When Naïve Bayes Nearest Neighbors Meet Convolutional Neural Networks

Ilja Kuzborskij[1,2,3], Fabio Maria Carlucci[1], Barbara Caputo[1,2]

[1]Sapienza Rome University, Dept. of Computer, Control and Management Engineering, Italy
[2]Idiap Research Institute, Switzerland
[3]École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
{kuzborskij, fmcarlucci, caputo}@dis.uniroma1.it

Abstract

Since Convolutional Neural Networks (CNNs) have become the leading learning paradigm in visual recognition, Naive Bayes Nearest Neighbor (NBNN)-based classifiers have lost momentum in the community. This is because (1) such algorithms cannot use CNN activations as input features; (2) they cannot be used as final layer of CNN architectures for end-to-end training , and (3) they are generally not scalable and hence cannot handle big data. This paper proposes a framework that addresses all these issues, thus bringing back NBNNs on the map. We solve the first by extracting CNN activations from local patches at multiple scale levels, similarly to [13]. We address simultaneously the second and third by proposing a scalable version of Naive Bayes Non-linear Learning (NBNL, [7]). Results obtained using pre-trained CNNs on standard scene and domain adaptation databases show the strength of our approach, opening a new season for NBNNs.

1. Introduction

The current easy access to terabytes of visual data, combined with the impressive ability of deep learning algorithms to exploit them, has led to a paradigm shift in visual recognition over the last few years. The so called shallow architectures, i.e. learning algorithms consisting of 1-3 levels, have survived only when *(a)* they have been able to scale over very large amount of data and classes (i.e. $\geq 10^6$ and $\geq 10^3$ respectively); *(b)* they could be used as the final layer of Convolutional Neural Network (CNN)s, allowing for end-to-end learning, and/or *(c)* they could use effectively the activation layers of pre-computed CNNs [5, 3] as input features. All shallow architectures which do not comply with these requirements have started to fade away.

One of those fading algorithms is the Naïve Bayes Nearest Neighbour (NBNN) classifier [2]. Indeed, the key requisites of NBNN-based approaches do not fit well with CNNs. To begin with, they require local feature representations without any vector quantization, as opposed to the global feature representation derived from the CNN activation layers [5, 3]. Moreover, NBNN-based algorithms rely on the Image-2-Class (I2C) paradigm: for every image, each local descriptor is considered as independently sampled from a class-specific feature distribution. Hence, each descriptor votes for the most probable class, and the collection of votes is used to label each image. As opposed to that, CNNs operate on another classification principle. These two intrinsic features of NBNN approaches led to a strong generalization ability, showcased by remarkable results in place classification [7] and domain adaptation [42]. Still, as of today no solution has been found for bridging these two approaches.

This paper fills this gap. We propose a simple way to compute local features from whole images, using pre-trained CNNs. Our starting point is the paper of Gong *et al*. [13], on which to a large extent we build. We extract CNN activations for local patches at multiple scale levels. As opposed to [13], we do not perform any pooling or concatenation. The resulting features can be used directly as input to any NBNN-based classifier. However, the total number of examples can be very large, especially when doing a dense sampling for the patches and tackling large scale problems. To deal with this, while at the same time maximizing the predictive power of NBNN-based approaches, we propose a scalable version of Naive Bayes Non-linear Learning (NBNL, [7]). NBNL tries to circumvent limitations of NBNN through non-linear learning powered by Latent Locally-Linear SVM [8], that to our knowledge is the current state of the art among NBNN-based classifiers. Our stochastic algorithm retains the generality and robustness of the original method, yet it wins by having low memory footprint. At the same time, it considerably increases its scalability during training, making it applicable also to problems

with hundreds of classes, where a dense sampling strategy might lead to 10^7 features or more. Moreover, we show that our smoothed version of NBNL could in principle be used as final layer for an end-to-end training of a CNN. Figure 1 shows schematically the whole framework.

We assess our approach on scene recognition and domain adaptation datasets. These two research areas are those where NBNN-based algorithms showed more promise in the pre-CNN era. We show that on the Scene 15 [22], UIUC Sports [23], and MIT Indoor [33] datasets we achieve the state of the art among single-features approaches. To the best of our knowledge, these are the first results reported where an NBNN-based method achieves the state of the art not only among other NBNN-based approaches, but also among traditional techniques. Regarding domain adaptation, experiments on the Office+Caltech256 [12] dataset show that by just using our approach to build a source classifier and then testing it on the target, we achieve remarkable results in the unsupervised setting, and the state of the art in the semi-supervised one. This further underlines the current power and remarkable future potential of our contribution.

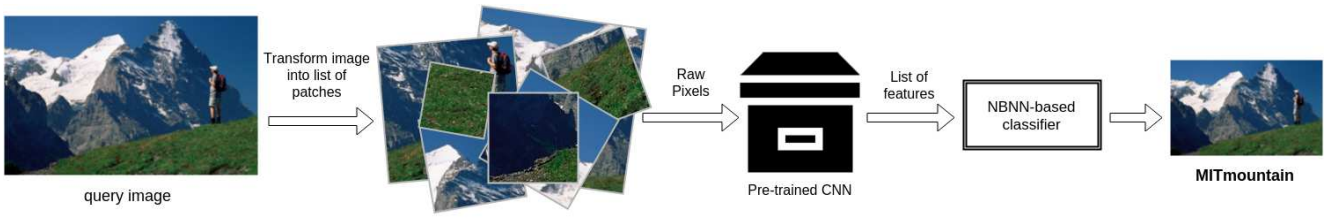
2. Related Work

NBNN [2] is a learning-free non-parametric image classification scheme. It proved its robustness and generalization ability on many different tasks, from image recognition [2, 43, 41, 40] to domain adaptation [43, 42] to action recognition [48]. A number of works went on to improve the generalization performance of NBNN by adding layers of learning. For example, in [45] the authors included a metric learning procedure, thus altering the metric space of 1-nearest neigbour. A similar idea was also investigated by Tommasi and Caputo [42], demonstrating that a plain NBNN performs very well in the domain adaptation setting, and even better when tuned-up with metric learning. Another route was pursued by works focused on patch subset selection and weighting [41, 7, 46]. A somewhat orthogonal direction was explored by fusing NBNN with kernel methods, proposing NBNN kernels [43, 34], which could be used in conjunction with linear classifiers and ultimately combined with another kernels over traditional representations. All of these methods were proposed before the advent of modern features induced by CNN, and typically were evaluated on feature descriptors such as SIFT or SURF, extracted from very small image patches. Since the seminal paper of Donahue *et al*. [5], the state of the art has been provided by CNNs' activations. Building on this, Gong *et al*. [13] proposed a multiscale orderless pooling of CNN features extracted from densely sampled patches. Later, Liu *et al*. [25] proposed a similar pooling scheme, called cross-convolutional-layer pooling, which focuses on using different convolutional layers together.

