Beyond Spatial Pyramids: Receptive Field Learning for Pooled Image Features

Yangqing Jia
UC Berkeley EECS
jiayq@berkeley.edu

Chang Huang
NEC Labs America
chuang@sv.nec-labs.com

Abstract

We examine 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, we argue that learning more adaptive receptive field increases performance even with a significantly smaller codebook size at the coding layer. To this end, we adopt the idea of overcompleteness by starting with a large number of receptive field candidates, and train a classifier that only uses a sparse subset of them. An efficient algorithm based on grafting is proposed to perform feature selection from a set of pooling candidates. With this method, we achieve the best published performance on the CIFAR-10 dataset, using a much lower dimensional feature space than previous methods did.

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 for the task. Many methods start from local image patches using either normalized raw pixel values or hand-crafted descriptors such as SIFT [16] or HOG [8], 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 [15, 21]. Such global representations are then fed into standard classifiers such as support vector machines to predict category labels for the image. Methods following such a pipeline achieved state-of-the-art performance on several challenging classification tasks, such as Caltech-101 and Pascal VOC [9].

During the last decade, much spotlight has been directed onto the coding step. On the one hand, dictionary learning algorithms have been discussed to find a set of basis that reconstructs the local image patches or descriptors well. On the other hand, several encoding methods are proposed to map the original data to a high-dimensional space that emphasizes certain properties, such as sparsity [17] or locality [20]. Recent papers [6, 19, 7] have explored the relationship between dictionary and coding, and have proposed simple yet effective approaches that achieves competitive results. A neuroscience justification of coding comes from simplex neurons in the human visual cortex V1, which has been believed to produce sparse and overcomplete activations [17].

Similarly, the idea of spatial pooling dates back to Hubel's seminal paper about complex cells in V1 [10], with the merit of finding mid-level image features that are invariant to small spatial shifting to some extent. The spatial invariance property also reflects the concept of locally orderless images [13]. Most recent researches on spatial pooling aim to find a good pooling operator, which could be seen as a function that produces informative statistics based on a set of 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]. On the other hand, the spatial regions for pooling, which we also call the recep-

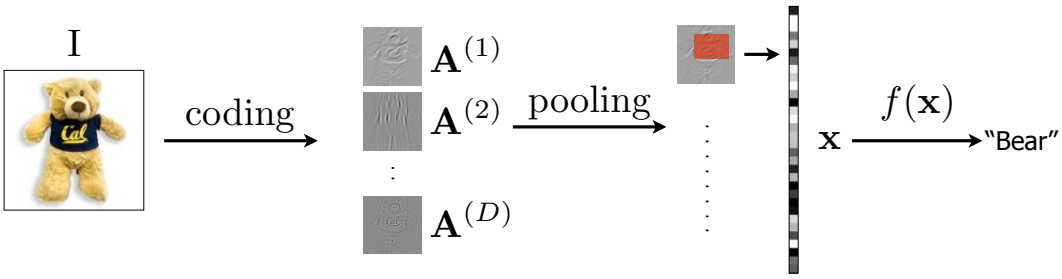


Figure 1: The image classification pipeline. See text for details.

tive fields for the pooled features, are usually defined via a spatial pyramid [15, 21], and relatively little effort has been put into better receptive field designs since the introduction of spatial pyramids.

In this paper, we ask the question "are spatial pyramids sufficient for image classification", and instead of arbitrarily defining receptive fields using such heuristics, we aim to learn them for pooled features. This could be seen as an orthogonal effort to the existing work of learning the pooling operators. While a pyramid of regions succeed in providing us information about the spatial layout of image features, one can reasonably question 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 may be captured by thin horizontal pooling regions over the image. Spatial grids, even with a pyramid structure, fail to provide such information.

Specifically, we propose to adaptively learn such regions by taking the receptive field of each pooled feature into consideration, and jointly learn the pooled features and the classifiers. The benefit of such an approach is two-fold: receptive fields tailored to classification performance increases 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 than conventional approaches. We experiment our algorithm on the benchmark CIFAR-10 dataset, and report a significant improvement in both accuracy and efficiency.

The paper is organized as follows. Section 2 introduces the general idea of adaptive receptive fields for spatial pooling. Section 3 proposes an efficient algorithm to learn such receptive fields. Experiments are presented in Section 4.

