Beyond Spatial Pyramids: Receptive Field Learning for Pooled Image Features

Yangqing Jia[1] Chang Huang[2] Trevor Darrell[1]
[1]UC Berkeley EECS & ICSI [2]NEC Labs America
[1]{jiayq,trevor}@eecs.berkeley.edu [2]chuang@sv.nec-labs.com

Abstract

In this paper we examine the effect of receptive field designs on classification accuracy in the commonly adopted pipeline of image classification. While existing algorithms usually use manually defined spatial regions for pooling, we show that learning more adaptive receptive fields increases performance even with a significantly smaller codebook size at the coding layer. To learn the optimal pooling parameters, we adopt the idea of over-completeness by starting with a large number of receptive field candidates, and train a classifier with structured sparsity to only use a sparse subset of all the features. An efficient algorithm based on incremental feature selection and retraining is proposed for fast learning. With this method, we achieve the best published performance on the CIFAR-10 dataset, using a much lower dimensional feature space than previous methods.

1. Introduction

State-of-the-art category-level image classification algorithms usually adopt a local patch based, multiple-layer pipeline to find good image features. Many methods start from local image patches using either normalized raw pixel values or hand-crafted descriptors such as SIFT [22] or HOG [10], and encode them into an overcomplete representation using various algorithms such as K-means or sparse coding. After coding, global image representations are formed by spatially pooling the coded local descriptors [33, 2, 4]. Such global representations are then fed into nonlinear classifiers [21] or linear classifiers [33], with the latter being more popular recently due to their computation efficiency. Methods following such a pipeline have achieved competitive performance on several challenging classification tasks, such as Caltech-101 and Pascal VOC [11].

During the last decade, much emphasis has been directed at the coding step. Dictionary learning algorithms have been discussed to find a set of basis that reconstructs local image patches or descriptors well [23, 9], and several encoding methods have been proposed to map the original data to a high-dimensional space that emphasizes certain properties, such as sparsity [24, 33, 34] or locality [31]. Recent papers [6, 28, 9] have explored the relationship between dictionary learning and encoding, and have proposed simple yet effective approaches that achieve competitive results. The neuroscience justification of coding comes from simple neurons in the human visual cortex V1, which have been believed to produce sparse and overcomplete activations [24].

Similarly, the idea of spatial pooling dates back to Hubel's seminal paper about complex cells in the mammalian visual cortex [13], which identifies mid-level image features that are invariant to small spatial shifting. The spatial invariance property also reflects the concept of locally orderless images [17], which suggests that low-level features are grouped spatially to provide information about the overall semantics. Most recent research on spatial pooling aims to find a good pooling operator, which could be seen as a function that produces informative statistics based on local features in a specific spatial area. For example, average and max pooling strategies have been found in various algorithms respectively, and systematic comparisons between such pooling strategies have been presented and discussed in [2, 4]. Recently, Coates et al. proposed to pool over multiple features in the context of deep learning [7].

However, relatively little effort has been put into better designs or learning of better spatial regions for pooling, although it has been discussed in the context of learning local descriptors [32]. A predominant approach to define the spatial regions for pooling, which we will also call the receptive fields (borrowing the terminology from neuroscience) for the pooled features, comes from the idea of spatial pyramids [21, 33], where regular grids of increasing granularity are used to pool local features. The spatial pyramids provide a reasonable cover over the image space with scale information, and most existing classification methods either use them directly, or use slightly modified/simplified versions.

In this paper, we ask the question *"are spatial pyramids optimal for image classification?"*, the answer to which is often neglected by existing algorithms. While a pyramid of regions succeeds in providing us information about the spatial layout of image features, one can reasonably ques-

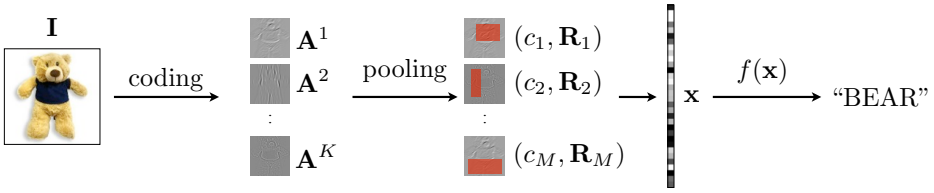

Figure 1. The image classification pipeline. See Section 2 for details.

tion their optimality, as the grid structure may not be adaptive enough to fit the spatial statistics of natural images. As a simple example, to distinguish most indoor and outdoor scenes, a human may look for the existence of the horizon, which could be captured by thin horizontal pooling regions over the image. Spatial grids, even with a pyramid structure, fail to provide such information.