In this work, we revisit NBNN considering its power in conjunction with CNN features, in both categorization and domain adaptation scenarios. Many proposed algorithms built on top of NBNNs were thoroughly empirically studied [40]. However, the amount of training data hardly ever exceeded $\approx 10^4$ images. This stems from the limitations of the nearest-neighbor search – the need to store all or most of training data, and the curse of dimensionality that is often suffered by non-parametric algorithms. Some variations have been proposed to improve the time and space complexity of NBNNs. McCann and Lowe [29] proposed to build one single search structure for all the classes and to consider only neighboring descriptors, thus offering an increase in performance. In Naïve Bayes Non-linear Learning (NBNL) [7], the authors retained the idea of patch-based classification as in NBNN, but followed the way of non-linear parametric classification. This allowed them to achieve a compact representation of the classes by learning a set of prototypes, allowing fast testing and improved accuracy. Unfortunately, their method was confined to the batch setting without much improvement in scalability compared to NBNN. In this paper we further develop the idea of NBNL by proposing a scalable stochastic *locally-linear* formulation, drawing inspiration from [7] and [8].

Many works in machine learning, such as [36], reside on the assumption that, although natural data live in a high-dimensional space, they are embedded into a low-dimensional manifold. Such algorithms try to learn about the manifold under the assumption that looking close enough, or *locally*, it appears approximately linear, thus can be captured by an hyperplane. A well-known stream of works on Local Coordinate Coding (LCC) [50, 49, 44] aims to learn the set of hyperplanes and weights that combine them locally. Often, this is done in the unsupervised way by minimizing the reconstruction error [50, 49, 52]. In these works a special attention is given to local weights of hyperplanes, or *codes*, which in visual learning problems are used as features. This approach was taken further by Locally-Linear Support Vector Machine [21], where codes are first found through clustering together with nearest-neigbour search, and then hyperplanes are learned in a single optimization problem. As these methods use separate unsupervised learning stage, they are unaware of the underlying discriminative task and scalability depends on the efficiency of this pre-training. This limitation is countered in the literature on Latent SVM [6] and Multiclass Latent Locally-Linear (ML3) SVM [8], where both, hyperplanes and codes are learned simultaneously through discriminative learning problem. Despite non-convexity, smart relaxations and optimization methods like Concave-Convex Procedure (CCCP), enable them to work well in practice. Unfortunately, these are typically batch algorithms with heuristical initialization [10], sometimes guided by in-domain knowledge, such as mining hard-negatives [6]. Other works

Figure 1: An example illustrating our framework bridging across NBNN-based methods and CNNs for the scene classification problem. Given a query image, we first compute CNN activations for local patches at different scales, from a pre-trained architecture. The resulting feature representation can be fed to any NBNN-based classifier, that will then output the image label. In the paper, we used [53] as pre-trained CNNs, and a scalable version of NBNL [7] as classifier. Note that the framework holds also for other choices of one or both of these two components.



proposed to scale up learning in this setting [19, 31], however, none of them demonstrated real scalability empirically. In this work we address these limitations proposing a simple scalable Stochastic Multiclass Latent Locally-Linear SVM, which does not require initialization tricks and easily handles the order of 10^6 training examples.

Our locally-linear formulation also conceptually reminds non-linearity used in Maxout Networks [14]. However, unlike [14], the inputs are weighted and combined with controlled degree of smoothness, which allows us to use analytic form of non-linearity. Thus, Maxout non-linearity is a special case of the locally-linear rule we employ.

3. Computing Local CNN Activations

As mentioned before, a key requirement for any NBNN-based framework is to deal with features that capture local information about the image. This concretely means to extract from each whole image a set of local patches at multiple scales, and compute feature descriptors from them. Following [13], we decide here to create orderless image representations from pre-trained CNN by extracting deep activation features from patches obtained at increasingly finer scales. The effectiveness of such features will depend on several designer choices, from the pre-trained CNN chosen, to the sampling rate for the patches, the patch size, and the computed CNN activations. In the following we discuss these points and our own designer choices.

Pre-trained CNN. The first hyper-parameter to chose is the CNN architecture to be used for computing the activations. The current off-the shelf state of the art choice for this task on whole images is the Caffe implementation [17], pre-trained on ILSVRC [37]. We decided to follow this route here with respect to the architecture type. As one of our benchmarks is the scene classification problem, we decided to use their network trained on a hybrid dataset composed from Places-205 [53] and ILSVRC [37]. Note that other architectures like VGG [3] or OverFeat[38] could be used in the same framework. Note also that, for any given CNN architecture within this framework, fine tuning on a validation set might further improve results.

Patch Extraction. The second set of hyper parameters to tune are those specifically related to the patch extraction, i.e. the sampling rate for the patches, the patches size and the number of scales. Regarding the sampling rate, we considered two patch sampling settings: (a) dense, with around 400 patches per image, and (b) sparse, with approximately 100 patches per image. Since each image has different proportions, the sampling stride was dynamically computed in order to approximately achieve the desired number of patches. Regarding the patches size and number of scales, we did set the size of the smallest patch from $\{16\text{px}, 32\text{px}, 64\text{px}\}$, and further doubled the size with each level. For example, if the size of the smallest patch is 16px and we consider 3 levels, we will extract patches of size 16×16px (level 1), 32×32px (level 2) and 64×64px (level 3). As level 0, we considered the whole image, where before extracting the patches, each image is resized to reduce its longest side to 200 pixels.

CNN activations. Finally, we have to choose the fully connected layer of CNN, whose outputs will be used as features. The most popular choice in the literature, adopted also in [53], is to take the output of the seventh fully connected layer after ReLU transformation, that is setting all negative values to zero. We compared this setting with other possibilities, namely taking the output of the sixth layer, on some pilot experiments. We found that also in the NBNN framework the mainstream approach is the most effective.

