Learning Smooth Pooling Regions for Visual Recognition

Mateusz Malinowski
mmalinow@mpi-inf.mpg.de
Mario Fritz
mfritz@mpi-inf.mpg.de

Scalable Learning and Perception
Max Planck Institute for Informatics
Saarbrücken, Germany

Abstract

From the early HMAX model to Spatial Pyramid Matching, spatial pooling has played an important role in visual recognition pipelines. By aggregating local statistics, it equips the recognition pipelines with a certain degree of robustness to translation and deformation yet preserving spatial information. Despite of its predominance in current recognition systems, we have seen little progress to fully adapt the pooling strategy to the task at hand. In this paper, we propose a flexible parameterization of the spatial pooling step and learn the pooling regions together with the classifier. We investigate a smoothness regularization term that in conjuncture with an efficient learning scheme makes learning scalable. Our framework can work with both popular pooling operators: sum-pooling and max-pooling. Finally, we show benefits of our approach for object recognition tasks based on visual words and higher level event recognition tasks based on object-bank features. In both cases, we improve over the hand-crafted spatial pooling step showing the importance of its adaptation to the task.

1. Introduction

Spatial pooling plays a crucial role in modern object recognition and detection systems. Motivated from biology [7, 9, 19, 25] and statistics of locally orderless images [12], the spatial pooling approach has been found useful as an intermediate step of many today's computer vision methods ranging from local features based approaches [16, 30] to higher-level semantic representations [22]. In order to form more robust features under translation or small object deformations, activations of codes and features are pooled over larger areas in a spatial pyramid scheme [16, 30] via a sum or max operator. Unfortunately, this critical decision, namely the spatial division, is most prominently based on hand-crafted layouts and therefore data and task independent.

We propose a flexible parameterization that allows for a richer set of possible pooling regions and show results on classification tasks using two different pipelines [2, 22]. Moreover, we extend the learnable pooling regions [23] to the events recognition task with object banks as high level features. The representation is learned jointly with the classifier to support the recognition task. In order to deal with the increased flexibility of the model, we investigate different regularizers and efficient learning schemes. In particular, we propose a smoothness regularizer that yields the strongest performance improvements in our experiments.

Related Work There is an increasing interest to push the boundary of learning based approaches towards fully optimized and adaptive architectures where design choices, that would potentially constrain or bias a model, are kept to a minimum. Neural networks have a great tradition of approaching hierarchical learning problems and training intermediate representations [18, 24]. Along this line, we propose a learnable spatial pooling strategy that can discriminatively shape the pooling regions. In contrast to convolutional neural architectures [24], our particular architecture has a direct interpretation as a global pooling strategy and therefore subsumes popular spatial pyramids as a special case. Yet we have the freedom to investigate different regularization terms that lead to new pooling strategies when optimized jointly with the classifier.

Recent progress has been made in learning pooling regions in the context of image classification using the Spatial Pyramid Matching (SPM) pipeline [16, 30]. Some researchers [6, 10, 11, 13, 14, 26, 27] have further investigated how to liberate the recognition from preconceptions of the hand crafted recognition pipelines. However, these methods still make quite strong assumptions on the solutions that can be achieved. For instance Jia and Huang [10] optimizes binary pooling strategies that are given by the superposition of rectangular basis functions, and Feng et al. [6] finds pooling regions by applying a linear discriminant analysis for individual pooling strategies and training a classifier afterwards. Krapac et al. [14] and Koniusz and Mikolajczyk [13] model spatial location of the visual words by fitting Mixture of Gaussians. Russakovsky et al. [26] and Sánchez et al. [27] have shown improvement over SPM by pooling the objects and background separately. Although the last two methods are image-dependent they strongly depend on the object localization which is a non-trivial task if bounding boxes are absent during training time. In contrast, our method learns the shape of the pooling region without resorting to the notion of the bounding boxes. However both Russakovsky et al. [26] and Sánchez et al. [27] can be combined with our approach as they are complementary. Our method is also complementary to van Gemert [29] which exploits bias in the photographic style and generalizes SPM to quantize and pool over such attributes as colorfulness, depth of field, viewpoint, lighting, and saliency. In contrast, we learn the pooling regions directly without the use of such additional cues.

