Sparse Coding Trees with Application to Emotion Classification

Hsieh-Chung Chen, Marcus Z. Comiter, H. T. Kung, and Bradley McDanel
Harvard University, Cambridge, MA

Abstract

We present Sparse Coding trees (SC-trees), a sparse coding-based framework for resolving misclassifications arising when multiple classes map to a common set of features. SC-trees are novel supervised classification trees that use node-specific dictionaries and classifiers to direct input based on classification results in the feature space at each node. We have applied SC-trees to emotion classification of facial expressions. This paper uses this application to illustrate concepts of SC-trees and how they can achieve high performance in classification tasks. When used in conjunction with a nonnegativity constraint on the sparse codes and a method to exploit facial symmetry, SC-trees achieve results comparable with or exceeding the state-of-the-art classification performance on a number of realistic and standard datasets.

1. INTRODUCTION

Automated emotion classification from facial expressions is an important component of security, medical, and market-research software [11]. As such, methods to detect users' emotional states through accurate, unobtrusive, and hardware-efficient means are greatly needed. Although individuals all feel the same types of emotions, there exists substantial variation between the ways individuals express these emotions. The importance of subtleties in expressions as well as differences in the subjects' physiology, pose, and environmental conditions make the problem challenging. Addressing these challenges is essential in the construction of an emotion classifier.

For this and other challenging applications, we present Sparse Coding trees (SC-trees), a novel type of supervised classification tree. SC-trees use node-specific dictionaries and classifiers to direct input images to children nodes, whose own dictionaries and classifiers are more specialized, and therefore able to perform more accurate classification. An example SC-tree is shown in Figure 1. SC-trees can be viewed as a general methodology of using tree branching for rectifying a *class conflict*, or the consistent misclassification that occurs when two or more distinct classes are grouped together during classification. By analogy, a class conflict is similar to a hash table collision in which two distinct values are hashed to the same hash value.

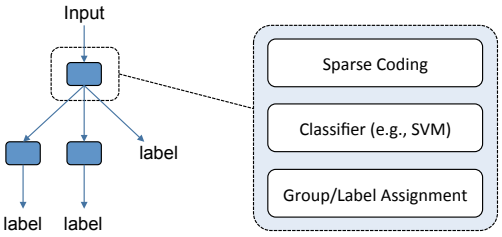

Figure 1. Illustration of an SC-tree. Each node has a node-specific dictionary for use with sparse coding and classifier that together assign a *coarse label* to an input image, and then dispatches the input image to a child node or outputs a *final label*.

SC-trees are a natural framework to incorporate expert knowledge. For example, a model based on expert knowledge may reveal that both the expressions "happiness" and "fear" often share a common feature of displaying teeth (see Figure 5). With this knowledge, an SC-tree may include a branch to a child node that is able to resolve the confusion between these two expressions. Alternatively, the structure of SC-trees can be guided by purely data-driven methodologies, in which coarse classification results form a confusion matrix that is used to derive the SC-tree's branching rules. In Section 6.1 we show an example of how expert-derived models and data-driven models arrive at similar branching rules, suggesting that SC-trees can incorporate expert knowledge both in designing branching rules and validating the branching rules found through purely data-driven methods. This ability to incorporate expert knowledge is especially useful when there is insufficient training data for automatic derivation of these rules.

A node in an SC-tree makes branching decisions by classifying sparse codes of the input in the feature space. As image data with emotion labels is relatively scarce in this domain, we introduce Mirrored Nonnegative Sparse Coding (MNNSC), a novel extension of sparse coding that exploits facial symmetry to improve statistical accuracy of the classifier at each node. MNNSC leverages two key insights:

978-1-4673-6759-2/15/$31.00 ©2015 IEEE

1. *Nonnegativity*: Imposing nonnegativity constraints during sparse coding has been shown to reduce overfitting for a given number of dictionary atoms [14].

2. *Mirroring*: By exploiting facial symmetry, mirroring achieves *reflection invariance*, which is similar to the translation invariance achieved by local maximum pooling often seen in deep learning [13, 35]. This technique allows us to use smaller dictionaries, improving both offline training and online encoding time.