4. Scalable Naïve Bayes Non-linear Learning

In this section we describe our main technical contribution, a novel Stochastic Multiclass Latent Locally-Linear (STOML3) SVM, designed to resolve the scalability issues of NBNN. Applied to the NBNN learning framework, it results in a scalable Naïve Bayes Non-linear Learning technique (sNBNL). First we introduce the background (sections 4.1-4.3), and present our algorithm in Section 4.4.

4.1. Definitions

We first introduce the notation and technical definitions used in the rest of the paper. Denote with small and capital bold letters respectively column vectors and matrices, e.g. $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_d]^T \in \mathbb{R}^d$ and $\boldsymbol{A} \in \mathbb{R}^{d_1 \times d_2}$. We will use a non-negative truncation function $[x]_+ = \max\{0, x\}$, and for the vectors, $[\boldsymbol{x}]_+ = [\max\{0, x_1\}, \ldots, \max\{0, x_d\}]^\top$. To denote the largest element of the vector, we will use notation $\max\{\boldsymbol{x}\} = \max\{x_1, \ldots, x_d\}$. We denote enumeration sets by $[n] = \{1, \ldots, n\}$ for $n \in \mathbb{N}$. Denote by $\mathcal{X}$ and $\mathcal{Y}$ respectively the input and output space of the learning problem. Let the training instance I, w.l.o.g., be composed from n sub-instances, $I = \{\boldsymbol{x}_i\}_{i=1}^n$. Then we denote the training set of size m by $S = \{(I_i, y_i)\}_{i=1}^m$, drawn from the probability distribution $\mathcal{D}$ over $\mathcal{X}^n \times \mathcal{Y}$. We will focus on the c-class classification problem so $\mathcal{Y} = [c]$, and, w.l.o.g., $\mathcal{X} = \{\boldsymbol{x} : \|\boldsymbol{x}\|_2 \leq 1, \boldsymbol{x} \in \mathbb{R}^d\}$. To measure the accuracy of a learning algorithm, we have a non-negative convex *loss* function $\ell(f(\boldsymbol{x}), y)$, which measures the cost incurred predicting $f(\boldsymbol{x})$ instead of y. Finally we will denote a *one nearest neighbor* function w.r.t. the support set Z by $\boldsymbol{\pi}_Z(\boldsymbol{x}) = \arg\min_{\boldsymbol{z} \in Z} \|\boldsymbol{x} - \boldsymbol{z}\|_2$. Alternatively, for $d \times n$ neighbor matrices we will use the notation $\boldsymbol{\pi}_{\boldsymbol{Z}}(\boldsymbol{x}) = \arg\min_{\boldsymbol{z} \in \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}} \|\boldsymbol{x} - \boldsymbol{z}\|_2$.

4.2. Naïve Bayes Nearest Neighbor Classification

The idea behind NBNNs [2] is to treat each image as a collection of uniformly or randomly sampled patches. Let I be the set containing visual descriptors of patches in the test image, let $X_1, \ldots, X_n$ be random variables taking values in the space of these descriptors, and let Y be taking values in the label set. Denoting by $p_Y(y|I)$ the unknown conditional probability density function, the NBNN predictor is,

$$f(I) = \arg\max_{y \in \mathcal{Y}} p_Y(y \mid I) \,. \tag{1}$$

The key statistical assumption made in NBNN is that patches are conditionally independent given the class. In addition, assuming that $p_Y(y)$ is uniform and switching to log-likelihood of $p_Y(y\,|I)$, we have that,

$$f(I) = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{n} \log(p_{X_i}(\boldsymbol{x}_i \mid y)) \,. \tag{2}$$

Since p_{X_i} is unknown, NBNN resorts to the non-parametric Kernel Density Estimator (KDE) [16] with Gaussian kernel function, and further lower-bounds the log-likelihood by Jensen's inequality, to make the predictor computationally efficient. In this form prediction involves nearest neighbor search, which can be very efficient when the intrinsic dimension of the data is small [4]. Denoting the support of the class y by $W_y = \cup_{(I', y') \in S \,:\, y = y'} I'$, the approximated empirical NBNN predictor is then,

$$\widehat{f}(I) = \arg\min_{y \in \mathcal{Y}} \sum_{\boldsymbol{x} \in I} \|\boldsymbol{x} - \boldsymbol{\pi}_{W_y}(\boldsymbol{x})\|^2 \,. \tag{3}$$

4.3. Naïve Bayes Non-Linear Learning

As NBNN is a nearest-neighbor-based approach, it shares its well-known scalability limits. Few works have explored the potential of NBNN-like schemes surpassing the order of 10^4 training examples. Here we review the recently proposed Naïve Bayes Non-linear Learning (NBNL) [7] that scales NBNN through parametric learning. It will be the starting point for our scalable algorithm.

Let $W = (\boldsymbol{W}_1, \ldots, \boldsymbol{W}_c) \in \mathbb{R}^{d \times k \times c}$ be the collection of k-sized supports of NBNN in matrix notation. Following [7], we will refer to the columns of any support $\boldsymbol{W}_y$ as *prototypes*. We will also assume that all prototypes have bounded norm, that is $\|\boldsymbol{w}\|^2 \leq \tau$. NBNL rests upon the observation that NBNN minimizes,

$$\begin{aligned} \sum_{\boldsymbol{x} \in I} \|\boldsymbol{x} - \boldsymbol{\pi}_{\boldsymbol{W}_y}(\boldsymbol{x})\|_2^2 &= \sum_{\boldsymbol{x} \in I} \min_{i \in [k]} \|\boldsymbol{x} - \boldsymbol{w}_{y,i}\|_2^2 \\ &\leq |I|(1+\tau) - 2 \sum_{\boldsymbol{x} \in I} \max\left\{\boldsymbol{W}_y^\top \boldsymbol{x}\right\} \,. \end{aligned} \tag{4}$$

The right hand side can be minimized over $y \in \mathcal{Y}$, similarly as in (3), which yields the *NBNL predictor*

$$f^{\text{nbnl}}(I) = \arg\max_{y \in \mathcal{Y}} \frac{1}{|I|} \sum_{\boldsymbol{x} \in I} \max\left\{\boldsymbol{W}_y^\top \boldsymbol{x}\right\} \,. \tag{5}$$