Outline First, we propose our parameterized pooling operator and show how to jointly optimize the parameters together with the classifier. To cope with the large number of parameters, we investigate regularizers and an efficient learning scheme. We evaluate our method on the CIFAR-10 and show strong improvements in the regime of small dictionaries where our flexible model shows its capability to make best use of the representation by exploring spatial pooling strategies specific to every coordinate of the code. We also show strong classification performance on the CIFAR-100 dataset where our method outperforms, to the best of our knowledge, the state-of-the-art on this dataset in the regime of spatial pyramid architectures. Finally, we also apply our model to higher level events classification tasks that utilize a representation based on object-bank features [22].

2. Method

In contrast to the methods that use fixed spatial pooling regions in the object classification task (e.g. [16, 30]) our method jointly optimizes both the classifier and the pooling regions. In this way, the learning signal available in the classifier can help shaping the pooling regions in order to arrive at better pooled features.

2.1. Parameterized pooling operator

The simplest form of the spatial pooling is computing histogram over the whole image. This can be expressed as $\Sigma(U) := \sum_{j=1}^{M} u_j$, where $u_j \in \mathbb{R}^K$ is an encoded patch extracted from the image (out of M such codes) and an index j refers to the spatial location that the code originates from[1]. Another popular pooling scheme that has been proven successful [30] is max-pooling: $\mathbb{M}(U) := \max_{j=1}^{M} u_j$. Since the pooling approach looses spatial information of the codes, Lazebnik et al. [16] proposed to first divide the image into subregions, and afterwards to create pooled features by concatenating histograms computed over each subregion. There are two problems with such an approach: first, the division is largely arbitrary and in particular independent of the data; second, discretization artifacts occur as spatially nearby codes can belong to two different regions as the 'hard' division is made.

In this paper we address both problems by using a parameterized version of the pooling operator

$$\Theta_w(U) := \rho_{j=1}^{M}(w_j \circ u_j) \tag{1}$$

where $a \circ b$ is the element-wise multiplication, and ρ is a pooling function. Here, we investigate either sum or max pooling functions and therefore $\rho \in \{\max, \sum\}$. Standard spatial division of the image can be recovered from Formula 1 by setting the vectors w_j either to a vector of zeros 0, or ones 1. For instance, features obtained from dividing the image into 2 subregions using sum pooling can be recovered from Θ by concatenating two vectors: $\sum_{j=1}^{\frac{M}{2}} 1 \circ u_j + \sum_{j=\frac{M}{2}+1}^{M} 0 \circ u_j$, and $\sum_{j=1}^{\frac{M}{2}} 0 \circ u_j + \sum_{j=\frac{M}{2}+1}^{M} 1 \circ u_j$, where $\left\{1, ..., \frac{M}{2}\right\}$ and $\left\{\frac{M}{2}+1, ..., M\right\}$ refer to the first and second half of the image respectively.

In general, let $\mathfrak{F} := \{\Theta_w\}_w$ be a family of the pooling functions given by Eq. 1, parameterized by the vector w, and let $w^{*,l}$ be the 'best' parameter chosen from the family $\mathfrak{F}$ based on the initial configuration l and a given set of images. First row of Table 2 shows four initial configurations that mimic the standard 2-by-2 spatial image division. Every initial configuration can lead to different $w^{*,l}$ as it is shown in Table 2. Clearly, the family $\mathfrak{F}$ contains all possible 'soft' and 'hard' spatial divisions of the image, and therefore is their generalization.

2.2. Learnable pooling regions

In the SPM architectures the pooling weights w are designed by hand, whereas here we aim for joint learning w together with the parameters of the classifier. Intuitively, the classifier during training has access to the classes that the images belong to, and therefore can shape the pooling regions. On the other hand, the method aggregates statistics of the codes over such learned regions and pass them to the classifier allowing to achieve higher accuracy. Such joint training of the classifier and the pooling regions can be done by adapting the backpropagation algorithm [1, 20], and so can be interpreted as a densely connected multilayer perceptron [1, 4].