We emphasize MNNSC's usefulness in two important scenarios. First, MNNSC can learn improved feature representations with relatively limited training data. Second, MNNSC allows for the use of greedy encoders. We provide empirical results for both scenarios in Section 5.3.

	SC	SC-Tree
Basic SC	70.1	73.6
NNSC	71.5	76.9
MNNSC	75.1	79.9

Figure 2. Overview of improvements in classification average recall on the baseline CK+ dataset achieved by SC-trees, mirroring, and the nonnegativity constraint. NNSC denotes SC with a nonnegativity constraint, and MNNSC denotes SC with both mirroring and a nonnegativity constraint.

We present extensive experimental evidence validating SC-tress and MNNSC in Section 5, demonstrating results comparable with or exceeding the state of the art. As sparse coding is well known to be effective for classification tasks [18], in this paper we focus on showing our improvements through SC-trees and MNNSC over basic sparse coding. For validating SC-trees, we perform tests on standard baseline datasets (to establish the beneficial properties of SC-trees on well-posed datasets), as well as datasets with greater subject and intra-class emotion variation (to test SC-trees when applied to more freely-posed datasets). Figure 2 previews a summary of results on the CK+ dataset.

Further, given the superior performance of SC-trees with MNNSC, we independently examine MNNSC to validate its discriminative power and explain how it can facilitate accurate classification within SC-tree nodes. We report performance results of MNNSC on a number of emotion-specific real-world datasets, and demonstrate how implementing sparse coding with a nonnegativity constraint and mirroring separately improves classification results, as shown in the rows of Figure 2, ultimately together achieving state-of-the-art results in emotion-specific classification.

2. BACKGROUND

2.1. Related Work in Emotion Classification

There are several common pre-processing steps applied to the raw input images in order to make them more amenable to classification methods: (1) Facial segmentation locates the subject's face in the frame, commonly by using face or eye detectors. (2) Facial registration corrects for rotational variations in the subject, as well as appropriately cropping and scaling the image to a consistent representation [7]. (3) Equalization normalizes the intensity of pixels between images. These preprocessing steps have been shown to impact classification accuracy [33][26], and are used in this paper.

Existing approaches to emotion classification fall into one of two categories based on feature selection: geometry-based or appearance-based. Geometry-based approaches use expert knowledge to identify and model landmarks in the face that discriminate between emotions, such as the shape of the subject's lips. Appearance-based approaches operate directly on pixel data and extract data-driven statistical models of emotions. This latter approach includes methods that compute transformations to a feature space that improves classification accuracy. Our proposed method is appearance-based, and also incorporates expert knowledge in forming the branching structure of the SC-trees in order to address easily confused emotion classes

Various types of feature representations have been used as input to classifiers in this problem domain. Box-filters, which are rectangular image filters, are a well-known technique in the signal processing community [33][20]. Gabor filters are linear filters, and are commonly used for detecting edges in an image [33]. Local binary patterns are another textural feature representation and have been applied to smile detection [28].

Final classification in this domain most-often uses Support Vector Machines (SVMs) due to their accuracy and resilience to overfitting [33]. Additionally, bagging has been shown to increase stability in classification accuracy [8].

The symmetry of the face has been used in other contexts for expression recognition, specifically with detecting asymmetric facial expressions [27]. However, in this context, the symmetry of the face is exploited in order to detect differences in the two sides of the face, rather than to improve feature representations as in our method.

2.2. Related Work in Sparse Coding

Sparse coding is a framework for approximating data using a few prominent exemplar patterns, called features. A dictionary of suitable features is learned from data, for example by unsupervised clustering. By encoding an input signal, such as an image, only in terms of its most prominent features, sparse coding computes a sparse representa-

tion vector that produces more stable statistical models in the presence of noise or missing data, and is well known for its effectiveness in extracting features for classification purposes [18]. One method to find this dictionary (D) is to train it offline from a set of data points x_i in the form of $n \times 1$ vectors[1] using matrix factorization with a sparsity constraint [19]:

$$\min_{D \in E, \alpha_i \in \mathbb{R}^{t \times 1}} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} ||x_i - D\alpha_i||_2^2 + \lambda ||\alpha_i||_1 \quad (1)$$