Instead of arbitrarily defining heuristic receptive fields, we aim to explicitly learn the receptive fields for classification tasks. Specifically, we propose to adaptively learn such regions by considering the receptive fields additional parameters, and jointly learning these parameters with the subsequent classifiers. The resulting benefit is two-fold: receptive fields tailored to classification tasks increase the overall accuracy of classification; in addition, with the help of such mid-level features, we are able to use a much lower-dimensional feature to achieve the state-of-the-art performance. We experiment with our algorithm on the benchmark CIFAR-10 dataset and other datasets, and report a significant improvement in both accuracy and efficiency.

The remainder of the paper is organized as follows. Section 2 provides a background review of the image classification pipeline, and Section 3 proposes the adaptive receptive field learning idea, as well as an efficient algorithm to carry out learning. Experiments are presented in Section 4, and we conclude the paper in Section 5.

2. The Classification Pipeline

In this section, we briefly review the image classification pipeline we adopted, which leads to the problem of learning the receptive fields for spatial pooling. Specifically, we will focus on two-layer classification approaches.

We illustrate the pipeline from raw images to the prediction of class labels in Figure 1. Specifically, starting with an input image $\mathbf{I}$, two stages are usually adopted to generate the global feature, as we formally define below.

(1) Coding. In the coding step, we extract local image patches, and encode each patch to K activation values based on a codebook of size K (learned via a separate dictionary learning step). These activations are typically binary (in the case of vector quantization) or continuous (in the case of e.g. sparse coding). It is generally believed that having an overcomplete ($K \gg$ the dimension of patches) codebook while keeping the activations sparse helps classification, especially when linear classifiers are used in the later steps.

Recently, Coates et al. [9] have shown that relatively simple dictionary learning and encoding approaches lead to surprisingly good performances. To learn a dictionary $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_K]$ of size K from randomly sampled patches $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\}$ each reshaped as a vector of pixel values, two simple yet effective approaches are advocated:

1. K-means, which minimizes the squared distance between each patch and its nearest code: $\min_{\mathbf{D}} \sum_{i=1}^{N} \min_j \|\mathbf{p}_i - \mathbf{d}_j\|_2^2$.

2. OMP-M, which learns a dictionary that minimizes the reconstruction error, with the constraint that each patch is modeled by a linear combination of at most M codes: $\min_{\mathbf{D},\boldsymbol{\alpha}_i} \sum_{i=1}^{N} \|\mathbf{p}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2$, where the length of each dictionary entry $\mathbf{d}_j$ is 1, and the cardinality of each reconstruction coefficient $\boldsymbol{\alpha}_i$ is at most M.

For encoding, Coates et al. also propose to substitute sparse coding by the following efficient approaches:

1. Triangle coding [6], which computes the activation of code k for a patch $\mathbf{p}$ as $f_k(\mathbf{x}) = \max\{0, \mu(\mathbf{z}) - z_k\}$, where z_k is the distance from $\mathbf{p}$ to the k-th code $\mathbf{d}_k$, and $\mu(\mathbf{z})$ is the mean of distances from $\mathbf{p}$ to all codes.

2. Soft thresholding, which computes the inner product between $\mathbf{p}$ and each code, with a fixed threshold parameter α: $f_k(\mathbf{x}) = \max\{0, \mathbf{d}_k^\top \mathbf{p} - \alpha\}$

We refer to [9] for a systematic discussion about different dictionary learning and encoding algorithms. In our experiment, we will adopt these standard approaches in order to isolate the contribution of spatial pooling from the choice of different coding methods. Since local patches are usually extracted densely in a grid-based fashion, we will organize the activations of image $\mathbf{I}$ as a set of matrices denoted by $\{\mathbf{A}^1(\mathbf{I})\mathbf{A}^2(\mathbf{I}), \cdots, \mathbf{A}^K(\mathbf{I})\}$, one for each code in the codebook, whose element $A_{ij}^k(\mathbf{I})$ contains the activation of code $\mathbf{d}_k$ for the local image patch at spatial location (i, j).

(2) Pooling. Since the coding result are highly overcomplete and highly redundant, the pooling layer aggregates the activations over certain spatial regions of the image to obtain an M dimensional vector $\mathbf{x}$ as the global representation

of the image. Each dimension of the pooled feature $\mathbf{x}_i$ is obtained by taking the activations of one code in a specific spatial region (shown as the red rectangular in Figure 1), and performing a predefined operator (usually average or max) on the set of activations.

We follow a similar approach to that in [3] to formally define pooled features. Specifically, given an operator op that maps a set of real values to a single real value (e.g. by taking their average), a pooled feature x_i can be defined based on the selection of a code indexed by c_i and a spatial region denoted by $\mathbf{R}_i$:

$$x_i = \text{op}(\mathbf{A}^{c_i}_{\mathbf{R}_i}) \tag{1}$$

Borrowing the definition from neuroscience, we call $\mathbf{R}_i$ the *receptive field* for the pooled feature, which could be seen as a binary mask over the image. $\mathbf{A}^{c_i}_{\mathbf{R}_i}$ is then the set of activations of code c_i in the receptive field $\mathbf{R}_i$.