The key idea is that prototypes in such a predictor need not be fixed, but can be *learned*. Fornoni and Caputo [7] proposed to learn prototypes through the regularized empirical risk minimization. Considering f^{nbnl}, the problem would be to minimize the following over W,

$$\frac{1}{m} \sum_{i=1}^{m} \ell\left(\frac{1}{n} \sum_{\boldsymbol{x} \in I_i} \max\left\{\boldsymbol{W}_{y_i}^\top \boldsymbol{x}\right\}, y_i\right) + \lambda \sum_{l \in \mathcal{Y}} \|\boldsymbol{W}_l\|_F^2 \,. \tag{6}$$

However, in [7], they ultimately proposed to solve a simpler relaxed problem (due to Jensen's inequality),

$$\min_W \left\{\frac{1}{mn} \sum_{i=1}^{mn} \ell\left(\max\left\{\boldsymbol{W}_{y_i}^\top \boldsymbol{x}_i\right\}, y_i\right) + \lambda \sum_{l \in \mathcal{Y}} \|\boldsymbol{W}_l\|_F^2\right\} \,. \tag{7}$$

Problem (7) is generally addressed by the family of latent [6] and *locally-linear* SVMs [21, 8]. In particular, [7] employed a non-linear ML3 Support Vector Machine (SVM) [8], which we briefly review next.

Multiclass Latent Locally-Linear (ML3) SVM. In ML3 SVM one aims to solve a problem similar to (7). ML3 SVM

is a locally-linear parametric classification algorithm, where we assume that in a given small locality the optimal decision boundary is approximately linear [50, 21, 18, 19]. Usually, in locally-linear versions of SVM we consider score functions $f^{\text{LL}}_{\boldsymbol{W}}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{\beta}(\boldsymbol{x})$, where $\boldsymbol{\beta}(\boldsymbol{x})$ is a function specifying local combination of hyperplanes $\boldsymbol{W}$ at a particular point of the input space. Typically one has to choose $\boldsymbol{\beta}(\boldsymbol{x})$ before solving the main optimization problem [6, 50, 21]. This amounts to the separate procedure dedicated just to learn and fix weights $\boldsymbol{\beta}(\boldsymbol{x})$. ML3 SVM addresses this by the score function with automatic weighting,

$$f^{\text{ML3}}_{\boldsymbol{W}}(\boldsymbol{x}) = \max_{\|\boldsymbol{\alpha}\|_p \leq 1, \boldsymbol{\alpha} \succeq 0} \{\boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{\alpha}\} = \|[\boldsymbol{W}^\top \boldsymbol{x}]_+\|_q \,, \quad (8)$$

for any $p \in [1; +\infty]$ and $q = \frac{p}{p-1}$. Given a point $\boldsymbol{x}$, this rule leads to the combination of hyperplanes, such that the margin of a combined linear classifier is maximized on $\boldsymbol{x}$.

The objective function of ML3 SVM is non-convex, however, by posing it as a difference of convex functions, we can find a reasonably good solution by Concave-Convex Procedure (CCCP) [51]. This essentially confines the algorithm to the batch setting, because we need to solve a separate convex optimization problem at every CCCP iteration. Besides its batch nature, ML3 heavily relies on heuristic weight initialization by first solving a linear SVM problem.

4.4. Stochastic ML3 SVM

In this section we fix the limitations of ML3 by introducing a novel scalable *stochastic* formulation, conceptually similar to the one of ML3. Namely, we propose a Stochastic Multiclass Latent Locally-Linear (STOML3) SVM which can run online, is free from any initialization tricks, and enjoys stationary point convergence guarantee. This stochastic formulation allows to use NBNL at scales out of reach for ML3 SVM and NBNN. We call this new version, the *scalable NBNL (sNBNL)*.

Rather than solving a regularized empirical risk as in (7), we will aim at minimizing a regularized risk directly, similarly as in the popular Stochastic Gradient Descent (SGD) approach to learning. More formally, our goal is to solve,

$$\min_{W} \left\{ \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{D}} [\ell(W, (\boldsymbol{x}, y))] + \lambda \sum_{l \in \mathcal{Y}} \|\boldsymbol{W}_l\|_F^2 \right\} , \quad (9)$$

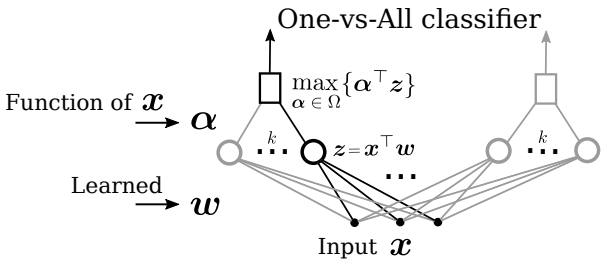
where we chose a differentiable multiclass logistic loss,

$$\ell(W, (\boldsymbol{x}, y)) = \log \left(1 + \sum_{r \neq y} \exp \left(f^{\text{ML3}}_{\boldsymbol{W}_r}(\boldsymbol{x}) - f^{\text{ML3}}_{\boldsymbol{W}_y}(\boldsymbol{x}) \right) \right) .$$

In practice we cannot solve (9) directly, since $\mathcal{D}$ is unknown, thus the gradient cannot be computed. However, we can still compute an unbiased estimate of the gradient given a point $(\boldsymbol{x}, y) \sim \mathcal{D}$, and thus update the solution iteratively. Alike the batch formulation of ML3 SVM, the resulting objective function is non-convex. We approach (9) through the Stochastic Majorization-Minimization (SMM) framework [27], which unlike SGD, provides a stationary point convergence guarantee, and converges faster in practice [28, 35]. We summarize the STOML3 SVM in pseudocode, and defer its technical derivation details to the following section. The computational complexity of every STOML3 SVM update is in $\mathcal{O}(|\mathcal{Y}|kd)$, however in practice we bringing it down to $\mathcal{O}(|\mathcal{Y}|)$ through GPU optimization.