Consider a sampling scheme and an encoding method producing M codes each K dimensional. Every coordinate of the code is an input layer for the multilayer perceptron. Then we connect every j-th input unit at the layer k to the l-th pooling unit a_l^k via the relation $w_{lj}^k u_j^k$. Since the receptive field of the pooling unit a_l^k consists of all codes at the layer k, we have $a_l^k := \sum_{j=1}^{M} w_{lj}^k u_j^k$ or $a_l^k := \max_{j=1}^{M} w_{lj}^k u_j^k$, and so in the vector notation

$$a_l := \rho_{j=1}^{M}(w_j^l \circ u_j) = \Theta_{w^l}(U) \tag{2}$$

[1]That is $j = (x, y)$ where x and y refer to the spatial location of the center of the extracted patch.

Next, we connect all pooling units with the classifier allowing the information to circulate between the pooling layers and the classifier. We use logistic regression which is connected to the pooling units via the formula

$$J(\Theta) := -\frac{1}{D}\sum_{i=1}^{D}\sum_{j=1}^{C} 1\{y^{(i)} = j\} \log p(y^{(i)} = j | a^{(i)}; \Theta) \tag{3}$$

where D denotes the number of all images, C is the number of all classes, $y^{(i)}$ is a label assigned to the i-th input image, and $a^{(i)}$ are responses from the 'stacked' pooling units $[a_l]_l$ for the i-th image[2]. We use the logistic function to represent the probabilities: $p(y = j|x;\Theta) := \frac{\exp(\theta_j^T x)}{\sum_{l=1}^{C}\exp(\theta_l^T x)}$. Since the classifier is connected to the pooling units, our task is to learn jointly the pooling parameters W together with the classifier parameters Θ, where W is the matrix containing all pooling weights. Finally, we use standard gradient descent algorithm that updates the parameters using the following fixed point iteration

$$X^{t+1} := X^t - \gamma \nabla J(X^t) \tag{4}$$

where in our case X is a vector consisting of the pooling parameters W and the classifier parameters Θ.

2.3. Regularization terms

In order to improve the generalization, we introduce regularization of our model as we deal with a large number of the parameters. For the classification Θ and pooling parameters W, we employ L_2 regularization terms: $||\Theta||_{l_2}^2$ and $\sum_k ||W^k||_{l_2}^2$. In order to maintain interpretable pooling regions we constraint the solution to the unit cube. This is implemented via projects onto the cube during the optimization. To reduce quantization artifacts of the pooling strategy as well as to ensure smoothness of the output w.r.t. small translations of the image, the model penalizes weights whenever the pooling region is non-smooth. This can be done by measuring the spatial variation $||\nabla_x W^k||_{l_2}^2 + ||\nabla_y W^k||_{l_2}^2$ for every layer k. Therefore our overall optimization objective is

$$\begin{aligned} &\underset{W,\Theta}{\text{minimize}}\, J_{\mathrm{R}}(\Theta, W) := \\ &-\frac{1}{D}\sum_{i=1}^{D}\sum_{j=1}^{C} 1\{y^{(i)} = j\} \log p(y^{(i)} = j | a^{(i)}; \Theta) \\ &+ \frac{\alpha_1}{2}||\Theta||_{l_2}^2 + \frac{\alpha_2}{2}||W||_{l_2}^2 \\ &+ \frac{\alpha_3}{2}\left(||\nabla_x W||_{l_2}^2 + ||\nabla_y W||_{l_2}^2\right) \\ &\text{subject to } W \in [0,1]^{K\times M\times L} \end{aligned} \tag{5}$$

where a_l is the l-th pooling unit described by Formula 2, and $||W||_{l_2}$ is the Frobenius norm.

[2]Providing the codes $U^{(i)}$ are collected from the i-th image and $a_l^{(i)} := \Theta_{w^l}(U^{(i)})$ then $a^{(i)} := [a_l^{(i)}]_l$.

2.4. Approximation of the model

The presented approach is demanding to train in the means of the CPU time and memory storage when using high dimensional representations. The number of the pooling parameters to learn grows as $K \times M \times L$, where K is dimensionality of codes, M is the number of patches taken from the image and L is the number of pooling units. Therefore, we propose two approximations to our method making the whole approach scalable to large dictionaries. However, we emphasize that learned pooling regions have very little if any overhead compared to standard spatial division approaches at test time.