This definition provides a general definition that embraces existing pooling algorithms. For example, commonly used operators involve computing the statistics of the activations under the p-norm:

$$x_i = \frac{1}{|\mathbf{R}_i|}\left(\sum\nolimits_{\alpha_i \in \mathbf{A}^{c_i}_{\mathbf{R}_i}} \alpha_i^p\right)^{\frac{1}{p}} \tag{2}$$

when $p = 1$ this corresponds to the average pooling, and when $p \to \infty$ this corresponds to the max pooling.

In our paper we focus on the definition of receptive fields for pooling. The simplest form of pooling takes the whole image as the receptive field, thus assuming a bag-of-words model where spatial information is ignored. The more commonly adopted spatial pooling approach [21, 33] pools features from multiple levels of regular grids, thus defining a pyramid of pooled features. Given a set of K codes and a set of N receptive fields, the pooled features are then defined by taking the Cartesian product of the codes and the receptive fields, yielding a KN-dimenisonal global feature.

Finally, a classifier, usually linear SVM or logistic regression, is trained using the global feature vector to predict the final label of the image as $y = f(\mathbf{x}; \boldsymbol{\theta})$.

3. Receptive Field Learning for Pooled Features

While significant efforts have been placed on the coding part of the classification pipeline, the pooling step has received relatively little attention. Existing research on pooling mainly focuses on the analysis of the pooling operator, such as in [4]. Specifically, spatial regions are almost always defined on regular grids [33]. In fact, regular grids may not guarantee to be optimal: for example, long horizontal bars may serve as better pooling regions for natural scenes, and such receptive fields may be dataset-dependent.

(a) (b) (c)

Figure 2. An example of overcomplete rectangular bins based on a 4×4 superpixel setting: (a) superpixels; (b) spatial pyramid bins; (c) overcomplete rectangular bins.

Inspired by the selectivity of complex cells in the visual cortex, we propose to learn the pooled features adaptively. Specifically, learning a set of M pooled features is equivalent to learning the parameters $\mathcal{C} = \{c_1, c_2, \cdots, c_M\}$ and $\mathcal{R} = \{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_M\}$ [1]. To this end, we note that the pooled features are directly fed into the final classifier, and propose to jointly learn the classifier parameters $\boldsymbol{\theta}$ together with the pooling parameters. Thus, given a set of training data $\mathcal{X} = \{(\mathbf{I}_n, \mathbf{y}_n)\}_{n=1}^N$, the joint learning leads to solving the following optimization problem:

$$\begin{aligned} \min_{\mathcal{C},\mathcal{R},\boldsymbol{\theta}} \quad & \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}(f(\mathbf{x}_n; \boldsymbol{\theta}), \mathbf{y}_n) + \lambda \,\text{Reg}(\boldsymbol{\theta}) \\ \text{where} \quad & x_{ni} = \text{op}(\mathbf{A}^{c_i}_{n,\mathbf{R}_i}) \end{aligned} \tag{3}$$

where we assume that the coding from $\mathbf{I}_n$ to $\{\mathbf{A}^{c_i}_n\}_{i=1}^K$ is done in an unsupervised fashion, as has been suggested by several papers such as [6]. We call this method receptive field learning, as the receptive fields are learned in such a way that information most relevant to the classification task will be extracted.

One practical issue is that solving the optimization problem (3) may be impractical, as there is an exponential number of receptive field candidates, leading to a combinatorial problem. Numerical solutions are also difficult, as the gradient with respect to the pooling parameters is not well-defined. Thus, instead of searching in the space of all possible receptive fields, we adopt the idea of over-completeness in the sparse coding community. Specifically, we start from a set of reasonably overcomplete set of potential receptive fields, and then find a sparse subset of such pooled features. The over-completeness enables us to maintain performance, while the sparsity allows us to still carry out classification efficiently during testing time.

3.1. Overcomplete Receptive Fields

The exponential number of possible receptive fields arises when we consider the inclusion and exclusion of single pixels individually. In practice this is often unnecessary, as we expect the active pixels in a receptive field to be

[1] For simplicity, we will use the max operator, but note that any operator could also be incorporated in our framework.

spatially contiguous. In this paper, we use receptive fields consisting of rectangular regions[2]: this provides us a reasonable level of over-completeness, as there are $O(n^4)$ different rectangular receptive fields for an image containing $n \times n$ pixels. In addition, since the motivation of spatial pooling is to provide tolerance to small spatial displacements, we build the rectangular regions upon superpixels, which are defined as dense regular grids on the image. Figure 2 shows an example of such rectangular receptive fields compared with regions defined by the spatial pyramid on a 4×4 grid.