$$E = \{D \in \mathbb{R}^{n \times t} \mid \forall j, ||d_j||_2^2 = 1\} \quad (2)$$

for a certain $\lambda > 0$, where N is the number of training samples, x_i is an input sample, α_i is the sparse code of x_i, and D is the dictionary of t columns with d_j being the j^{th} column (i.e., atom) in D. Penalizing the $\ell 1$ norm of α_i in this formulation has the effect of inducing *sparse solutions*, or solutions with few non-zero values. As such, when encoded with this dictionary, images are represented as the linear combination of a small number of atoms, many of which represent particular regions of the face such as the teeth, lips, and cheeks. Further, we impose the additional nonnegativity constraint, $\alpha_i \in \mathbb{R}_{\geq 0}^{t \times 1}$ and $D \in \mathbb{R}_{\geq 0}^{n \times t}$ in (1), (4), and (3), which has been shown to reduce over fitting in practice [14]. Figure 3 shows examples of dictionary atoms trained with this formulation.

There are two families of encoding algorithms on which we provide empirical results in this paper: basis pursuit and greedy algorithms, both of which are widely used. The Least Absolute Shrinkage and Selection Operator (LASSO) is a basis pursuit algorithm, which computes the sparse representation α_i of an input patch x_i given a dictionary D under $\ell 1$-minimization [5]:

$$\min_{\alpha_i \in \mathbb{R}^{t \times 1}, D} 0.5 ||x_i - D\alpha_i||_2^2 + \lambda ||\alpha_i||_1 \quad (3)$$

It is known that $\ell 1$-minimization leads to sparse codes and can be robust to irrelevant features [2][23]. For higher computational efficiency, there are greedy algorithms for $\ell 0$-minimization such as Orthogonal Matching Persuit (OMP) [31] and CoSaMP [22], which solves for:

$$\min_{\alpha_i \in \mathbb{R}^{t \times 1}, D} ||x_i - D\alpha_i||_2^2 \text{ s.t. } ||\alpha_i||_0 < K \quad (4)$$

where K is a small constant.

Authors including Lee et al. [12] have shown that nonnegativity constraints, which force image data to be explained as an additive sum of parts, are not only a reasonable model for data, but also produce more accurate classification results. Lin and Kung [14] have shown that, when paired with greedy approximation algorithms for Eq. 4, nonnegative sparse coding produces stable and state-of-the-art results.

Sparse coding is also starting to be applied within the expression recognition community. Mahoor et al. [17] have applied sparse representations to automatic facial action unit recognition. Liu et al. [15] have recently proposed Histograms of Log-Transformed Nonnegative Sparse Coding (HLNNSC) to advance facial expression recognition. However, we have found that for difficult domains such as emotion classification, in which only subtle differences exist between some classes and media can be captured in *extreme* conditions, new approaches (such as the sparse-coding trees introduced in this paper) for resolving classification conflicts are needed in using sparse coding.

Figure 3. Examples of atoms trained with Eq. 1 from the node-specific dictionary for "fear" and "happiness" classes. The atom highlighted in red corresponds to a feature learned from "fear" samples and the one in blue corresponds to a feature learned from "happiness" samples. Similar to results reported by Lee [12] and Hoyer [9], the nonnegativity constraint leads to learned atoms which correspond to local parts.

3. SC-TREES

3.1. Overview

We now present an overview of SC-trees, a novel type of classification tree. SC-trees seek to correct common misclassification patterns in classification problems. We may identify the patterns that need to be corrected either offline by using expert knowledge, online by using data-driven methods, or with a combination of the two methods.

As a contrast, in a basic sparse coding-based classifier, input vectors are coded using a learned dictionary, and the resulting sparse codes are used as input to a classifier such as an SVM. However, for challenging scenarios where input vectors from distinct classes may have the same or similar sparse codes, resulting classification often *consistently* confuses two distinct classes. For example, many images from the "fear" class are consistently labeled as "happiness", as noted earlier.

