A BoW-equivalent Recurrent Neural Network for Action Recognition

Alexander Richard
richard@iai.uni-bonn.de
Juergen Gall
gall@iai.uni-bonn.de

Institute of Computer Science III
University of Bonn
Bonn, Germany

Abstract

Bag-of-words (BoW) models are widely used in the field of computer vision. A BoW model consists of a visual vocabulary that is generated by unsupervised clustering the features of the training data, *e.g.*, by using kMeans. The clustering methods, however, struggle with large amounts of data, in particular, in the context of action recognition. In this paper, we propose a transformation of the standard BoW model into a neural network, enabling discriminative training of the visual vocabulary on large action recognition datasets. We show that our model is equivalent to the original BoW model but allows for the application of supervised neural network training. Our model outperforms the conventional BoW model and sparse coding methods on recent action recognition benchmarks.

1 Introduction

Bag-of-words (BoW) or bag-of-features representations for images or videos have been widely used in computer vision applications like texture classification [40], object classification [8], object discovery [28], or object retrieval [27]. Although other representations like improved Fisher vectors [23] usually lead to better classification results, BoW models are still very popular since they can be efficiently computed and have a low memory footprint in comparison to more advanced representations [23]. This is in particular relevant for videos. BoW models are therefore widely used for action recognition [11, 24, 32, 36].

Usually, kMeans or a Gaussian mixture model (GMM) is used to generate a visual vocabulary based on which histograms of visual words are computed in order to quantize and represent the input data. However, generating the visual vocabulary with kMeans or GMMs has several drawbacks. State-of-the-art feature extraction algorithms, such as dense trajectories (DT) [36] for action recognition, generate a huge amount of data. It is usually infeasible to cluster the complete data, which thus needs to be sparsely sampled. For instance in [36], several visual vocabularies are created by kMeans and based on a heuristic the best one is selected. Furthermore, the visual vocabulary is created without supervision and not optimized for the task to solve. This results in a loss of discriminative power as we will show in our experiments.

In this work, we propose a method to generate visual words using discriminative training by converting the kMeans-based BoW model into an equivalent recurrent neural network.

In this way, we can learn a visual vocabulary by maximizing the class posterior probability instead of minimizing the sum of squared distances of the training data to their nearest visual words. This results in a more discriminative visual vocabulary. Our approach is the first that allows for codebook training directly on video level rather than on descriptor level only.

For evaluation, we use the state-of-the-art action recognition framework proposed in [34, 36] and replace the kMeans-based BoW model by our recurrent neural network. We validate on four recent action recognition datasets that our model outperforms the kMeans baseline by two to five percent points and allows for a considerable reduction in the amount of extracted features. We further compare our method to state-of-the-art feature encodings that are widely used in action recognition and image classification.

2 Related work

There have been several approaches to improve the standard BoW model. Perronnin *et al*. [22] propose to extend the global vocabularies with class-specific vocabularies and represent an image by multiple histograms. Cai *et al*. [3] apply metric learning to learn a weighted codebook. In [16], a dictionary is learned by combining a Gaussian mixture model with a logistic regression model. In the context of sparse coding, LLC [37] and ScSPM [38] have shown good results in image classification. Boureau *et al*. [1] show that sparse coding methods can be improved by supervised dictionary learning. In the work of Goh *et al*. [9], a restricted Boltzman machine is used to learn a sparse coding which can be refined with supervised fine-tuning. Their approach, however, only allows for training on descriptor level, but not for training on video level directly. Yet, neither of the approaches has been applied to action recognition.

In parallel, neural networks are of increasing interest for action recognition. Karpathy *et al*. trained a convolutional neural network (CNN) on one million weakly annotated Youtube videos [12]. In [25], a two-stream CNN processing single frames as well as optical flow is successfully applied to two action recognition benchmarks. The authors of [6] and [30] use a CNN to extract features on each video frame and train an LSTM network to explore temporal information. Still, these approaches do not yet outperform hand-crafted features such as the dense trajectories of [36].