Given the set of P overcomplete regions, which we denote by $\mathcal{R} = \{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_P\}$, and the dictionary $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_K\}$ of size K, we can define a set of PK potential pooled features based the Cartesian product $\mathcal{R} \times \mathcal{D}$. Specifically, the i-th receptive field and the j-th code jointly defines the $(K \times i + j)$-th pooled feature as $x_{K \times i + j} = \text{op}(\mathbf{A}^j_{\mathbf{R}_i})$. Note that when the coding and pooling are both carried out in an overcomplete fashion, the resulting pooled feature is usually very high-dimensional.

3.2. Structured Sparsity for Receptive Field Learning

While it is possible to train a linear classifier using the high-dimensional pooled feature $\mathbf{x}$ above, in practice it is usually beneficial to build a classifier using relatively low-dimensional features. In addition, for multiple-label classification, we want the classifiers of different labels to share features. This brings two potential advantages: feature computation could be minimized, and sharing features among different classifiers is known to provide robustness to the learned classifiers. To this end, we adopt the idea of structured sparsity [26, 29], and train a multiple-class linear classifier $\mathbf{y} = f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ via the following optimization problem:

$$\min_{\mathbf{W},\mathbf{b}} \quad \frac{1}{N}\sum_{n=1}^{N} l(\mathbf{W}^\top \mathbf{x}_n + \mathbf{b}, \mathbf{y}_n) + \frac{\lambda_1}{1}\|\mathbf{W}\|^2_{\text{Fro}} + \lambda_2 \|\mathbf{W}\|_{1,\infty} \tag{4}$$

where $\mathbf{y}_i$ is the L-dimensional label vector coded in a $1-of-L$ fashion, with values taken from $\{-1, +1\}$ given L classes. $\mathbf{x}_i$ is an M-dimensional feature vector defined by overcomplete pooling in the previous subsection, and $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_L]$ is a $M \times L$ weight matrix containing the weight vector for the L classifiers.

Two regularization terms are adopted in the optimization. The squared Frobenius norm $\|\mathbf{W}\|^2_{\text{Fro}}$ aims to minimize the structured loss in the classical SVM fashion, and the second regularizer is the $L_{1,\infty}$ norm of the matrix $\mathbf{W}$:

$$\|\mathbf{W}\|_{1,\infty} = \sum_{i=1}^{M} \|\mathbf{W}_{i,\cdot}\|_\infty = \sum_{i=1}^{M} \max_{j \in \{1,\cdots,L\}} |W_{ij}| \tag{5}$$

where $\mathbf{W}_{i,\cdot}$ denotes the i-th row of the matrix W. This regularizer introduces structured sparsity by encouraging the weight matrix $\mathbf{W}$ to be row-wise sparse, so that the classifiers for different classes tend to agree on whether to use a specific feature, and when combined together, only jointly use a subset of the overcomplete pooled features. The addition of the $L_{1,\infty}$ norm also provides a elastic-net like regularization, which is known to perform well when the dimension of data is much higher than the number of data points [37].

For optimization considerations, we use the multi-class extension of the binomial negative log likelihood (BNLL) loss function [25]:

$$l(\mathbf{W}^\top \mathbf{x} + \mathbf{b}, \mathbf{y}) = \sum_{i=1}^{L} \ln(1 + e^{-\mathbf{y}_i(\mathbf{W}^\top_{\cdot,i}\mathbf{x} + b_i)}) \tag{6}$$

The choice of the BNLL loss function over the hinge loss is mainly for computational simplicity, as the gradient is easier to compute for any input. In practice, the performance does not change much if we use the hinge loss instead.

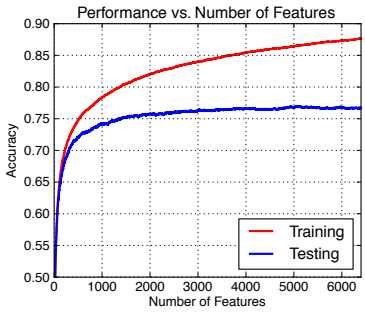

Figure 3. Performance vs. number of selected features, with the experiment setting in Table 1 of Section 4.

3.3. Fast Approximate Learning by Feature Selection

Jointly optimizing (4) is still a computationally challenging task despite its convexity, due to the over-completeness in both coding and pooling. While it is possible to carry out the computation on smaller-scale problems like Caltech-101, we adopt a greedy approach to train the model for larger-scale problems. Inspired by the matching pursuit algorithm in dictionary training and the grafting algorithm [25] in machine learning, we start with an empty set of selected features, incrementally add features to the set, and retrain the model when new features are added.