2 Adaptive Receptive Field Learning

In this section we briefly review the pipeline of image classification, and formally formulate the problem of adaptive receptive field learning for spatial pooling.

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

- Coding: dense local image patches are extracted, and each patch is transformed to D activation values based on a codebook of size D, 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). Also, it is generally believed that having an overcomplete ($D \gg$ the dimension of patches) codebook while keeping the activations sparse helps classification, especially when linear classifiers are used in the later steps. We will organize the activations as matrices denoted by $\mathbf{A}^1(\mathbf{I}), \mathbf{A}^2(\mathbf{I}), \cdots, \mathbf{A}^D(\mathbf{I})$, one for each code in the codebook, where $\mathbf{A}^k_{ij}(\mathbf{I})$ contains the activation of code k for the local image patch at spatial location (i, j).
- Pooling: the activations are pooled over certain spatial regions of the image to obtain a 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.

- Classification: a classifier (usually linear SVM or logistic regression) is applied to the feature vector to predict the final label of the image as $y = f(\mathbf{x}; \boldsymbol{\theta})$.

We follow a similar approach to [3] to formally define a pooled feature. Specifically, given an operator op (such as average or max), a pooled feature x_i is defined with a selected code c_i and a receptive field $\mathbf{R}_i$:

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

where $\mathbf{R}_i$ could be seen as a binary mask over the image, and $\mathbf{A}^{c_i}_{\mathbf{R}_i}$ is 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: the Bag-of-words model uses the average operator, all the D codes, and the receptive field covering the whole image for each code, creating a D-dimensional pooled feature.

To learn a set of M pooled features is equivalent to learn the parameters $\mathcal{C} = \{c_1, c_2, \cdots, c_M\}$ and $\mathcal{R} = \{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_M\}$, and sometimes the operator op as well. To this end, we note that pooled features are directly fed into the final classifier, and propose to jointly learn the classifier parameters $\boldsymbol{\theta}$ together with the pooling parameters. 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:

$$\begin{aligned} \min_{\mathcal{C},\mathcal{R},\boldsymbol{\theta}} \quad & \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}(f(\mathbf{x}_n;\boldsymbol{\theta}), \mathbf{y}_i) + \lambda \,\mathrm{Reg}(\boldsymbol{\theta}) \\ \text{where} \quad & x_{ni} = \mathrm{op}(\mathbf{A}^{c_i}_{\mathbf{R}_i}(\mathbf{I}_n)) \end{aligned} \tag{2}$$

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

3 Approximate Feature Learning with Structured Sparsity

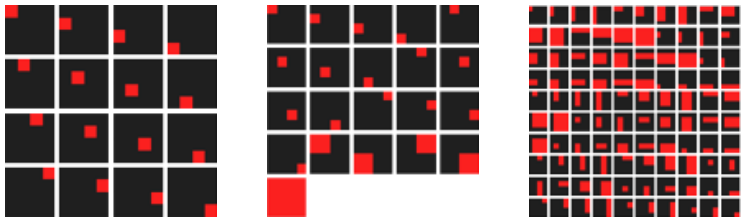
Solving optimization problem (2) may be impractical, as there is an exponential number of receptive field candidates, leading to a combinatorial problem. Thus, instead of searching in the space of all possible receptive fields, we adopt the idea of over-completeness in the sparse coding community. Specifically, 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.

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 spatially contiguous. In this paper, we use receptive fields that have rectangular shapes: this provides us a reasonable level of over-completeness, as for a $n \times n$ image, there is a polynomial number of $O(n^4)$ different rectangular receptive fields. In addition, since the motivation of spatial pooling is to provide tolerance to small spatial displacements, we build the rectangles 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.

Given the set of K overcomplete regions, which we denote by $\mathcal{R} = \{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_K\}$, together with the codebook $\mathcal{C}$ of size D, we can define the set of $M = DK$ pooled features based the cartesian product $\mathcal{R} \times \mathcal{C}$. Specifically, the i-th receptive field and the j-th code jointly defines the $(D*i+j)$-th pooled feature as $x_{D*i+j} = \mathrm{op}(\mathbf{A}^{c_j}_{\mathbf{R}_i})$. Note that when the coding and pooling are carried out both in an overcomplete fashion, the resulting pooled feature is usually very high-dimensional[1].