In the context of action recognition, improved Fisher vectors [23] recently became a topic of interest [20, 21, 34]. Although they usually lead to better classification results, the computation and memory requirements are higher than for a BoW model. Sydorov *et al*. [31] investigate similarities in the structure of Fisher vectors and neural networks and propose a method to optimize Fisher vectors and a classifier jointly. In [26], an architecture of several stacked Fisher layers is proposed and a dimension reduction is learned discriminatively in each layer.

3 Bag-of-Words Model as Neural Network

In this section, we first define the standard BoW model and propose a neural network representation. We then discuss the equivalence of both models.

3.1 Bag-of-Words Model

Let $\mathbf{x} = (x_1, \dots, x_T)$ be a sequence of D-dimensional feature vectors $x_i \in \mathbb{R}^D$ extracted from a video, $\mathcal{C} = \{1, \dots, C\}$ the set of classes, and $\{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_N, c_N)\}$ the training data. In the case of action recognition, for example, each observation $\mathbf{x}_i$ is a sequence of feature vectors extracted from a video and c_i is the action class of the video. Note that the sequences usually have different lengths, *i.e.* for two different observations $\mathbf{x}_i$ and $\mathbf{x}_j$, usually $T_i \neq T_j$.

The objective of a BoW model is to quantize each observation $\mathbf{x}$ using a fixed vocabulary $\mathcal{V} = \{v_1, \dots, v_K\} \subset \mathbb{R}^D$ of K visual words. To this end, each sequence is represented as a histogram of posterior probabilities $p(v|x)$,

$$\mathcal{H}(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} h(x_t), \quad h(x_t) = \begin{pmatrix} p(v_1|x_t) \\ \vdots \\ p(v_K|x_t) \end{pmatrix}. \tag{1}$$

Frequently, kMeans is used to generate the visual vocabulary. In this case, $h(x_t)$ is a unit vector, *i.e.* the closest visual word has probability one and all other visual words have probability zero. Based on the histograms, a probability distribution $p(c|\mathcal{H}(\mathbf{x}))$ can be modeled. Typically, a support vector machine (SVM) in combination with a non-linear kernel is used for classification.

3.2 Conversion into a Neural Network

The result of kMeans can be seen as a mixture distribution describing the structure of the input space. Such distributions can also be modeled with neural networks. In the following, we propose a transformation of the BoW model into a neural network.

The nearest visual word $\hat{v} = \arg\min_k \|x - v_k\|^2$ for a feature vector x can be seen as the maximizing argument of the posterior form of a Gaussian distribution,

$$p_{\text{KM}}(v_k|x) = \frac{p(v_k)p(x|v_k)}{\sum_{\tilde{k}} p(v_{\tilde{k}})p(x|v_{\tilde{k}})} = \frac{\exp\left(-\frac{1}{2}(x - v_k)^\intercal (x - v_k)\right)}{\sum_{\tilde{k}} \exp\left(-\frac{1}{2}(x - v_{\tilde{k}})^\intercal (x - v_{\tilde{k}})\right)}, \tag{2}$$

assuming a uniform prior $p(v_k)$ and a normal distribution $p(x|v_k) = \mathcal{N}(x|v_k, \mathbf{I})$ with mean v_k and unit variance. Using maximum approximation, *i.e.* shifting all probability mass to the most likely visual word, a probabilistic interpretation for kMeans can be obtained:

$$\hat{p}_{\text{KM}}(v_k|x) = \begin{cases} 1, & \text{if } v_k = \arg\max_{\tilde{k}} p_{\text{KM}}(v_{\tilde{k}}|x), \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Inserting $\hat{p}_{\text{KM}}(v_k|x)$ into the histogram equation (1) is equivalent to counting how often each visual word v_k is the nearest for the feature vectors $x_1, \dots, x_T$ of a sequence $\mathbf{x}$.