The first approximation does a fine-grained spatial partition of the image (3 by 3 pixels), and then pools the codes over such subregions. This operation reduces the number of spatial locations by the factor of the pre-pooling size. The second approximation divides a K dimensional code into $\frac{K}{D}$ batches, each D dimensional. Then we train our model on all such batches in parallel to obtain the pooling weights. Afterwards, we train the classifier on top of the concatenation of the trained, partial models. We also consider a redundant set of such batches in our experiments in order to compensate for potential approximation errors. As opposed to the approximations proposed by Le et al. [17], our training is fully parallel and doesn't need communication between different batches/machines. In addition, the training of the small models per batch shows on average 5 times faster convergence than the full models.

Implementation details To learn the parameters of the model we use the limited-memory BFGS algorithm[3]. The hyperparameters were selected by 5-fold cross-validation. Our implementation is available at http://www.d2.mpi-inf.mpg.de/datasets.

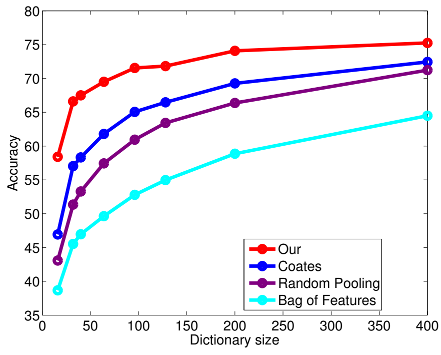
3. Experimental Results

First, we evaluate our method on the CIFAR-10 and CIFAR-100 object recognition datasets [15]. Furthermore, we provide insights into the learned pooling strategies as well as investigate transfer between datasets. Second, we show that our method also translates to a high level recognition task of events in a max pooling setting with object bank features [22] on the UIUC sports events dataset. [21]. We start by describing our experimental setup.

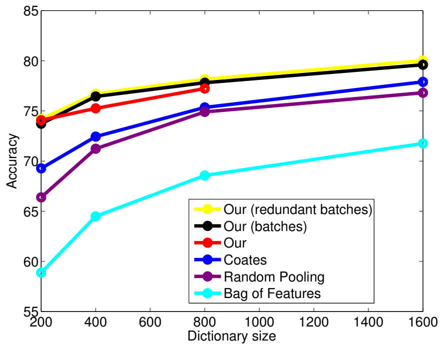
Datasets The CIFAR-10 and CIFAR-100 datasets contain 50000 training color images and 10000 test color images from respectively 10 and 100 categories, with 6000 and 600 images per class respectively. All images have the same size: 32×32 pixels, and were sampled from the 80 million tiny images dataset [28]. UIUC sports events [21] is a dataset containing 8 sports categories such as rowing, badminton, polo, bocce, snowboarding, croquet, sailing, and rock climbing. The number of images varies per class from 137 to 250. We follow Li-Jia et al. [22] and use 70 images per class for training, and 60 images per class for testing.

Feature representations In order to insure comparability we follow the evaluation pipeline of Coates and Ng [2] for the object recognition experiment. We extract normalized and whitened 6×6 patches from images using a dense, equispaced grid with a unit sample spacing. As the next step, we employ the K-means assignment and triangle encoding [2, 3] to compute codes – a K-dimensional representation of the patch. As we want to be comparable to Coates et al. [3], who uses a spatial division into 2-by-2 subregions which results in $4 \cdot K$ pooled features, we use 4 pooling units, too. Furthermore, we use a standard division

[3]implementation by Mark Schmidt: http://www.di.ens.fr/~mschmidt/Software/minFunc.html



(a)



(b)

Figure 1: Figure 1(a) shows accuracy of the classification with respect to the number of dictionary elements on smaller dictionaries. Figure 1(b) shows the accuracy of the classification for bigger dictionaries when batches, and the redundant batches were used. Experiments are done on CIFAR-10.