Mathematically, we maintain a set $\mathcal{S}$ recording the set of currently selected features. At each iteration, for each

[2] As a side note, we also experimented with receptive fields that are sampled from an Ising model on the fly during training, but rectangular regions worked empirically better, possibly because the additional flexibility of Ising models leads to over-fitting the training data.

feature index j that has not been not selected, we compute the score of the feature as the 2-norm of the gradient of the objective function (4), denoted by $\mathcal{L}(\mathbf{W}, \mathbf{b})$, with respect to the corresponding weight vectors:

$$\text{score}(j) = \left\| \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{W}_{j,\cdot}} \right\|_{\text{Fro}}^2 \tag{7}$$

We then select the feature with the largest score, and add it to the selected set $\mathcal{S}$. The model is retrained using the previously learned optimum solution as the starting point. From a boosting perspective, this can be considered as incrementally learning weak classifiers, but our method differs from boosting in the sense that the weights for already selected features are also updated when new features are selected.

As the speed of retraining drops when more features are added, we adopt an approximate retraining strategy: for each iteration t, we select an active subset $\mathcal{S}_A$ of $\mathcal{S}$ based on the score above. We then retrain the model with respect to the active set and the bias term only:

$$\mathbf{W}_{\mathcal{S}_A,\cdot}^{(t+1)}, \mathbf{b} = \arg\min_{\mathbf{W}_{\mathcal{S}_A,\cdot}, \mathbf{b}} \mathcal{L}(\mathbf{W}, \mathbf{b}) \tag{8}$$

with the constraint that $\mathbf{W}_{\bar{\mathcal{S}}_A,\cdot}$ keep unchanged. The intuition is that with an already trained classifier from the previous iteration, adding one dimension will only introduce small changes to the existing weights.

In practice, we found the performance of this approximate algorithm with the active set size less than 100 to be very close to the full retraining algorithm with a significant increase in computation speed. Figure 3 shows typical curves of the training and testing accuracy with respect to the number of iterations. The performance usually stabilizes with a significantly smaller number of features, showing the effectiveness of introducing structured sparsity into classifier learning.

4. Experiments

We will mainly report the performance of our algorithm on the CIFAR-10 dataset[3], which contains 50,000 32×32 images from 10 categories as training data, and 10,000 images as testing data.

We fix the dictionary learning algorithms to k-means clustering and the coding algorithms to triangular coding as proposed in [6] for CFAR-10. Such a coding strategy has been shown to be particularly effective in spite of its simplicity. We also tested alternative dictionary learning and coding algorithms, which led to similar conclusions. As our main focus is on learning receptive fields for pooled features, the results of different coding algorithms are omitted, and we refer to [9] for a detailed discussion about dictionary learning and coding algorithms.

[3] http://www.cs.toronto.edu/ kriz/cifar.html

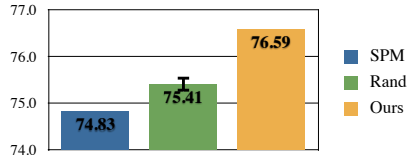

Figure 4. Performance comparison among spatial pyramid pooling, random feature selection and our method, all using the same number of features for the final classification. It can be observed that a few selected features could already achieve a comparatively high performance.

For classification, when we use pre-defined receptive fields such as spatial pyramids, the SVM regularization term is chosen via 5-fold cross validation on the training data. When we perform feature selection, we fix $\lambda_1 = 0.01$ (which is the best value when performing 5-fold cross validation for max pooling on a 2×2 regular grid) and drop λ_2, since the incremental feature selection already serves as a greedy approximation of the sparse constraint. Although the parameters are not tuned specifically for each configuration, we found it to perform well empirically under various scenarios.

4.1. Spatial Pyramid Revisited

It is interesting to empirically evaluate the performance of spatial pyramid regions against other choices of receptive fields. To this end, we trained a dictionary of size 200 (for speed considerations), and tested the performance of 3-layer spatial pyramid pooling against two algorithms based on overcomplete receptive fields: (1) random selection from the overcomplete pooled features, and (2) our method, both selecting the same number of features that spatial pyramid pooling uses. Results are shown in Figure 4. Our method outperforms SPM, but a more interesting finding is that the predefined spatial pyramid regions perform consistently worse than random selection, indicating that arbitrarily defined pooled features may not capture the statistics of real-world data well. With explicit learning of the pooling parameters, we achieved the highest performance among the three algorithms, showing the effectiveness and necessity of learning adaptive receptive fields.

4.2. The Effect of Spatial Over-completeness

One may ask if the performance increase could be obtained without overcompletenes by simply using a denser grid. To answer this question, we examined the performance of our algorithm against the 2×2 pooling grid (which is used in [9] to obtain very high performance) and a denser 4×4 grid, with both average and max poolings. We also compared our method against random feature selection from the same pooling candidates. Table 1 summarizes the

Pooling Area	Method	Features	Accuracy
2×2	Ave	800	70.24
4×4	Ave	3,200	72.24
2×2	Max	800	66.31
4×4	Max	3,200	73.03
3-layer SPM	Max	4,200	74.83
OC + feat select	Max	800	73.42
		3,200	76.28
		4,200	76.59
		6,400	**76.72**
OC, all features	Max	20,000	76.44
OC + rand select	Max	800	69.48
OC + rand select	Max	3,200	74.42
OC + rand select	Max	4,200	75.41

Table 1. Comparison of different pre-defined pooling strategies and our method (overcomplete (OC) + feature selection). Random selection from the same overcomplete pooled features is also listed, showing the necessity of better receptive field learning.

testing accuracy under various experimental settings, using a codebook size of 200.