[1] In practice, although it seems difficult to store all the overcomplete pooled features, notice that for a rectangular composed of multiple superpixels, its corresponding feature can be computed trivially using the features of the corresponding superpixels, and only the latter needs to be stored. Thus overcomplete receptive fields do not create a significant storage overhead.



(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.

3.2 Approximate Receptive Field Learning via Structured Sparsity

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 classifications we want the classifiers of different labels to share features, so feature computation could be minimized. To this end, we adopt the idea of structured sparsity [12], and train a multiple-class linear classifier $\mathbf{y} = f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ via the following optimization problem:

$$\min_{\mathbf{W},\mathbf{b}} \quad \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\mathbf{W}^\top \mathbf{x}_i + \mathbf{b}, \mathbf{y}_i) + \frac{\lambda_1}{1}\|\mathbf{W}\|_{\text{Fro}}^2 + \lambda_2 \|\mathbf{W}\|_{1,\infty} \tag{3}$$

where $\mathbf{y}_i$ is the L-dimensional label vector with values taken from $\{-1, +1\}$ given L classes, $\mathbf{x}_i$ is an M-dimensional feature vector, 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 $\|\mathbf{W}\|_{\text{Fro}}^2$ aims to minimize the structured loss in the SVM context, and the second regularizer, which is the $L_{1,\infty}$ norm of the matrix $\mathbf{W}$:

$$\|\mathbf{W}\|_{1,\infty} = \sum_{i=1}^{M} \|\mathbf{W}_{i,\cdot}\|_\infty = \sum_{i=1}^{M} \max_{j \in \{1,\cdots,L\}} |W_{ij}| \tag{4}$$

introduces the structured sparsity. Specifically, it encourages 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 combined together only use a subset of the overcomplete pooled features. 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 [24].

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

$$\mathcal{L}(\mathbf{W}^\top \mathbf{x} + \mathbf{b}, \mathbf{y}) = \sum_{i=1}^{L} \ln(1 + e^{-\mathbf{y}_i(\mathbf{W}_{\cdot,i}^\top \mathbf{x} + b_i)}) \tag{5}$$

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

3.3 Incremental Feature Selection and Retraining

Jointly optimizing (3) is still a computationally challenging task albeit its convexity, due to the over-completeness of both coding and pooling. Thus, we adopt a greedy approach to train the model: starting with an empty set of selected features, incrementally add features to the set, and retrain the model when new features are added. The grafting algorithm [18] provides a recipe for such an approach. At each iteration, the feature with the largest gradient w.r.t. the current classifier is selected, and 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 differs in the sense that old weak classifiers are updated when new weak classifiers arrive.

As the speed of retraining drops when there are more features, we adopt an approximate retraining strategy: in each iteration, we select an active subset of all the selected features based on the gradient of the objective function. We then retrain the model with respect to the active set and the bias term only. In practice, we found the performance of this approximate grafting algorithm to be very close to the full grafting algorithm with a significant increase in computation speed.

4 Experiments

We report the performance of our algorithm on the CIFAR-10 dataset[2], which contains 50,000 32×32 images from 10 categories as training data, and 10,000 images as testing data.

We will fix the dictionary learning algorithms to k-means clustering, and the coding algorithms to triangular coding as proposed in [6] throughout the experiments. Such a coding strategy has been shown to be particularly effective albeit its simplicity. As a side note, 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 is omitted, and we refer to [7] for a detailed discussion about dictionary learning and coding algorithms.

For classification, if we use pre-defined receptive fields such as spatial pyramids, the SVM regularization term is chosen from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ via 5-fold cross validation on the training data; if we perform feature selection, we fix the regularization term to be $\lambda_1 = 0.01$ and $\lambda_2 = 0.001$, which is the best value when performing 5-fold cross validation for max pooling on a 2×2 regular grid. Note that although the parameter is not tuned specifically for our method, we found it to perform well empirically under various scenarios.