(first row of Table 2) as an initialization of our model. In addition to the Coates and Ng [2] pipeline, we also apply our architecture to max pooling and object banks [22]. The latter use object filters [5] and spatial pyramid matching [16, 30] to build a high-level representation of the image. For both feature representations we use the source code provided by the authors.

Evaluation of our method on small dictionaries Figure 1(a) shows the classification accuracy of our full method against the baseline [2]. Since we train the pooling regions without any approximations in this set of experiments the results are limited to dictionary sizes up to 800. Our method outperforms the approach of Coates by 10% for dictionary size 16 (our method achieves the accuracy 57.07%, whereas the baseline only 46.93%). This improvement is consistent up to the bigger dictionaries although the margin is getting smaller. Our method is about 2.5% and 1.88% better than the baseline for 400 and 800 dictionary elements respectively.

Scaling up to sizable dictionaries In subsection 2.4 we have discussed how to divide the codes into low dimensional batches and learn the pooling regions on those. In the following experiments we use batches with 40 entries extracted from the original code, as those fit conveniently into the memory of a single, standard machine (about 5 Gbytes for the main data) and can all be trained in parallel.

Besides a reduction in the memory requirements, the batches have shown multiple benefits in practice due to smaller number of parameters. We need less computations per iterations as well as observe faster convergence. Figure 1(b) shows the classification performance for larger dictionaries where we examined the full model [Our], the baseline [Coates], random pooling regions (described in subsection 3), bag of features, and two possible approximation - the batched model [Our (batches)], and the redundantly batched model [Our (redundant batches)].

Our test results are presented in Table 1. We observe little if any drop in accuracy when using our approximation scheme. We attribute this to the better conditioned learning problem of the smaller codes within one batch. With an accuracy for the batched model of 79.6% we

Method	Dict. size	Features	Acc.
Jia	1600	6400	80.17%
Coates	1600	6400	77.9%
Our (batches)	1600	6400	79.6%
Our (redundant)	1600	12800	80.02%

Table 1: Comparison of our methods against the baseline [2] and Jia and Huang [10] with respect to the dictionary size, number of features and the test accuracy on CIFAR-10.

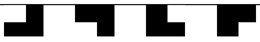
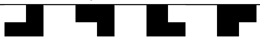
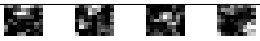
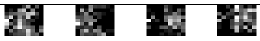
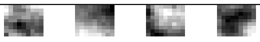
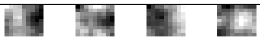
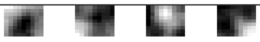
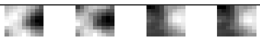
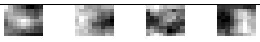
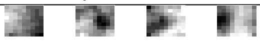
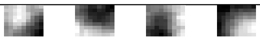
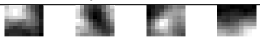
regularization	pooling weights	
dataset: CIFAR-10 ; dictionary size: 200		
Coates (no learn.)		
l2		
smooth		
smooth & l2		
dataset: CIFAR-10 ; dictionary size: 1600		
smooth & batches		
dataset: CIFAR-100 ; dictionary size: 1600		
smooth & batches		

Table 2: Visualization of different pooling strategies obtained for different regularizations, datasets and dictionary size. Every column shows the regions from two different coordinates of the codes. First row presents the initial configuration also used in standard hand-crafted pooling methods. Brighter regions denote larger weights.

outperform the Coates baseline by 1.7%. Interestingly, we gain another small improvement to 80.02% by adding redundant batches which amounts to a total improvement of 2.12% compared to the baseline. Our method performs comparable to the pooling strategy of Jia and Huang [10] which uses more restrictive assumptions on the pooling regions and employs feature selection algorithm.

To the best of our knowledge Goodfellow et al. [8] achieves the best results on the CIFAR-10 dataset with an accuracy 90.62% with a method based on convolutional maxout networks architecture and data augmentation – different from global pooling architectures that we investigate in our study.