Connection to Neural Network Learning. Latent locally-linear classification, ML3 SVM, and STOML3 SVM can be interpreted as a variant of a shallow artificial neural network, Figure 2. The main difference between

Figure 2: Latent locally-linear classification.



traditional models such as multilayer perceptrons, is that the hidden layer consists of linear units ($\boldsymbol{z} = \boldsymbol{w}^\top \boldsymbol{x}$), whereas the weights of the output layer, $\boldsymbol{\alpha}$ are adjusted automatically depending on the outputs of hidden layer $\boldsymbol{z}$, thus for learned $\boldsymbol{w}$, $\boldsymbol{\alpha}$ is a function of input $\boldsymbol{x}$. Specifically, these weights are adjusted to maximize the margin by combining outputs of hidden units. Clearly, for different regions of the input space, resulting combinations are different, yielding non-linear decision surface.

From the artificial neural network learning point of view, it would be interesting to consider deeper architectures of STOML3. Another possibility would be to combine it with convolutional layers to investigate end-to-end locally-linear classification. We leave these directions to the future work.

4.4.1 Derivation

To derive STOML3 SVM we use the Stochastic Majorization-Minimization (SMM) framework proposed by Mairal [27]. SMM deals with minimization of a differentiable function that has a form of expectation, by minimizing its simpler approximate convex upper-bound. Specifically, after we sample a training example, we minimize an upper bound on the term inside of expectation with realization fixed. In our case, the objective is (9),

Stochastic Multiclass Latent Locally-Linear (STOML3) SVM

Input: W^0 (initial prototypes), $\lambda \in \mathbb{R}_+$ (regularization parameter), $q \geq 1$ (boundary smoothness).
Output: W (learned prototypes).

$\phi(\boldsymbol{z}) := \|[\boldsymbol{z}]_+\|_q$

$\boldsymbol{A}_k^0 \leftarrow \mathbf{0}, \boldsymbol{B}_k^0 \leftarrow \mathbf{0}, \bar{\boldsymbol{W}}_k^0 \leftarrow \mathbf{0}, \quad \forall k \in \mathcal{Y}$.
for $t = 1, 2, \ldots$ **do**
  Draw example $(\boldsymbol{x}_t, y_t) \sim \mathcal{D}$.
  $\gamma_t \leftarrow 1 - \frac{1}{\sqrt{t}}$
  $\boldsymbol{s}_k \leftarrow \boldsymbol{W}_k^{t-1\top} \boldsymbol{x}_t, \ \forall k \in \mathcal{Y}$
  for $k \in \mathcal{Y}$ **do**
    $\boldsymbol{A}_k^t \leftarrow \gamma_t \boldsymbol{A}_k^{t-1} + \frac{1}{\sqrt{t}} \frac{\exp(\phi(\boldsymbol{s}_k))}{\sum_{j\in\mathcal{Y}} \exp(\phi(\boldsymbol{s}_j))} \nabla\phi(\boldsymbol{s}_k) \boldsymbol{x}_t^\top$
    $\boldsymbol{B}_k^t \leftarrow \gamma_t \boldsymbol{B}_k^{t-1} + \frac{\mathbb{I}\{k=y_t\}}{\sqrt{t}} \nabla\phi(\boldsymbol{s}_{y_t}) \boldsymbol{x}_t^\top$
    $\bar{\boldsymbol{W}}_k^t \leftarrow \gamma_t \bar{\boldsymbol{W}}_k^{t-1} + \frac{1}{\sqrt{t}} \boldsymbol{W}_k^{t-1}$
    $\boldsymbol{W}_k^t \leftarrow \frac{1}{1+\lambda} \left(\bar{\boldsymbol{W}}_k^t - \boldsymbol{A}_k^t + \boldsymbol{B}_k^t \right)$
  end for
end for

and thus for a realization $(\boldsymbol{x}, y) \sim \mathcal{D}$ we need to specify a convex upper-bound of a regularized loss function,

$$g(W) := \ell(W, (\boldsymbol{x}, y)) + \lambda \sum_{l\in\mathcal{Y}} \|\boldsymbol{W}_l\|_F^2 \ . \qquad (10)$$

More formally, in SMM such a convex upper-bound is called the *surrogate* function of an objective, defined as:

Strongly Convex First-Order Surrogate Functions [27]. *Fix $V \in \mathbb{R}^{d\times k\times c}$, and let h be a strongly convex function such that $h \geq g$ and $h(V) = g(V)$. Let $h - g$ be differentiable and the gradient $\nabla(h-g)$ be L-Lipschitz continuous. We will call h the first order surrogate function of g.*

We can choose among many different surrogates, but we have to keep in mind that it should be easily minimized with every incoming training example. That said, we choose, $h(W) = g_1(V) + \nabla g_1(V)^\top (W - V) + \frac{L}{2}\|W - V\|^2 + g_2(W)$, where $g_1 = \ell$, g_2 is the regularizer. This choice is motivated by efficiency, because we can find minimum of $h(W)$ analytically. It is also not hard to see that h is a strongly convex first-order surrogate function. Thus, given optimal W, the rest of the derivation follows the optimization template of Mairal [27], summarized in our pseudocode.

5. Experiments

In this section we test experimentally our framework. We considered two tasks, scene recognition and domain adaptation, where in the past NBNN methods showed promise. Our experiments aim to verify two claims: first, that such methods coupled with local CNN activations at multiple scales are able to achieve results competitive with, or even better than, end-to-end, fine tuned CNN architectures. Second, that scalable NBNL outperforms NBNN, thus paving the way for the use of our approach on large scale scenarios that have been so far prohibitive for NBNN methods. In the rest of the section we describe the datasets and experimental settings used, and the variants of our framework that were tested (Section 5.1). Section 5.2 describes the results obtained in scene recognition, exploring how the performance changes when varying the parameters relative to the patch extraction, and the scalability of the approach. Section 5.3 reports results obtained in the domain adaptation setting.

5.1. Experimental Settings

Datasets. For the scene recognition setting, we used the Scene 15 [22], UIUC Sports [23], and MIT Indoor [33] databases. For Scene 15, we used 100 images per class for training and 100 for testing. For UIUC Sports, we used 70 images per class for training and 60 images for testing. For MIT Indoor, we used 80 images per class for training and 20 for testing. These choices are all consistent with the standard protocols reported in the literature. Each configuration is tested on 5 splits. For the large scale experiments, we used the SUN-397 database [47] that totals 1.6 million image patches. We strictly followed the experimental procedure described in [47]. For all scene experiments, we concatenated the CNN activations with the absolute position of every patch. For the domain adaptation scenario, we considered the Office + Caltech database [12], which contains a subset of ten classes shared between Office and Caltech256 [15]. Here we keep 20 images per class for training (15 if the target is either Webcam or DSLR) and use the rest as test set. Each configuration was tested on 10 splits.