Now, consider a single-layer neural network with input $x \in \mathbb{R}^D$ and K-dimensional softmax output that defines the posterior distribution

$$p_{\text{NN}}(v_k|x) := \text{softmax}_k(\mathbf{W}^\intercal x + b) = \frac{\exp\left(\sum_d w_{d,k} x_d + b_k\right)}{\sum_{\tilde{k}} \exp\left(\sum_d w_{d,\tilde{k}} x_d + b_{\tilde{k}}\right)}, \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{D \times K}$ is a weight matrix and $b \in \mathbb{R}^K$ the bias. With the definition

$$\mathbf{W} = (v_1 \dots v_K), \tag{5}$$

$$b = -\frac{1}{2}(v_1^\intercal v_1 \dots v_K^\intercal v_K)^\intercal, \tag{6}$$

an expansion of the right hand side of Equation (2) reveals that

$$p_{\text{NN}}(v_k|x) = \frac{\exp\left(-\frac{1}{2}v_k^\intercal v_k + v_k^\intercal x\right)}{\sum_{\tilde{k}}\exp\left(-\frac{1}{2}v_{\tilde{k}}^\intercal v_{\tilde{k}} + v_{\tilde{k}}^\intercal x\right)} = p_{\text{KM}}(v_k|x). \tag{7}$$

A recurrent layer without bias and with unit matrix as weights for both the incoming and recurrent connection is added to realize the summation over the posteriors $p_{\text{NN}}(v_k|x)$ for the histogram computation, *c.f.* Equation (1). The histogram normalization is achieved using the activation function

$$\sigma_t(z) = \begin{cases} z & \text{if } t < T, \\ \frac{1}{T}z & \text{if } t = T. \end{cases} \tag{8}$$

Given an input sequence $\mathbf{x}$ of length T, the output of the recurrent layer is

$$\sigma_T\Big(h(x_T) + \sigma_{T-1}\big(h(x_{T-1}) + \sigma_{T-2}(h(x_{T-2}) + \dots)\big)\Big) = \frac{1}{T}\sum_{t=1}^{T} h(x_t) = \mathcal{H}(\mathbf{x}). \tag{9}$$

So far, the neural network computes the histograms $\mathcal{H}(\mathbf{x})$ for given visual words $v_1, \dots, v_K$. In order to train the visual words discriminatively and from scratch, an additional softmax layer with C output units is added to model the class posterior distribution

$$p(c|\mathcal{H}(\mathbf{x})) = \text{softmax}_c(\widetilde{\mathbf{W}}^\intercal \mathcal{H}(\mathbf{x}) + \widetilde{b}). \tag{10}$$

It acts as a linear classifier on the histograms and allows for the application of standard neural network optimization methods for the joint estimation of the visual words and classifier weights. Once the network is trained, the softmax output layer can be discarded and the output of the recurrent layer is used as histogram representation. The complete neural network is depicted in Figure 1.

Note the difference of our method to other supervised learning methods like the restricted Boltzman machine of [9]. Usually, each feature vector x_t extracted from a video gets assigned the class of the respective video. Then, the codebook is optimized to distinguish the classes based on the representations $h(x_t)$. For the actual classification, however, a global video representation $\mathcal{H}(\mathbf{x})$ is used. In our approach, on the contrary, the codebook is optimized to distinguish the classes based on the final representation $\mathcal{H}(\mathbf{x})$ directly rather than on an intermediate quantity $h(x_t)$.