4.1 Can Spatial Pyramids be Improved?

It is interesting to empirically evaluate the performance of spatial pyramid bins. To this end, we trained a dictionary of size 200, and tested the performance of 3-layer spatial pyramid pooling against two algorithms based on overcomplete receptive fields: random selection from the overcomplete pooled features, and performing grafting, both selecting the same number of features that spatial pyramid pooling does. Figure 3 shows the performance comparison. To one's surprise, the predefined spatial pyramid regions perform even slightly worse than random selection, indicating that arbitrarily defined pooled features may not capture the statistics of real-world data well. With grafting, we achieve the highest performance among the three algorithms, showing the effectiveness of learning adaptive receptive fields.

4.2 The Importance of Adaptive Receptive Field Learning

One may ask if the performance increase could be obtained by simply using a denser grid without going overcomplete. To answer this question, we examined the performance of our algorithm against the 2×2 pooling grid, which is used in e.g. [7] to get the previously best performance, and a denser 4×4 grid, with both average and max poolings. We also compare our method against random feature selection from the same overcomplete pooled features. Table 1 lists the testing accuracy under various experimental settings, using a codebook size of 200 for speed consideration, as denser grids lead to higher dimensions.

The first thing to notice in the table is 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: it is worth pointing out that with overcomplete pooled features, even random feature selection gives us comparable or better performance 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: on the one hand, 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

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

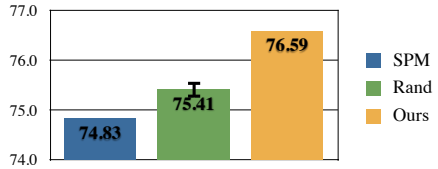


Figure 3: Performance comparison among spatial pyramid pooling, random feature selection and grafting based on overcomplete pooled features, both using the same number of features for final classification.

Figure 4: Performance vs. number of selected features, corresponding to the experiment in Table 1.

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 + grafting	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) + grafting). Random selection from the same overcomplete pooled features is also listed, showing the necessity to do smart feature selection.

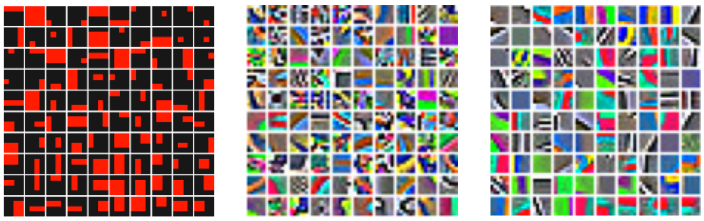
compared to using all the features, but with a significantly lower dimensionality. For example, we achieved an 76.72% accuracy with only 200 codes, already close with those obtained with state-of-the-art algorithms with much larger codebook sizes (Table 2). On the other hand, performing feature selection has the potential benefit of removing redundant features, thus reducing the possibility of overfitting to the training data, especially when the original feature dimension is relatively high compared with the number of training data. In our experiment in Table 1, the best performance is achieved with grafting a few thousands of features, instead of using all the features. Similarly, we found that with larger codebook sizes, using all the overcomplete pooled features actually decreases performance, partially due to an overly large number of irrelevant features. Figure 4 shows typical curves of the training and testing performance during the grafting procedure. Usually, feature selection enables us to achieve a high performance with a few features, and adding remaining features will only grind the final small fraction of the overall performance. This indicates one advantage of feature selection - redundant information contained in the overcomplete representation could be efficiently removed.

Method	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
Conv. DBN [14]	9,024	78.9
Improved LCC [23]	4,096 [a]	74.5
Deep NN [5]	100 [b]	80.49
Coates et al., [7], d=6000	48,000	81.5

[a] Different from other methods, [23] trains LCC on the whole image without pooling.

[b] Although the final layer is small, the authors used an 8-layer deep neural network with interleaving convolution and max pooling layers, while other methods use up to 2 layers.

Table 2: Test results for our method and several best methods in the literature, with the number of features used to train the final layer of classifiers.



(a) (b) (c)

Figure 5: Visualization of the features selected by our algorithm (sorted row-major by their final contribution to the classifier). (a) The sorted pooling regions from most important to least important; (b) the most important codes; (c) the least important codes.

4.3 Best Practice

Our best practice on the CIFAR-10 dataset was achieved by using a codebook size of 6,000, performing max pooling on overcomplete rectangular bins based on a 4×4 superpixel 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 this, we achieved an 83.11% accuracy on the testing data. To the best of our knowledge, this is the best published result on CIFAR-10. Table 2 lists the performance comparison against 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.