Baselines For every scenario, for every setting, we always used the following three variants of our framework: (1) *CNN-NBNN*: this consists of using the NBNN classifier as originally proposed [2] , combined with the local CNN activations. (2) *CNN-NBNL*: the same as (1), using NBNL as classifier [7]. (3) *CNN-sNBNL*: the same as (1), (2), but using our scalable version of NBNL.

5.2. Scene Classification Experiments

We performed extensive experiments over Scene 15, UIUC Sports and MIT Indoor for assessing how performance changes when varying the parameters of the CNN activation extraction. Specifically, we varied the sampling density, patch size and the number of levels. We also compared results when taking the activations before or after ReLU. As classifier, we always used NBNN (preliminary experiments using also NBNL and sNBNL did not show any significant variation in behaviors). Figure 4 reports a representative set of our findings. We see that larger patch sizes generally yield better performance, but combining patches taken at different scales further improves accuracy. For ex-

Figure 3: Top-scoring patches from "snowboarding" and "polo" categories of Sports 8 dataset.

ample, using only 64×64px patches gives a worse accuracy than using 32×32px and 64×64px patches. This shows that distinct scales hold complementary information. Dense sampling does not improve the accuracy significantly.

Overall, using together 32px, 64px, and 128px patches seems to be the best and most stable configuration. The stability of results breaks down when we supply smaller patches of 16px. We speculate that at this patch size there is not enough visual information for CNN to provide meaningful representation. Finally, we note that CNN features extracted before ReLU generally perform better. That said, in the rest of the paper we always use simultaneously 32px, 64px, and 128px patches, no ReLU and sparse sampling.

Next we compare sNBNL against NBNL in efficiency and effectiveness. Our goal is to confirm the ability of sNBNL to reach the same results as NBNL at a lower computational cost. Table 2 shows the results obtained using NBNL and sNBNL on the three databases, in terms of accuracy and training time. We see that the two algorithms achieve basically the same results, as confirmed by a sign-test ($p < 0.05$). Instead w.r.t. the training time these differences are remarkable, with sNBNL achieving on average a speed up of 25 times compared to NBNL. This is a first experimental confirmation of the scalability of our approach. Table 1 compares our results with previous work. We see that we achieve consistently the best accuracy among the single cue methods. This is impressive for an approach that uses an off-the-shelf pre-trained CNN, without any fine tuning. Moreover, on the Scene 15 database, our performance surpasses also that of multi-cue approaches.

We conclude this section by probing the potential of our framework on a larger scale experiment. We run experiments on the SUN-397 [47] dataset. Note that this dataset is out of reach for NBNN, and prohibitive also for NBNL. We trained GPU-optimized implementation of STOML3 SVM

Table 1: Comparison of previous work with our approach. Legend: **bold** indicates the best performance among single feature methods, **red bold** indicates the overall best.

Method	Scene 15	Sports 8	MIT67
NBNN (Surf)[7]	72.8	67.6	–
NBNL (Surf)[7]	82.42	85.54	42.15
CNN-NBNN	88.24 ± 0.99	94.46 ± 0.47	63.92 ± 1.63
Lin. SVM(CNN)	90 ± 0.63	94.16 ± 1.13	64.62 ± 1.04
CNN-NBNL	92.42 ± 0.64	$\mathbf{95.29 \pm 0.61}$	$\mathbf{73 \pm 0.36}$
CNN-sNBNL	$\mathbf{92.88 \pm 0.89}$	95.28 ± 0.68	72.79 ± 0.73
Hybrid CNN[53]	91.59	94.22	70.8
LScSPM[9]	89.78	85.27	–
MOP-CNN[13]	–	–	68.88
DDSFL + CAFFE[54]	92.81	**96.78**	**76.23**
ISPR + IFV[24]	91.06	92.08	68.50
CNN Fusion[20]	92.1	94.8	70.1

Table 2: NBNL vs sNBNL in accuracy, training and testing time in seconds, over the three scene recognition databases.

	Sports 8			Scenes 15			ISR 67		
	Acc.	Train	Test	Acc.	Train	Test	Acc.	Train	Test
NBNL	94.2	1024.4	13.9	91.5	5729.2	95.9	72.5	9690	63.2
sNBNL	95.2	63.5	0.4	91.6	210.3	1.9	72.7	304.2	1.3
Speed-up	-	×16	×34	-	×27	×50	-	×32	×49

in minibatches of 2500 examples on 10 splits originally proposed in [47]. As in the previous scene recognition experiments, we concatenated the absolute patch positions with the feature vector. We perform data standardization and we set the regularization parameter λ to 1 – note that even better results can be obtained by tuning it. CNN-sNBNL achieves a performance of $55.8 \pm 0.29\%$, which surpasses recently reported results by Zhou *et al.* [53] of $53.86 \pm 0.21\%$ and $54.32 \pm 0.14\%$. These were obtained by training a linear SVM on Hybrid and Places-205 CNN features respectively.

We also comment on the patch importance by showing the high-scoring patches in representative images. We focus on Sports-8 and select patches which have the highest score according to the STOML3 predictor (8). We highlight those and dim the rest of the image in Fig. 3. Notably, NBNL puts higher score on patches semantically related to the category.

We conclude that the reported results clearly showcase the power of our framework in the scene recognition setting.

5.3. Domain Adaptation Experiments

We report here experiments performed on the Office+Caltech database, both in the unsupervised and semi-supervised scenarios. Note that none of the three instantiations of our framework are a domain adaptation algorithm, hence we simply use each of them on the source data, and test the obtained classifier on the target. Concretely, in the unsupervised setting we simply train NBNN/NBNL/sNBNL on the source; for the semi-supervised setting, we add three target images to the source and proceed as for the unsupervised case. A similar experiment was first presented in [42], showing that NBNN gener-

Figure 4: Results obtained by NBNN on CNN features computed with different patch sizes, sampling rates, on three datasets.