3.3 Equivalence Results

There is a close relation between single-layer neural networks and Gaussian models [10, 18]. We consider the special case of kMeans here. Following the derivation in the previous section, the kMeans model can be transformed into a single-layer neural network. For the other direction, however, the constraint that the bias components are inner products of the weight matrix rows (see Equations (5) and (6)) is not met when optimizing the neural network parameters. In fact, the single-layer neural network is equivalent to a kMeans model with non-uniform visual word priors $p(v_k)$. The visual words and their priors can be obtained from the weight matrix $\mathbf{W}$ and the bias b,

$$v_k = (\mathbf{W}_{1,k} \dots \mathbf{W}_{D,k})^\intercal, \tag{11}$$

$$p_{\text{NN}}(v_k) = \frac{\exp\left(b_k + \frac{1}{2}v_k^\intercal v_k\right)}{\sum_{\tilde{k}}\exp\left(b_{\tilde{k}} + \frac{1}{2}v_{\tilde{k}}^\intercal v_{\tilde{k}}\right)}. \tag{12}$$

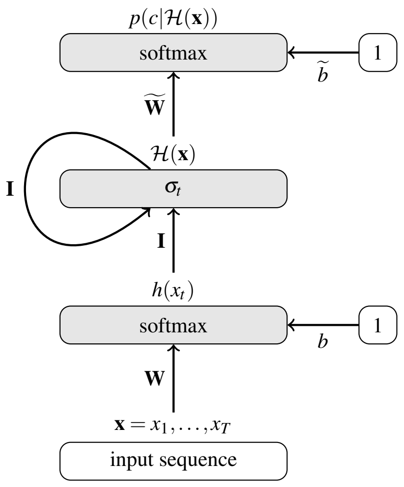


Figure 1: Neural network encoding the BoW model. The output layer is discarded after training and the histograms from the recurrent layer are used for classification in combination with an SVM.

4 Experimental Setup

We extract improved dense trajectories as described in [34], resulting in five descriptors with an overall number of 426 features per trajectory, and apply a z-score normalization to the data. A visual vocabulary and histograms of visual words are computed for each video. An SVM with RBF-χ^2 kernel is then applied to the histograms. We use LIBSVM [4] with a *one-against-rest* approach to address the multiclass problem.

For the baseline, we follow the approach of [34]: kMeans is run eight times on a randomly sampled subset of $100,000$ trajectories. The result with lowest sum of squared distances is used as visual vocabulary.

If the histograms are computed with a neural network, the trajectories of each video are uniformly subsampled to reduce the total amount of trajectories that are used for training. The network is trained according to the cross-entropy criterion, which maximizes the likelihood of the posterior probabilities. We use RProp as optimization algorithm and iterate until the objective function does not improve further. Since the network output is not used directly for classification, overfitting is not a critical issue. Thus, strategies like regularization or dropout do not need to be applied. Furthermore, we could not observe any advantages when initializing with a kMeans model. Normalization, in contrast, is crucial. If the network input is unnormalized, neural network training is highly sensitive to the learning rate and RProp even fails to converge.

The number of visual words is set to $4,000$ for all experiments. The histograms for each of the five descriptors are combined using a multichannel RBF-χ^2 kernel,

$$K(i,j) = \exp\left(-\frac{1}{5}\sum_{c=1}^{5}\frac{D(\mathcal{H}(\mathbf{x}_i^c),\mathcal{H}(\mathbf{x}_j^c))}{A_c}\right), \tag{13}$$

where $\mathbf{x}_i^c$ is the c-th descriptor type of the i-th video, $D(\cdot,\cdot)$ is the χ^2-distance between two histograms, and A_c is the mean distance between all histograms for descriptor c in the training set.

We evaluate our method on four action recognition benchmarks. The datasets range from medium scale to large scale.

The **Olympic Sports** dataset [19] contains 783 action clips of 16 different Olympic disciplines. We follow the setup of [19] and use the suggested split into a training set containing 649 videos and a test set containing 134 videos. Roughly 40 million trajectories are extracted for the training set and 7.8 million for the test set. We report mean average precision.

The **HMDB-51** dataset [13] is a large action recognition dataset containing $6,849$ videos collected from several movies and public databases. The video clips comprise 51 action categories, each of which is represented by at least 101 videos. Three splits into training and test sets are provided by the authors. For each of the splits, 42 million trajectories are extracted for training and 18 million trajectories for testing. We report the average accuracy.