Results from Table 1 demonstrates that denser pooling does help performance. The 4×4 grid increases the performance by about 3 percent compared to 2×2 pooling. However, with overcomplete receptive fields we can almost always increase performance further. We achieved an 76.72% accuracy with only 200 codes, already close with state-of-the-art algorithms using much larger codebook sizes (Table 2). It is also worth pointing out that even random feature selection gives us comparable or better performance when compared to pre-defined pooling grids under the same number of feature dimension (e.g. compare the performance between 4×4 max pooling and randomly selecting $3,200$ features from an overcomplete set of pooled features).

Further, the importance of feature selection lies in two aspects: first, simply using all the features is not practical during testing time, as the dimension can easily go to hundreds of thousands when we increase the codebook size. Feature selection is able to get very close performance compared to using all the features, but with a significantly lower dimensionality, which is essential in many practical scenarios. Usually, feature selection enables us to achieve a high performance with only a few features (Figure 3). Adding remaining features will only contribute negligibly to the overall performance. Second, performing feature selection has the potential benefit of removing redundancy, thus increasing the generalization ability of the learned classifiers [25, 30]. In our experiment in Table 1, the best performance is achieved with a few thousands features. Similarly, we found that with larger codebook sizes, using all the overcomplete pooled features actually decreases performance, arguably due to the decrease of the generalization ability.

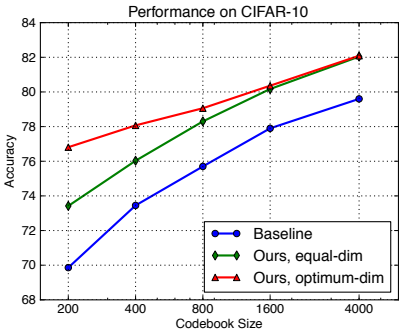

Figure 5. Testing accuracy on CIFAR-10 with and without overcomplete pooling. In the figure, "equal-dim" selects the same number of features as the baseline (Coates et al.[6]), and "optimum-dim" selects the optimum number of features determined by cross-validation. (X-axis in log scale)

4.3. Larger Codebook vs. Better Spatial Pooling

Under the two-stage pipeline adopted in this paper, there are effectively two possible directions to increase the performance: to increase the codebook size and to increase the pooling over-completeness. We argue that these two directions are complementary: the performance gain from our effort on pooling could not simply be replaced by increasing the codebook size, at least not easily. More importantly, as the codebook size grows larger, it becomes more difficult to obtain further performance gain, while it is still relatively easy to obtain gains from better pooling.

To empirically justify this argument, we trained multiple codebooks of different sizes, and compared the resulting accuracies with and without overcomplete pooling in Figure 5. As can be observed, it becomes harder to obtain further performance gain by increasing the codebook size when we already have a large codebook, while using a better pooling strategy always brings additional accuracy gains. In fact, with our method, we are able to use a codebook of half the size (and half the number of pooled features) while maintaining performance (compare the green and blue curves). It is particularly interesting that, by selecting more features from the overcomplete spatial regions, we are able to achieve state-of-the-art performance with a much smaller number of codes (the red curve), which has the potential in time-sensitive or memory-bounded scenarios.

4.4. Best Performance

Our best performance on the CIFAR-10 dataset was achieved by training a codebook size of 6,000, performing max pooling on overcomplete rectangular bins based on a 4×4 grid, and selecting features up to 24,000 dimensions. We also note that the accuracy has not saturated at this number of features, but we would like to test the performance when the number of mid-level features is limited to a reasonable scale. With these settings, we achieved an accuracy

Method	Pooled Features	Accuracy
ours, d=1600	6,400	80.17
ours, d=4000	16,000	82.04
ours, d=6000	24,000	**83.11**
Coates et al. [6], d=1600	6,400	77.9
Coates et al. [6], d=4000	16,000	79.6
Coates et al. [9], d=6000	48,000	81.5
Conv. DBN [18]	N/A	78.9
Improved LCC [35]	N/A	74.5
8-layer Deep NN [5]	N/A	80.49
3-layer Deep NN [8]	N/A	82.0

Table 2. Performance on the CIFAR-10 dataset. The first and second blocks compare performance between our method and Coates et al. [6, 9] under similar codebook sizes, where the only difference is the spatial pooling strategy. The third block reports the performance of several state-of-the-art methods in the literature.

of 83.11% on the testing data. To the best of our knowledge, this is the best published result on CIFAR-10 without increasing the training set size by morphing the images.