Random pooling regions Our investigation also includes results using random pooling regions where the weights for the parameterized operator (Eq. 2) were sampled from normal distribution with mean 0.5 and standard deviation 0.1, that is $w^l_j \sim \mathcal{N}(0.5, 0.1)$ for all l. This notion of the random pooling differs from the Jia et al. [11] where random selection of rectangles is used. The experiments show that the random pooling regions can compete with the standard spatial pooling (Figure 1(a) and 1(b)) on the CIFAR-10 dataset, and suggest that random projection can still preserve some spatial information. This is especially visible in the regime of bigger dictionaries where the difference is only 1.09%. The obtained results indicate that hand-crafted division of the image into subregions is questionable, and call for a learning-based approach.

Investigation of the regularization terms Our model (Eq. 5) comes with two regularization terms associated with the pooling weights, each imposing different assumptions on the

Regularization	CV Acc.	Test Acc.
free	68.48%	69.59%
l2	67.86%	68.39%
smooth	73.36%	73.96%
l2 + smooth	70.42%	70.32%

Table 3: We investigate the impact of the regularization terms on the CIFAR-10 dataset with dictionary size equals to 200. Term "free" denotes the objective function without both regularization terms. The cross-validation accuracy and test accuracy are shown.

Method	Dict. size	Features	Acc.
Jia	1600	6400	54.88%
Coates	1600	6400	51.66%
Our (batches)	1600	6400	56.29%

Table 4: The classification accuracy on CIFAR-100, where our method is compared against the Coates and Ng [2] and Jia and Huang [10].

pooling regions. Hence, it is interesting to investigate their role in the classification task by considering all possible subsets of $\{l2, \text{smooth}\}$, where "l2" and "smooth" refer to $||W||_{l_2}^2$ and $\left(||\nabla_x W||_{l_2}^2 + ||\nabla_y W||_{l_2}^2\right)$ respectively. Table 3 shows our results on CIFAR-10. We choose a dictionary size of 200 for these experiments, so that we can evaluate different regularization terms without any approximations. We conclude that the spatial smoothness regularization term is crucial to achieve a good predictive performance of our method whereas the l2-norm term can be left out, and thus also reducing the number of hyper-parameters. Based on the cross-validation results (second column of Table 3), we select this setting for further experiments.

Experiments on the CIFAR-100 dataset We also investigate how the model performs on more demanding CIFAR-100 dataset with 100 classes. Our model with the spatial smoothness regularization term on the 40 dimensional batches achieves 56.29% accuracy. To our best knowledge, this result constitutes the state-of-the-art performance on this dataset in the regime of SPM architecture, outperforming Jia and Huang [10] by 1.41%, and the baseline by 4.63%. Non-global pooling schemes like the convolutional max-out networks have recently achieved a performance of up to 61.43% [8].

Transfer of the pooling regions between datasets Beyond the standard classification task, we also examine if the learned pooling regions are transferrable between datasets. In this scenario the pooling regions are first trained on the source dataset and then used on the target dataset to train a new classifier. We use dictionary of 1600 with 40-dimensional batches. Our results (Table 5) suggest that the learned pooling regions are indeed transferable between both datasets. While we observe a decrease in performance when learning the pooling strategy on the less diverse CIFAR-10 dataset, we do see improvements for learning on the richer CIFAR-100 dataset. We arrive at a test accuracy of 80.35% which is an additional improvement of 0.75% and 0.18% over our best results (batch-based approximation) and Jia and Huang [10] respectively.

Visualization and analysis of pooling strategies Table 2 visualizes different pooling strategies investigated in this paper. The first row shows the widely used rectangular spatial di-

Source	Target	Accuracy
CIFAR-10	CIFAR-100	52.86%
CIFAR-100	CIFAR-10	80.35%

Table 5: We train the pooling regions on the 'Source' dataset. Next, we use such regions to train the classifier on the 'Target' dataset where the test accuracy is reported.

	UIUC sports
Object Banks + SPM [22]	76.3%
Object Banks + our method	79.4%

Table 6: Our approach described in section 2 with max pooling function and object banks.

vision of the image. The other visualizations correspond to pooling weights discovered by our model using different regularization terms, datasets and dictionary size. The second row shows the results on CIFAR-10 with the "l2" regularization term. The pooling is most distinct from the other results, as it learns highly localized weights. This pooling strategy has also performed the worst in our investigation (Table 3). The "smooth" pooling performs the best. We see that weights are localized but vary smoothly over the image. The weights expose a bias towards initialization shown in the first row. All methods with the spatial smoothness regularization tend to focus on similar parts of the image, however "l2 & smooth" is more conservative in spreading out the weights. The last two rows show weights trained using our approximation. Visual inspection shows a similar level of localization and smoothness to the regions obtained without approximation. This further supports the use of our division into independent batches.