For classification purposes, such an example constitutes a consistent failure of sparse coding-based classification algorithms. However, because we observe misclassifications during training, we can learn which of the classes are *consistently* confused with one another. As this confusion between distinct classes is consistent, this suggests the use of a new tailored basis (i.e., dictionary that can distinctly encode and discern these classes). Then, all images whose *coarse classification label* is one of the confused classes

[1] For example, 64×32 half-images are unrolled to become 2048×1 vectors.

can be subsequently encoded with this new dictionary, and then properly classified. This is the key insight that SC-trees leverage.

3.2. Building an SC-tree

SC-trees are built one node at a time from the top down. There are three steps in building each node: learning a node-specific dictionary for use in sparse coding, learning a node-specific classifier such as an SVM, and learning branching rules from this node to children nodes and leaves (which provide final classifications). We now detail each step.

1. *Dictionary Learning*: A dictionary is learned as described in Eq. 1. The input to this step is a set of unlabeled vectors.

2. *Classifier Learning*: The second step trains a node-specific classifier using the sparse codes found from encoding input with the dictionary learned in the previous step. We achieve best results for the task of classifying facial expressions using MNNSC (Section 4), but in general a number of sparse coding algorithms (including kernelized sparse coding [29]) can be used in encoding. In this paper, we use a multi-class linear SVM.

3. *Branching Rules Derivation*: The final step is determining the branching rules from the node. Groups consisting of two or more commonly confused classes are used to branch from the current node to a new child node trained specifically to differentiate classes within the group. Groups that consist of a single class simply output a final classification label.

   As noted earlier, these groups are determined using expert knowledge, data-driven methods, or a combination of the two approaches. For data driven methods, we first obtain a confusion matrix C using the node's dictionary and classifier. Once we have calculated C, we can find groups of "conflicting classes" that are hard to differentiate. To find these groups, we can apply standard clustering methods such as spectral clustering to an inter-class affinity matrix $A = \frac{1}{2}(C + C^T)$ [24]. See Figure 4 for an example of the data-driven branching method.

   Alternatively, branching rules can be directly specified based on expert knowledge such as by branching on classes that are known to be qualitatively similar. We demonstrate building an SC-tree using expert knowledge in Section 6.1.

3.3. Using an SC-tree

Once the SC-tree has been trained, it can be used for classification. Given an input vector, the SC-tree performs the following actions to provide a classification label: First, initialize the root node as the active node a. Second, encode the input vector into a sparse code using the active node a's dictionary, and then obtain a *coarse classification label* of the encoded input using a's classifier. Third, based on the *coarse classification label*, follow the branching rules to either branch to a child node, or output a label and terminate. Fourth, if branched to a child node c_i, the c_i becomes the active node a, and return to the second step.

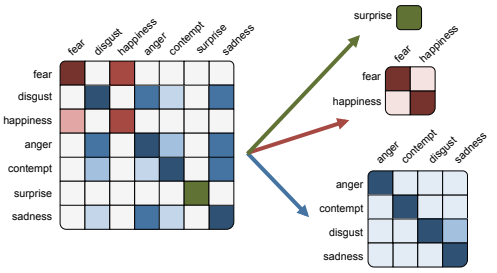

Figure 4. For the CK+ dataset, the root node of the SC tree is a coarse seven-way emotion classifier. This produces a 7×7 confusion matrix (shown above to the left), where darker shades correspond to higher counts. We apply spectral clustering to this and find 3 groups of classes (shown above to the right) that determine the structure of the tree. In this example, samples classified as "surprise" by the coarse classifier are labeled as "surprise" as final output (as it is the only class in the group). Samples classified as "happiness" or "fear" are directed to one child node, and the remaining samples are directed to a second child node.

3.4. Resolving Misclassification in Practice

The essence of SC-trees is their ability to correct for consistent misclassification between groups of classes. We now turn our attention to an example of this misclassification in practice, and show how SC-trees resolve the following problem: in our experiments, we find that fear is often confused with happiness when using MNNSC without SC-trees for classification. Figure 5 shows examples of images from the "fear" class that are misclassified as "happiness". Interestingly, we see the subjects in each of these images display their teeth, a trait that these images share with image examples in the "happiness" class. However, in the "fear" class, the subjects display teeth in a manner such that the corners of the lips are not turned up as in a happiness, but rather are stretched out. This small but significant difference is a main discriminating feature between the two classes.