Table 2 lists the performance of several state-of-the-art methods. It is also worth pointing out that, to achieve the same performance, our algorithm usually uses a much lower number of features compared with other well-performing algorithms.

4.5. Results on MNIST

We can view the set of learned receptive fields for pooling as a saliency map for classification [14]. To visually show the saliency map and verify its empirical correctness, we applied our method to handwritten digit recognition on the MNIST dataset, on which convolutional deep learning models are particularly effective. To this end, we adopted a similar pipeline as we did for CIFAR-10: dense 6x6 local patches with ZCA whitening are used; a dictionary of size 800 is trained with OMP-1, and thresholding coding with $\alpha = 0.25$ (untuned) is adopted. The features are then max-pooled on overcomplete rectangular areas based on a 6×6 regular grid. Note that we used a different coding method from the CIFAR-10 experiment to show that the overcomplete spatial pooling method is agnostic of the choice of low-level coding algorithms. Any parameter involved in the pipeline such as SVM regularization weights is tuned on a random 50k/10k split of the training data.

Figure 6 shows the 1-vs-1 saliency maps between digits. It can be seen that by learning receptive fields, the classifier focuses on regions where the digits have maximal dissimilarity, e.g., the bottom part for 8 and 9, and the top part for 3 and 5, which matches our intuition about their appearances. For 10-digit classification, we achieved an error rate of 0.64%, on par with several state-of-the-art algorithms (Figure 6 left). A gap still exists between our method and the best deep-learning algorithm, and combining receptive learning with deeper structures is future work.

Method	err%
Baseline [9][a]	1.02
Our Method	**0.64**
Lauer et al. [20]	0.83
Labusch et al. [19]	0.59
Ranzato et al. [27]	0.62
Jarrett et al. [15]	0.53

[a]Our implementation.

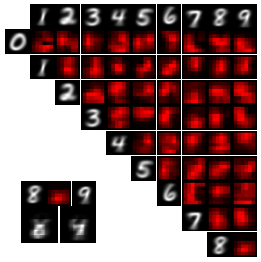

Figure 6. Left: Performance comparison (error rate in percentage) on MNIST. Top box: comparison between algorithms using similar pipelines. Bottom box: performance of other related algorithms in the literature. Right: 1-vs-1 saliency maps learned on MNIST. The left-bottom corner plots the mean of digit 8 and 9 multiplied by the corresponding saliency map, showing that the classifier focuses on the bottom part which intuitively also distinguishes the two digits best.

Method	Codebook	Pooling	Performance
ScSPM [33]	1024 (SC)	SPM	73.2±0.54
LCC+SPM [31]	1024	SPM	73.44
Our Method	1024 (SC)	OC	**75.3±0.70**
Boureau et al. [3]	64K	SPM	77.1±0.7

SPM [21]	64.6±0.7
NBNN [1]	72.8±0.39 (15 training)
Jarret et al. [15]	65.6±1.0
RLDA [16]	73.7±0.8
Adaptive Deconv. Net [36]	71.0±1.0
Feng et al. [12]	82.6

Table 3. Performance comparison (accuracy in percentage) on Caltech-101. Top: comparison between algorithms using similar pipelines. Bottom: performance of other related algorithms in the literature.

4.6. Results on Caltech-101

Lastly, we report the performance of our algorithm compared with SPM on the Caltech-101 dataset in Table 3. State-of-the-art performance following similar pipelines are also included in the table. Specifically, we used the same two-step pipeline as proposed by Yang et al. [33]: SIFT features are extracted from 16×16 patches with a stride of 8, and are coded using sparse coding with a codebook of size 1024. For SPM, the coded features are pooled over a pyramid of $1 \times 1, 2 \times 2, 4 \times 4$ regular grids; for a fair comparison we also use the 4×4 regular grid as our base regions, and select the same number of features as SPM uses.

As can be observed in the table, our pooling algorithm outperforms spatial pooling, although a gap still exists between our result and state-of-the-art methods, which uses more complex coding schemes than that we used. The results suggest that coding is a more dominant factor for the performance of Caltech-101. Existing research, especially the Naive Bayes nearest neighbor method [1], has also shown a consistent increase of accuracy with higher-

dimensional coding output [3, 34]. However, we still obtain a consistent gain by adopting more flexible receptive fields for pooling, which justifies the effectiveness of the proposed algorithm. Note that the best performance reported by Feng et al. [12] was obtained by jointly learning the pooling operator (p in p-norm pooling) and a per-code spatial saliency map in addition to a larger dictionary, which also follows the idea of learning better spatial information beyond SPM.