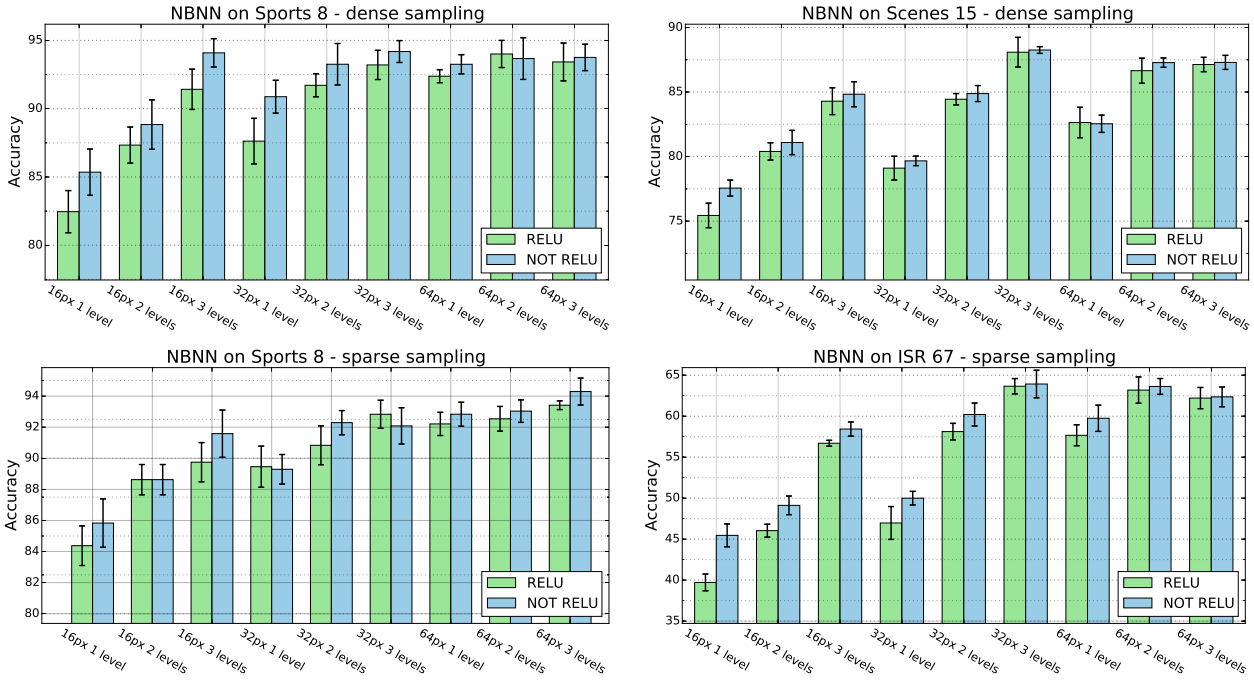


Table 3: Unsupervised domain adaptation results

Alg. \Dataset	A→ W	A→ C	W→ A	W→ C	C→ A	C→ W
NBNN[42]	31.8	31.3	37.4	26.8	41	28.4
DA-NBNN[42]	35	41	42	33	55	36
CNN-NBNN	60.23 ± 3.5	75.2 ± 1.0	66.87 ± 1.3	63.3 ± 1.2	79.03 ± 0.9	61.28 ± 4.6
CNN-NBNL	62.61 ± 3.5	71.61 ± 2.1	56.84 ± 2.6	50.08 ± 2.4	79.97 ± 2.3	61.05 ± 3.7
CNN-sNBNL	61.93 ± 3.7	72 ± 2	63.45 ± 1.9	55.81 ± 1.5	80.91 ± 2.0	64.84 ± 3.4
GFK[12]	35.7	37.9	35.5	29.3	40.4	–
SWAP[11]	37.6	41.3	38.2	32.2	46.2	46.1
Landmark[11]	46.1	45.5	40.2	35.4	56.7	49.5
LapCNN[26]	–	83.6	–	77.8	92.1	81.6
DDC[26]	–	84.3	–	76.9	91.3	85.5
DAN[26]	–	86	–	81.5	92	–

Table 4: Semi supervised domain adaptation results

Alg. \Dataset	A→ W	A→ C	W→ A	W→ C	C→ A	C→ W
NBNN[42]	56.9	34	43.5	31.6	50.2	57.7
DA-NBNN[42]	62	46	58	42	65	61
CNN-NBNN	$\mathbf{88.9 \pm 2.9}$	$\mathbf{76.93 \pm 1.7}$	$\mathbf{80.6 \pm 1.5}$	$\mathbf{70.5 \pm 1.7}$	84.67 ± 1.2	$\mathbf{90.03 \pm 1.9}$
CNN-NBNL	84.87 ± 3.7	74.31 ± 1.1	77.14 ± 2.4	68.17 ± 2.8	83.77 ± 1.5	86.52 ± 3.6
CNN-sNBNL	87.54 ± 2.3	76.74 ± 1.9	79.38 ± 1.6	70.17 ± 1.6	$\mathbf{85.62 \pm 1.1}$	87.28 ± 2.5
H-L2L[32]	77.1	38.6	51.6	34.0	55.32	–
DASH-N[30]	75.5	54.9	70.4	50.2	71.6	–
SDDL[39]	72	27.4	49.4	29.7	49.5	–
HMP[1]	70	51.7	61.5	46.8	67.7	–

alizes well across the domains without DA-specific design in mind. As features, we use the same configuration employed in the scene recognition experiments, that is patches of size 32px, 64px and 128px without ReLU. We also performed experiments with sparse sampling.

Tables 3,4 report the results obtained in the unsupervised and in the semi-supervised settings. We see that, in the unsupervised setting, our approach is powerful enough to outperform several learning-based baselines, in spite of its simplicity. Performance on the semi-supervised setting further improves, as we achieve the state of the art in all settings. We stress that this is accomplished by the methods that are *not* designed for domain adaptation scenario. Note that we could not run DA-NBNN, the only existing NBNN-based domain adaptation method on our local CNN multi scale activations because of its computational limitations. These results further confirm the power of the proposed framework, and its potential for future work.

