, no. 3, pp. 650–662, 2010.

[19] D. D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *Machine learning: European Conference on Machien learning-98*. Springer, 1998, pp. 4–15.

[20] Y.-G. Jiang, C.-W. Ngo, and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *Proceedings of the 6th ACM international conference on Image and video retrieval*. Association for Computing Machinery, 2007, pp. 494–501.

[21] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[22] J. C. van Gemert, C. J. Veenman, A. W. Smeulders, and J.-M. Geusebroek, "Visual word ambiguity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1271–1283, 2010.

[23] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *International journal of computer vision*, vol. 73, no. 2, pp. 213–238, 2007.

[24] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1458–1465.

[25] E. Hadjidemetriou, M. D. Grossberg, and S. K. Nayar, "Spatial information in multiresolution histograms," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–702.

[26] F. Perronnin, C. Dance, G. Csurka, and M. Bressan, "Adapted vocabularies for generic visual categorization," in *Computer Vision–European Conference on Computer Vision 2006*. Springer, 2006, pp. 464–475.

[27] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. California, USA, 1967, pp. 281–297.

[28] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[29] C. Igel and M. Hüsken, "Improving the Rprop learning algorithm," in *Proceedings of the second international symposium on neural computation (NC 2000)*, vol. 2000. International Conference on Semantic Computing, 2000, pp. 115–121.