We can get greater insight into this phenomenon by examining the difference between a general dictionary trained without SC-trees and an SC-tree child node's dictionary trained specifically for differentiating fear and happiness, shown in Figure 3. Unlike the general dictionary, the child node's dictionary captures *both variations* of displaying teeth: displaying teeth with the corners of the lips turned

Figure 5. These images, taken from the "fear" class, are examples of images that are misclassified as "happiness" in using a MNNSC classification pipeline without SC-trees. Note that the subjects in each of these images display their teeth. SC-trees, branching on a ("happiness", "fear") grouping, corrects these misclassifications.

upwards in "happiness", and displaying teeth with the corners of the mouth stretched outwards in "fear". We note that the introduction of SC-trees increases the overall recall rate, and more specifically, increases the recall rate for the "fear" class substantially (38%), correcting all but one of the misclassifications of "fear" as "happiness".

4. Mirrored Nonnegative Sparse Coding

We now present an overview of MNNSC, a sparse coding algorithm augmented with nonnegativity and a mirroring procedure. In Section 5, we show that SC-trees that use MNNSC at each node for the sparse coding step achieve best results on all datasets. Note that MNNSC uses a dictionary trained with nonnegative matrix factorization.

4.1. Nonnegativity Constraint

A danger in sparse coding is overfitting to the input signal. This can hurt classification by making the encoding sensitive to small variations or noise in the signal. Rather than directly enforcing a hard sparsity constraint, nonnegativity allows sparse coding to have an automatic stopping point in searching for nonzero coefficients. This deterministic algorithm provides a fundamental solution to overfitting. For example, Figure 6 shows the difference between unconstrained and nonnegative sparse coding for our application domain. With unconstrained sparse coding, as sparseness decreases past its optimal point, we see a decrease in performance, a consequence of overfitting to the signal. However, with the nonnegativity constraint, the algorithms are forced to terminate when no more atoms have positive correlation with the signal, therefore avoiding overfitting and leading to higher performance. We denote sparse coding with this nonnegativity constraint as NNSC.

4.2. Mirroring

Our mirroring procedure splits the input image vertically in half and reflects the right patch such that both patches have the same orientation (i.e., both resemble the right side of the face). Then, both patches are encoded using a learned dictionary. Following encoding, the two patches are aggregated via a maximum pooling operation in the feature space, as illustrated in Figure 7. We denote sparse coding

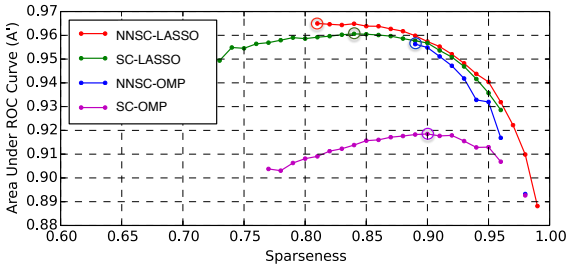

Figure 6. The nonnegativity constraint allows code with lower sparseness to be used without overfitting to the signal. We use *sparseness* as defined by Hoyer [9]. The circled points denote the optimal sparseness for each encoding method. The unconstrained variants of OMP and LASSO decrease in performance past the optimal sparseness. This figure uses the GENKI-4K dataset.

with a nonnegativity constraint and mirroring procedure as MNNSC.

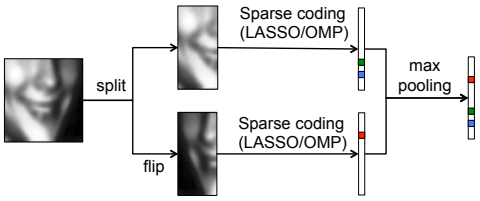

Figure 7. Illustration of the benefit of our mirroring procedure. Images resembling reflections of one another will have similar feature representations — we call this *reflection invariance*. The colored blocks indicate active elements in the sparse codes.

This mirroring procedure allows the sparse codes to reflect the features with the stronger response from either side of the face. This often results in more robust representation for facial images that are not well posed. As shown in Figure 8, the feature vectors following mirroring are mainly characterized by the side of face with the clearer view.