6. Conclusions

This paper provides a method for using CNN activation features combined with NBNN-based classifiers. The two key ingredients are: (1) extraction of CNN activations from local patches at different scales, and (2) a scalable NBNN-based algorithm that exploits the learning power of locally linear SVMs. We present an instantiation of this framework using a pre-trained Caffe architecture, applied to the scene classification and domain adaptation problems. Results are very strong: on scene classification we achieve the state of the art among single cue methods on three widely used benchmark databases. On domain adaptation, the simple use of the framework on the source only, leads to promising results, competitive against many learning methods proposed so far. Future work will further explore the framework in an end-to-end setting and domain adaptation.

Acknowledgements. This work is supported by the ERC Starting Grant RoboExNovo.

References

[1] L. Bo, X. Ren, and D. Fox. Hierarchical matching pursuit for image classification: Architecture and fast algorithms. In *Advances in neural information processing systems (NIPS)*, 2011.

[2] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition (CVPR). IEEE Conference on*, 2008.

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference (BMVC)*, 2014.

[4] K. L. Clarkson. Nearest-neighbor searching and metric space dimensions. In G. Shakhnarovich, T. Darrell, and P. Indyk, editors, *Nearest-neighbor methods for learning and vision: theory and practice*, pages 15–59. MIT Press, 2006.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning (ICML)*, 2014.

[6] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference On*, 2008.

[7] M. Fornoni and B. Caputo. Scene recognition with naive bayes non-linear learning. In *Pattern Recognition (ICPR), International Conference on*, 2014.

[8] M. Fornoni, B. Caputo, and F. Orabona. Multiclass latent locally linear support vector machines. In *Asian Conference on Machine Learning (ACML)*, 2013.

[9] S. Gao, I. W.-H. Tsang, and L. Chia. Laplacian sparse coding, hypergraph laplacian sparse coding, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):92–104, 2013.

[10] R. Girshick and J. Malik. Training deformable part models with decorrelated features. In *Computer Vision (ICCV), IEEE International Conference on*, 2013.

[11] B. Gong, K. Grauman, and F. Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *ICML*, 2013.

[12] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2012.

[13] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision (ECCV)*, 2014.

[14] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International Conference on Machine Learning (ICML)*, 2013.

[15] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, Caltech, 2007.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements Of Statistical Learning*. Springer, 2009.

[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, 2014.

[18] C. Jose, P. Goyal, P. Aggrwal, and M. Varma. Local deep kernel learning for efficient non-linear SVM prediction. In *International Conference on Machine Learning (ICML)*, 2013.

[19] A. Kantchelian, M. C. Tschantz, L. Huang, P. L. Bartlett, A. D. Joseph, and J. Tygar. Large-margin convex polytope machine. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[20] M. Koskela and J. Laaksonen. Convolutional network features for scene recognition. In *ACM International Conference on Multimedia*, 2014.

[21] L. Ladicky and P. Torr. Locally linear support vector machines. In *International Conference on Machine Learning (ICML)*, 2011.

[22] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, IEEE Conference on*, 2006.

[23] L.-J. Li and L. Fei-Fei. What, where and who? classifying events by scene and object recognition. In *Computer Vision (ICCV), IEEE International Conference on*, 2007.

[24] D. Lin, C. Lu, R. Liao, and J. Jia. Learning important spatial pooling regions for scene classification. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2014.

[25] L. Liu, C. Shen, and A. van den Hengel. The treasure beneath convolutional layers: cross convolutional layer pooling for image classification. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2015.

[26] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning (ICML)*, 2015.

[27] J. Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[28] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[29] S. McCann and D. G. Lowe. Local naive bayes nearest neighbor for image classification. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2012.

[30] H. V. Nguyen. *Non-Linear and Sparse Representations for Multi-Modal Recognition*. PhD thesis, University of Maryland, 2013.

[31] H. Oiwa and R. Fujimaki. Partition-wise linear models. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[32] N. Patricia and B. Caputo. Learning to learn, from transfer learning to domain adaptation: A unifying perspective. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2014.

[33] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2009.

[34] K. Rematas, M. Fritz, and T. Tuytelaars. The pooled nbnn kernel: Beyond image-to-class and image-to-image. In *Asian Conference on Computer Vision (ACCV)*, 2013.

[35] N. L. Roux, M. Schmidt, and F. R. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[36] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, pages 1–42, 2015.

[38] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[39] S. Shekhar, V. M. Patel, H. Nguyen, and R. Chellappa. Generalized domain-adaptive dictionaries. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2013.

[40] R. Timofte, T. Tuytelaars, and L. Van Gool. Naive bayes image classification: beyond nearest neighbors. In *Asian Conference on Computer Vision (ACCV)*, 2013.

[41] R. Timofte and L. Van Gool. Iterative nearest neighbors. *Pattern Recognition*, 48(1):60–72, 2015.

[42] T. Tommasi and B. Caputo. Frustratingly easy nbnn domain adaptation. In *Computer Vision (ICCV), IEEE International Conference on*, 2013.

[43] T. Tuytelaars, M. Fritz, K. Saenko, and T. Darrell. The nbnn kernel. In *Computer Vision (ICCV), IEEE International Conference on*, 2011.

[44] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2010.

[45] Z. Wang, Y. Hu, and L.-T. Chia. Image-to-class distance metric learning for image classification. In *European Conference on Computer Vision (ECCV)*, 2010.

[46] P. Wohlhart, M. Kostinger, M. Donoser, P. M. Roth, and H. Bischof. Optimizing 1-nearest prototype classifiers. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2013.

[47] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), IEEE Conference on*, 2010.

[48] X. Yang and Y. Tian. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. In *Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Conference on*, 2012.

[49] K. Yu and T. Zhang. Improved local coordinate coding using local tangents. In *International Conference on Machine Learning (ICML)*, 2010.

[50] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *Advances in neural information processing systems (NIPS)*, 2009.

[51] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural computation*, 15(4):915–936, 2003.

[52] Z. Zhang, L. Ladicky, P. Torr, and A. Saffari. Learning anchor planes for classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[53] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems, NIPS*, 2014.

[54] Z. Zuo, G. Wang, B. Shuai, L. Zhao, and Q. Yang. Exemplar based deep discriminative and shareable feature learning for scene image classification. *Pattern Recognition*, 48(10):3004–3015, 2015.