J-HMDB [11] is a subset of HMDB-51 with 928 videos and 21 action classes. The clips are rather short, having a length of 15 to 40 frames. As for HMDB-51, the authors propose three splits, each of them using 70% of the video clips as training data and 30% for testing. For each split, the training data comprises roughly two million trajectories and the test data $750,000(\pm 50,000)$ trajectories. This dataset is the smallest of the four in terms of extracted trajectories. We use it to validate that our method not only works for large scale tasks, but also on small datasets. As for HMDB-51, average accuracy over the three splits is reported.

The **UCF101** dataset [29] is a large scale action recognition dataset with $13,320$ video clips of 101 action classes. The videos are collected from Youtube and contain actions of five types (human-object interaction, body motion only, human-human interaction, playing musical instruments, and sports). The authors suggest three splits, each containing approximately $9,500$ clips for training and $3,700$ for testing. This results in roughly 230 million trajectories for the training set and 90 million trajectories for the test set. We report average accuracy over the three splits.

For Olympic Sports and HMDB-51, we use the human bounding boxes provided by Wang and Schmid.[1] For the other datasets, improved dense trajectories are extracted without human bounding boxes.

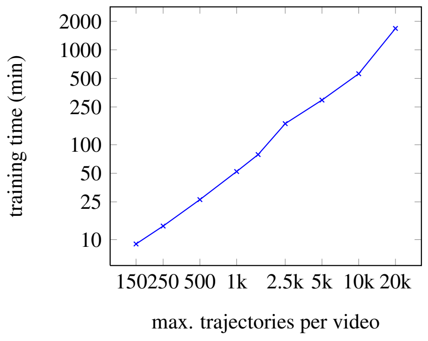
5 Experimental Results

In this section, we evaluate our method empirically. First, the effect of subsampling the trajectories is analyzed. We show that even with a small number of trajectories, satisfying results can be obtained while accelerating the training time by up to two orders of magnitude. Then, we evaluate our method on four action recognition benchmarks and compare it to some well known coding methods on two small image classification datasets. We conclude with a comparison to the current state-of-the-art in action recognition.

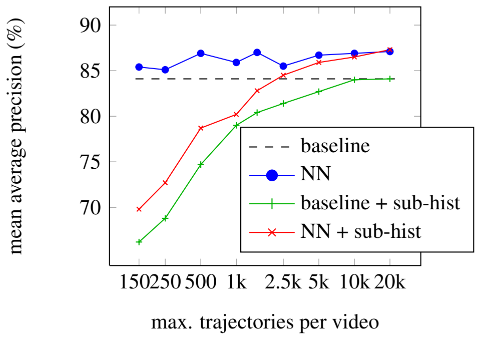
5.1 Effect of Feature Subsampling

We evaluate the runtime and accuracy of our method when reducing the number of trajectories per video on Olympic Sports. The networks are trained on a GeForce GTX 780 with 3GB memory. We limit the number of trajectories per video to values from 150 to $20,000$

[1] http://lear.inrialpes.fr/people/wang/improved_trajectories



(a) Time required to train the neural network. x- and y-axis are in logarithmic scale.



(b) Performance of the neural network (NN) based BoW. For the blue curve, the number of trajectories has only been limited for neural network training. For the red and green curve, it has also been limited for the histogram computation on the training and test set (sub-hist).

Figure 2: Runtime and mean average precision on Olympic Sports when limiting the maximal number of trajectories per video. Note that the x-axes are in logarithmic scale.