Results using object banks Lastly, we investigate event recognition on the UIUC Sports database based on object bank features. Li-Jia et al. [22] proposes a spatial pyramid matching architecture on top of the object bank features – which makes it an application target for our learned pooling regions. Please note that this setting is quite different form the previous task as high level event recognition is addressed and we optimize pooling regions in a max pooling context. In the experiments we use 4 pooling units with max pooling function on top of the response maps from the object bank filters [5, 22]. Our results (Table 6) show the importance of adaptive approaches also in this high level recognition context. We improve the results from [22] that use a hand crafted SPM architecture by 3.1%.

4 Conclusion

In this paper we propose a flexible parameterization of global pooling operators which can be trained jointly with the classifier. We study the effect of different regularizers showing the importance of the smoothness. To train the large set of parameters we propose approximations to our model allowing efficient and parallel training without loss of accuracy. Our method outperforms popular hand-crafted pooling-based methods. While our improvements are consistent over the whole range of dictionary sizes, the margin is most impressive for small dictionaries with the improvement up to 10% compared to the baseline [2]. Finally, we apply our method and improve over SPM to high level event recognition using object-banks representation. We believe that our method is a flexible framework to further investigate different pooling strategies and is broadly applicable in spatial pooling architectures.

References

[1] C. M. Bishop. *Neural Network for Pattern Recognition*. Oxford University Press, 1999.

[2] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, 2011.

[3] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

[4] R. Collobert and S. Bengio. Links between perceptrons, mlps and svms. In *ICML*, 2004.

[5] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[6] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric lp-norm feature pooling for image classification. In *CVPR*, 2011.

[7] K. Fukushima and S. Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, 15(6):455–469, 1982.

[8] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013.

[9] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

[10] Y. Jia and C. Huang. Beyond spatial pyramids: Receptive field learning for pooled image features. In *NIPS Workshop on Deep Learning*, 2011.

[11] Y. Jia, C. Huang, and T. Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *CVPR*, 2012.

[12] J. J. Koenderink and A. J. Van Doorn. The structure of locally orderless images. *International Journal of Computer Vision*, 31(2):159–168, 1999.

[13] P. Koniusz and K. Mikolajczyk. Spatial coordinate coding to reduce histogram representations, dominant angle and colour pyramid match. In *ICIP*, 2011.

[14] J. Krapac, J. Verbeek, and F. Jurie. Modeling spatial layout with fisher vectors for image categorization. In *ICCV*, 2011.

[15] A. Krizhevsky and G. Hinton. Convolutional deep belief networks on cifar-10. Technical report, 2010.

[16] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[17] Q. V. Le, R. Monga, M. Devin, G. Corrado, K. Chen, M. A. Ranzato, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. 2012.

[18] Q. V. Le, M. A. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

[19] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990.

[20] Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient backprop. *Neural networks: Tricks of the trade*, pages 546–546, 1998.

[21] L. Li-Jia and L. Fei-Fei. What, where and who? classifying events by scene and object recognition. In *ICCV*, 2007.

[22] L. Li-Jia, S. Hao, E. P. Xing, and L. Fei-Fei. Object bank: A high-level image representation for scene classification and semantic feature sparsification. In *NIPS*, 2010.

[23] M. Malinowski and M. Fritz. Learnable pooling regions for image classification. In *ICLR Workshop*, 2013.

[24] M. A. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007.

[25] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2009.

[26] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei. Object-centric spatial pooling for image classification. In *ECCV*. 2012.

[27] J. Sánchez, F. Perronnin, and T. de Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 2012.

[28] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *PAMI*, 2008.

[29] J. C. van Gemert. Exploiting photographic style for category-level image classification by generalizing the spatial pyramid. In *ICMR*, 2011.

[30] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.