4.4 Analysis of Selected Features

The feature selection algorithm enables us to examine the importance of different receptive fields as well as codes for the classification area. To this end, we sort the pooling receptive fields and the codes by their contribution to the classifier, which is defined as the squared sum of the corresponding classifier weights. Figure 5 shows the sorted pooling regions from most important to least important, and the most/least important 100 codes selected by our algorithm.

Two types of pooling regions usually help: small pooling regions and thin regions spanning the whole horizontal or vertical scale of the image. we infer that on the one hand, as objects in the CIFAR-10 dataset are usually well-centered, their appearances have relatively small spatial displacement, favoring smaller spatial regions for pooling; on the other hand, cross-image bars provides us scene-level information, which may help us identify objects in the scene. Such explanation makes sense empirically too. For example, we found that objects like dogs and frogs favors small regions more while others like airplanes and boats favors cross-image bars more. The visualization of codes reveals that the most useful codes usually takes non-smooth appearance, instead of simple gabor-like

Classification on	Feature Selection on	Accuracy
CIFAR-100	CIFAR-10	**54.88**
	CIFAR-100	54.83
	Random Selection	54.48±0.25
CIFAR-10	CIFAR-100	78.88
	CIFAR-10	**80.17**
	Random Selection	78.95±0.20

Table 3: The performance of transferring pooled feature between CIFAR-10 and CIFAR-100 compared against natively learned features and random selection.

ones. Interestingly, when we run k-means, such discriminative codes tend to have a smaller number of cluster members than the smoother but less discriminative dictionary entries. More of such clustering centers appear with the increase of the number of clusters K. This result partially explains why current overcomplete representations almost always increases performance and overfitting is rarely observed even with a huge dimensionality [22]. We also note that similar findings about the discrimination power of dictionary entries have also been found and discussed in [1].

4.5 Transferring Class-independent Pooling Knowledge

Beyond classifying existing labels during training, we are also interested in examining whether the learned receptive fields work equally well on unseen classes. While several papers have suggested that simple cells in V1 performs sparse encoding independent from class labels, and that unsupervised feature learning performs well for the coding step, little is known about the pooling strategy. Learning class-independent pooling knowledge is closely connected to the visual attention model [11], which answers the question "what does an object look like in general".

To examine the performance of our method against new classes, we utilize the CIFAR-100 dataset, which contains 100 categories with 500 training examples per class. We extract features in the same fashion, and train the SVM classifier with learned codes and receptive fields from CIFAR-10. The classification result is compared against the accuracy rate obtained from directly learning the receptive fields on CIFAR-100, and a baseline that does random feature selection from the same set of overcomplete features. For a fair comparison, all methods use a codebook size of 1,600 and select 6,400 dimensional features. We also tested learning pooled features on CIFAR-100 and testing on CIFAR-10, and the performances are reported in Table 3.

The result we obtained showed a mixed message. While the features are leaned from CIFAR-10, they perform well on the CIFAR-100 dataset, and the performance is even better than natively learned features. For the other direction, transferring learned features does not show a statistically significant difference compared with random selection. A possible explanation of such scenario may be that with less training data per class on CIFAR-100, the feature selection algorithm may suffer more from overfitting than it does on CIFAR-10, reducing the generalization ability of the learned features. In general, our experiment does show the hope for a better and class-independent pooling strategy. Possible future work may involve utilizing larger-scale image databases, and exploring pooled feature learning in an unsupervised approach, which may further reveal valuable pooling strategies.

5 Conclusion

In this paper we examine 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 field increases performance even with a significantly smaller codebook size at the coding layer. We adopt 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 achieve the best published performance on the CIFAR-10 dataset, using a much lower dimensional feature space than previous methods did. 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.

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

[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.

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

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

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

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

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

[9] 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.

[10] 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.

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

[12] Y Jia, M Salzmann, and T Darrell. Factorized latent spaces with structured sparsity. In *NIPS*, 2010.

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

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

[15] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178, 2006.

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

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

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

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

[20] J Wang, J Yang, K Yu, F Lv, T Huang, and Y Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 1–8, 2010.

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

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

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

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