Descriptors:	concatenated		separate	
	baseline	neural network	baseline	neural network
Olympic Sports	84.1%	**86.7**%	84.4%	**85.9**%
J-HMDB	56.6%	**57.6**%	59.1%	**61.9**%
HMDB-51	45.8%	**50.6**%	52.2%	**54.0**%
UCF101	67.8%	**73.3**%	73.3%	**76.9**%

Table 1: Comparison of the baseline with the neural network model for concatenated and separate descriptors on four different datasets.

via uniform subsampling. This corresponds to an overall number of trajectories between 100,000 and 12 million. In Figure 2a, the runtime is shown. For 150 trajectories per video, neural network training takes nine minutes, which is about the same time needed for the generation of the visual vocabulary for the baseline using our GPU implementation of kMeans. The training time scales linearly with the number of trajectories per video. In Figure 2b, the performance of the system with limited number of trajectories is illustrated. For the blue curve, the number of trajectories has only been limited for neural network training, but the histograms are computed on all extracted trajectories. The performance of the neural network models is always above the baseline (dashed line). The curve stabilizes around 5,000 trajectories per video, suggesting that this number is sufficient for the neural network based BoW. Note that for kMeans, we observed only small fluctuations around one percent when changing the number of clustered trajectories, the subsampling strategy, or the initialization.

If the histograms for the training and test set are also computed on the limited set of trajectories (red and green curve), the performance of the systems is much more sensitive to the number of subsampled trajectories. However, when more than 5,000 trajectories per video (overall 3.2 million trajectories) are used for the neural network histogram computation, the difference to taking all extracted trajectories is small. Hence, it is possible to achieve sat-

Method	Codebook Size	Caltech-101	15-Scenes
Hard assignment [15]	200	64.6%	81.1%
Kernel Codebooks [33]	200	64.1%	76.7%
Soft assignment [17]	1000	74.2%	82.7%
ScSPM [38]	1024	73.2%	80.3%
LLC [37]	2048	73.4%	-
Multi-way local pooling [2]	1024×65	77.8%	83.1%
Unsupervised SS-RBM [9]	1024	75.1%	84.1%
Ours	**1024**	**74.5%**	**83.5%**

Table 2: Comparison of our method to other coding methods on Caltech-101 and 15-scenes.

isfying results with only 8% of the originally extracted trajectories, allowing to accelerate both, the histogram computation and the feature extraction itself. A similar reduction is also possible with kMeans, but the loss in accuracy is higher.

During training of the neural network, a class posterior distribution $p(c|\mathcal{H}(\mathbf{x}))$ is modeled. Using this model instead of the SVM with RBF-χ^2 kernel for classification is worse than the baseline. For concatenated descriptors, the result on Olympic Sports is 82.3%. Regularization, dropout, and adding additional layers did not yield any improvement. However, considering that the model for $p(c|\mathcal{H}(\mathbf{x}))$ is only a linear classifier on the histograms (*c.f.* Section 3), the result is remarkable as the baseline with a linear SVM reaches only 69.6%.

5.2 Evaluation on Various Datasets

We evaluate our method on the four action recognition datasets Olympic Sports, J-HMDB, HMDB-51, and UCF101. On J-HMDB, all extracted trajectories are used for neural network training. On Olympic Sports and HMDB-51, we limit the number of trajectories per video to $5,000$ as proposed in Section 5.1. For terms of efficiency, we further reduce this number to $2,500$ for the largest dataset UCF101. We conduct the experiments for concatenated descriptors, *i.e.* we directly use a 426 dimensional feature vector for each trajectory, and for separate descriptors as originally proposed in [34]. The results are shown in Table 1.

The neural network outperforms the baseline on all datasets. For the medium-sized datasets J-HMDB and Olympic Sports that have only few classes, the improvement is between 1% and 2.6% in case of concatenated descriptors. For the large datasets, however, the baseline is outperformed by around 5%. In case of separate descriptors, the improvement is smaller but still ranges from 1.5% to 3.6%.

Comparing the neural network with concatenated descriptors (second column of Table 1) and the baseline with separate descriptors (third column of Table 1) reveals that both systems achieve similar accuracies. However, for the baseline with separate descriptors, visual vocabularies and histograms have to be computed for each of the five descriptors separately. For the neural network with concatenated descriptors, in contrast, it is sufficient to train a single system.