5. Conclusion

In this paper, we examined the effect of receptive field designs on the classification accuracy in the commonly adopted pipeline of image classification. While existing algorithms use manually defined spatial regions for pooling, learning more adaptive receptive fields increases performance even with a significantly smaller codebook size at the coding layer. We adopted the idea of over-completeness and structured sparsity, and proposed an efficient algorithm to perform feature selection from a set of pooling candidates. With this method, we achieved the best published performance on the CIFAR-10 dataset, using a much lower dimensional feature space than previous methods. Possible future work involves more flexible definition of pooling receptive fields, and unsupervised learning of such pooled features.

References

[1] O Boiman, E Shechtman, and M Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008. 7

[2] Y Boureau, F Bach, Y LeCun, and J Ponce. Learning mid-level features for recognition. In *CVPR*, 2010. 1

[3] Y Boureau, N Le Roux, F Bach, J Ponce, and Y LeCun. Ask the locals: multi-way local pooling for image recognition. In *ICCV*, 2011. 3, 7, 8

[4] Y Boureau and J Ponce. A theoretical analysis of feature pooling in visual recognition. In *ICML*, 2010. 1, 3

[5] DC Cireşan, U Meier, J Masci, LM Gambardella, and J Schmidhuber. High-performance neural networks for visual object classification. *ArXiv e-prints*, 2011. 7

[6] A Coates, H Lee, and AY Ng. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2010. 1, 2, 3, 5, 6, 7

[7] A. Coates and A.Y. Ng. Selecting receptive fields in deep networks. In *NIPS*, 2011. 1

[8] A Coates and AY Ng. Selecting receptive fields in deep networks. In *NIPS*, 2011. 7

[9] A Coates and AY Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, 2011. 1, 2, 5, 7

[10] N Dalal. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1

[11] M Everingham, L Van Gool, CKI Williams, J Winn, and A Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010. 1

[12] J Feng, B Ni, Q Tian, and S Yan. Geometric Lp-norm Feature Pooling for Image Classification. In *CVPR*, 2011. 7, 8

[13] DH Hubel and TN Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160:106–154, 1962. 1

[14] L Itti and C Koch. Computational modeling of visual attention. *Nature reviews neuroscience*, 2001. 7

[15] K Jarrett, K Kavukcuoglu, MA Ranzato, and Y LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009. 7

[16] S Karayev, M Fritz, S Fidler, and T Darrell. A probabilistic model for recursive factorized image features. In *CVPR*, 2011. 7

[17] JJ Koenderink and AJ van Doorn. The structure of locally orderless images. *IJCV*, 31(2/3):159–168, 1999. 1

[18] A Krizhevsky. Convolutional deep belief networks on CIFAR-10. *Technical Report*, 2010. 7

[19] K Labusch, E Barth, and T Martinetz. Simple method for high-performance digit recognition based on sparse coding. *IEEE TNN*, 19(11):1985–1989, 2008. 7

[20] F Lauer, CY Suen, and G Bloch. A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*, 40(6):1816–1824, 2007. 7

[21] S Lazebnik, C Schmid, and J Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 1, 3, 7

[22] D Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 1

[23] J Mairal, F Bach, J Ponce, and G Sapiro. Online learning for matrix factorization and sparse coding. *JMLR*, 11:19–60, 2010. 1

[24] B Olshausen and DJ Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997. 1

[25] S Perkins, K Lacker, and J Theiler. Grafting: fast, incremental feature selection by gradient descent in function space. *JMLR*, 3:1333–1356, 2003. 4, 6

[26] A Quattoni, M Collins, and T Darrell. Transfer learning for image classification with sparse prototype representations. In *CVPR*, 2008. 4

[27] MA Ranzato, FJ Huang, Y Boureau, and Y LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007. 7

[28] R Rigamonti, MA Brown, and V Lepetit. Are sparse representations really relevant for image classification? In *CVPR*, 2011. 1

[29] M Schmidt, K Murphy, G Fung, and R Rosales. Structure learning in random fields for heart motion abnormality detection. In *CVPR*, 2008. 4

[30] R Tibshirani. Regression shrinkage and selection via the lasso. *JRSS Series B*, pages 267–288, 1996. 6

[31] J Wang, J Yang, K Yu, F Lv, T Huang, and Y Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 1, 7

[32] S Winder and M Brown. Learning local image descriptors. In *CVPR*, 2007. 1

[33] J Yang, K Yu, and Y Gong. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009. 1, 3, 7

[34] J Yang, K Yu, and T Huang. Efficient highly over-complete sparse coding using a mixture model. In *ECCV*, 2010. 1, 8

[35] K Yu and T Zhang. Improved local coordinate coding using local tangents. In *ICML*, 2010. 7

[36] MD Zeiler, GW Taylor, and R Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011. 7

[37] H Zou and T Hastie. Regularization and variable selection via the elastic net. *JRSS*, 67(2):301–320, 2005. 4