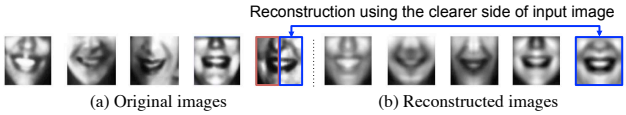

Figure 8. Examples of original and reconstructed images under mirroring. The input images (on the left side) are split in half vertically (indicated by the red and blue box), encoded, and then maximum pooling is applied to the sparse codes. As such, note that each reconstructed image (on the right side) is determined by the side of face with the stronger response (corresponding to the blue box).

5. EXPERIMENTS

In this section, we empirically validate SC-trees. We provide results on two popular datasets, and show that SC-trees provide a substantial increase in average recall over sparse coding algorithms, achieving best results when used in conjunction with MNNSC. We also provide results of MNNSC in isolation on two additional datasets. Note that we need to validate MNNSC in isolation so we can evaluate its effectiveness as part of the directing mechanism used in SC-trees. As the search space is relatively small, all parameters for dictionary and classifier learning are found using grid-search [30].

5.1. Datasets

We validate the SC-tree classifier on both the Cohn-Kanade Extended dataset (CK+) [16] and the Emotions in the Wild dataset (EitW) [4][3]. The CK+ dataset contains labeled video sequences of 118 subjects taken in a controlled, laboratory environment. Each subject is recorded in a series of still images captured in controlled conditions progressing from a neutral to one of seven emotions (anger, contempt, disgust, fear, happiness, sadness, and surprise). We extract the last frame of each sequence for classifier training and testing (a total of 373 images). The EitW dataset contains short video and audio segments of subjects displaying emotions in movies, and the same emotions as the CK+ dataset, excluding contempt. We extract the middle frame of each sequence for classifier training and testing (a total of 300 images) and do not use the audio component.

Additionally, we further validate MNNSC on two emotion-specific datasets: GENKI-4K [32][10] and Affective-MIT Facial Expression Dataset (AM-FED) [21]. The GENKI-4K dataset is the publicly available subset of the larger GENKI dataset [34], and consists of real-world images downloaded from the Internet labeled either "happiness" or "neutral". The AM-FED dataset consists of images of subjects taken with a commodity webcam in real-world environments. As these datasets contain only a single labeled emotion ("happiness") and neutral frames, we do not use SC-trees.

For all datasets, the images are preprocessed by cropping the region of interest (ROI) around the mouth, which is found using OpenCV's Haar-feature based cascading classifier [1], scaling the ROI to 64 by 64 pixels, converting to gray scale, and normalizing such that each image has a unit norm. For the EitW dataset, the same preprocessing is employed except that the pre-cropped images of the entire face included with the dataset and obtained by the method used in [3] were used as input. Note that while using these pre-cropped frames ignores difficulties in adjusting for pose and other variations present in this uncontrolled environment, we choose to use these precise frames as input to our pipeline in order better focus and understand the benefits of SC-trees in isolation, as attempting to simultaneously address this other difficulty would complicate analysis and understanding of our results.

		Dataset	
		CK+	EitW
Method	SC-tree MNNSC	79.9	33.0
	MNNSC	75.1	29.4
	SC-tree NNSC	76.8	29.7
	NNSC	71.5	28.1
	SC-tree Basic SC	73.6	28.6
	Basic SC	70.1	26.5

Table 1. Performance results (average of per class recall) on the CK+ and Emotions in the Wild (EitW) dataset demonstrating the increase in average recall achieved by SC-Trees with a number of sparse coding algorithms (all using LASSO). Note that for both datasets, the best results are achieved with SC-trees used in conjunction with MNNSC.

5.2. SC-tree Performance Results

Table 1 shows the increase in average recall achieved through the use of SC-trees with three different sparse coding algorithms: MNNSC, nonnegative sparse coding (NNSC), and K-SVD (Basic SC). These results are calculated by taking the average of each class' recall. This measure is appropriate, as the number of images in each class differs by a large margin. The structure of the SC-tree is derived using data-driven methods, arriving at the structure shown in Figure 4.