Note that the performance gain over the traditional BoW model is due to the fact the codebook for the neural network is trained discriminatively to best separate the classes. Particularly, it is not related to the fact that the neural network implicitly models a non-uniform prior. To validate this hypothesis, we trained a traditional BoW model with a non-uniform prior on Olympic Sports. The mean average precision is 84.0% compared to 84.1% with

Method	HMDB-51	UCF101
Traditional models		
Improved DT + bag of words	52.2%	73.3%
Improved DT + fisher vectors [34, 35] (*)	57.2%	85.9%
Improved DT + LLC	50.8%	71.9%
Stacked fisher vectors [21] (*)	66.8%	-
Multi-skip feature stacking [14] (*)	65.4%	89.1%
Super-sparse coding vector [39]	53.9%	-
Neural networks		
Two-stream CNN [25] (**)	59.4%	88.0%
Slow-fusion spatio-temporal CNN [12] (**)	-	65.4%
Composite LSTM [30]	44.0%	75.8%
Ours	**54.0%**	**76.9%**

Table 3: Comparison of our model to published results on HMDB-51 and UCF101. We also provide results with BoW and LLC encoding as a direct comparison to our method. Methods marked with (*) use Fisher vectors, those marked with (**) use additional training data.

a uniform prior. It can be seen that the non-uniform prior does not lead to a significant difference.

5.3 Application to Image Classification

Although designed to meet some specific problems in action recognition, our method is applicable to image datasets, too. We compare to existing coding methods on two small image datasets, Caltech-101 [8] and 15-scenes [7]. Following the setup of [9], we densely extract SIFT features, compute spatial pyramids, and use a linear SVM for classification. Note that our method is not particularly designed for such a setting since we do not train our encoding directly on the spatial pyramid features that are finally used for classification. In contrast to the methods [2, 9, 37, 38], we do not introduce any sparsity constraints. Still, our method shows competitive results compared to several other coding methods, see Table 2.

5.4 Comparison to State of the Art

In Table 3, our method is compared to the state-of-the-art on HMDB-51 and UCF101. Our approach outperforms other approaches based on BoW, sparse coding [39], locality-constrained linear coding (LLC) [37], or neural networks [12, 30]. The approach [25] is not directly comparable since the accuracy is mainly boosted by the use of additional training data. Only the methods that use Fisher vectors achieve a better accuracy than our method. However, extracting Fisher vectors is more expensive in terms of memory than a BoW model. If Fisher vectors are extracted per frame, the storage of the features would require around 1 TB for the 2.4 million frames of UCF101 compared to 35 GB for our method. For applications with memory and runtime constraints, our approach is a very useful alternative.

6 Conclusion

We have introduced a neural network model that enables discriminative and supervised training of a visual vocabulary directly on video level rather than on descriptor level only. The model is equivalent to the standard BoW model and differs only in the way it is trained. Our model outperforms the baseline by two to five percent points on four state-of-the-art action recognition datasets and allows for a significant reduction in the amount of extracted features, and thus in training and test time, without a considerable loss of performance. Moreover, we have shown the applicability of our model to image classification and achieved competitive results compared to other coding methods, although no additional constraints like sparsity are required.

Acknowledgements: Authors acknowledge financial support from the DFG Emmy Noether program (GA 1927/1-1).

References

[1] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 2559–2566, 2010.

[2] Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun. Ask the locals: multi-way local pooling for image recognition. In *Int. Conf. on Computer Vision*, pages 2651–2658, 2011.

[3] Hongping Cai, Fei Yan, and Krystian Mikolajczyk. Learning weights for codebook in image classification and retrieval. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 2320–2327, 2010.

[4] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–27, 2011.

[5] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, European Conf. on Computer Vision*, 2004.

[6] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*, 2014.

[7] Li Fei-Fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 524–531, 2005.

[8] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, pages 59–70, 2007.

[9] Hanlin Goh, Nicolas Thome, Matthieu Cord, and Joo-Hwee Lim. Unsupervised and supervised visual codes with restricted boltzmann machines. In *European Conf. on Computer Vision*, pages 298–311, 2012.

[10] Georg Heigold, Ralf Schlüter, and Hermann Ney. On the equivalence of gaussian hmm and gaussian hmm-like hidden conditional random fields. In *Interspeech*, pages 1721–1724, 2007.

[11] Hueihan Jhuang, Jürgen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J. Black. Towards understanding action recognition. In *Int. Conf. on Computer Vision*, pages 3192–3199, 2013.

[12] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[13] Hilde Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: a large video database for human motion recognition. In *Int. Conf. on Computer Vision*, pages 2556–2563, 2011.

[14] Zhenzhong Lan, Ming Lin, Xuanchong Li, Alexander G Hauptmann, and Bhiksha Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. *arXiv preprint arXiv:1411.6660*, 2014.

[15] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 2169–2178, 2006.

[16] Xiao-Chen Lian, Zhiwei Li, Changhu Wang, Bao-Liang Lu, and Lei Zhang. Probabilistic models for supervised dictionary learning. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 2305–2312, 2010.

[17] Lingqiao Liu, Lei Wang, and Xinwang Liu. In defense of soft-assignment coding. In *Int. Conf. on Computer Vision*, pages 2486–2493, 2011.

[18] Wolfgang Macherey and Hermann Ney. A comparative study on maximum entropy and discriminative training for acoustic modeling in automatic speech recognition. In *Interspeech*, 2003.

[19] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *European Conf. on Computer Vision*, pages 392–405, 2010.

[20] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. Action and event recognition with fisher vectors on a compact feature set. In *Int. Conf. on Computer Vision*, pages 1817–1824, 2013.

[21] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action recognition with stacked fisher vectors. In *European Conf. on Computer Vision*, pages 581–595, 2014.

[22] Florent Perronnin, Christopher Dance, Gabriela Csurka, and Marco Bressan. Adapted vocabularies for generic visual categorization. In *European Conf. on Computer Vision*, pages 464–475, 2006.

[23] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *European Conf. on Computer Vision*, pages 143–156, 2010.

[24] Kishore K Reddy and Mubarak Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24:971–981, 2013.

[25] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[26] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep fisher networks for large-scale image classification. In *Advances in Neural Information Processing Systems*, pages 163–171, 2013.

[27] Josef Sivic and Andrew Zisserman. Video google: a text retrieval approach to object matching in videos. In *Int. Conf. on Computer Vision*, pages 1470–1477, 2003.

[28] Josef Sivic, Bryan C Russell, Alexei A Efros, Andrew Zisserman, and William T Freeman. Discovering object categories in image collections. Technical report, Massachusetts Institute of Technology, 2005.

[29] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[30] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 2015.

[31] Vladyslav Sydorov, Mayu Sakurada, and Christoph H Lampert. Deep fisher kernels–end to end learning of the fisher kernel gmm parameters. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 1402–1409, 2014.

[32] Ekaterina H Taralova, Fernando De la Torre, and Martial Hebert. Motion words for videos. In *European Conf. on Computer Vision*, pages 725–740, 2014.

[33] Jan van Gemert, Cor Veenman, Arnold Smeulders, and Jan-Mark Geusebroek. Visual word ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32: 1271–1283, 2010.

[34] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Int. Conf. on Computer Vision*, pages 3551–3558, 2013.

[35] Heng Wang and Cordelia Schmid. Lear-inria submission for the thumos workshop. In *ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013.

[36] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal on Computer Vision*, 103:60–79, 2013.

[37] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 3360–3367, 2010.

[38] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 1794–1801, 2009.

[39] Xiaodong Yang and YingLi Tian. Action recognition using super sparse coding vector with spatio-temporal awareness. In *European Conf. on Computer Vision*, 2014.

[40] Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal on Computer Vision*, 73:213–238, 2007.