For the purposes of generating results on these two datasets, we employ "leave-one-subject-out" cross-validation for training/testing [16]: for a given human subject, we take all images of that subject as the testing set, and use the remaining images of all other subjects as the training set, repeating this procedure for all subjects in the dataset.

On the CK+ dataset, SC-trees provide a 2.5-5% increase in average recall as compared to using each sparse coding algorithm without the SC-trees. Of the three sparse coding methods, MNNSC gives the best performance. We achieve comparable results to other data driven methods [25], but cannot compare directly as they report the accuracy over *all frames* rather than the average of per class recall as we do. However, we note that there is a large discrepancy between class sizes (the distribution is 45, 17, 55, 25, 69, 28, 82 for "anger", "contempt", "disgust", "fear", "happiness", "sadness", and "surprise", respectively). As such, under the metric used in [25], an increase in the accuracy of, for example, surprise at the expense of contempt could lead to a substantial increase in the overall score, as surprise has many more examples than contempt. Further, because "surprise" does not generally get confused with the other emotions (as noted in Figure 4) taking the accuracy across all frames skews the overall score even further. In our metric, we do not have this issue as we report the average per-class

		Dataset	
		GENKI-4K	AM-FED
Method	MNNSC-LASSO	97.0	92.3
	NNSC-LASSO	96.7	91.2
	SC-LASSO	95.1	89.7
	MNNSC-OMP	96.2	92.1
	NNSC-OMP	95.7	88.8
	SC-OMP	93.1	86.0
	Gabor	95.7	88.9
	Reported*	96.1 [33]	90.0 [21]

*Best results reported in prior literature

Table 2. Performance results in area under the ROC curve (A') comparing LASSO and OMP sparse coding methods on datasets containing "happiness" and "neutral". Note that MNNSC-LASSO performs the best on both datasets, with MNNSC-OMP performing comparably.

recall.

A similar pattern of results is seen testing on the Emotions in the Wild dataset. We again see that the best results are achieved with MNNSC, and that SC-trees provide the largest boost in average recall (3.6% increase) when MNNSC is used. We note that [3] reports baseline classification accuracy of 27.2% on the Emotions in the Wild dataset using the videos as input. While we can not make a direct comparison between our results and these baseline numbers (as we do not include neutral frames in our classification problem, nor do we use the full videos), these baseline numbers demonstrate the difficulty of this dataset.

5.3. MNNSC-LASSO and MNNSC-OMP Performance Results

Table 2 shows the area under the ROC curve (A') of MNNSC compared to other sparse coding-based approaches using a number of settings, as well as the results of Gabor filters and results reported in the literature, demonstrating that data-driven sparse coding approaches outperform those based on hand-designed features. For both datasets, MNNSC using nonnegative LASSO (MNNSC-LASSO), gives the best performance. In generating results for GENKI-4K and AM-FED datasets, an 80/20 train and test split with cross-validation is used.

We have done our best to reproduce the results of other work and achieved results comparable to those reported in the literature for each dataset. We have implemented our own version of Gabor encoding, which uses a filter bank of 40 filters at different frequencies and orientations. These parameters were selected from the previous state-of-the-art result for this problem [33], and we followed the same methodology outlined in Section III.c of that paper.

Mirroring substantially improved performance for both LASSO (MNNSC-LASSO) and OMP (MNNSC-OMP) on the AM-FED dataset. Recall that this dataset is composed of images captured in less-constrained conditions, and contains large variations in lighting conditions and poses. By exploiting symmetry, we can better handle these difficult scenarios provided at least one of the mirrored sides has the information we wish to detect.

Overall, LASSO gives better results than OMP, and outperforms OMP at all ranges of training data. This is reflected in Figure 9 for the GENKI-4K dataset. Similar trends are observed for the AM-FED dataset as well. However, Figure 9 also demonstrates that as the amount of training data increases, the gap between MNNSC-OMP and MNNSC-LASSO decreases substantially. Therefore, MNNSC-OMP, given enough training data, can have comparable performance with MNNSC-LASSO. This observation can motivate the choice of encoder, given MNNSC-OMP's significantly lower running time relative to MNNSC-LASSO. We discuss these efficiency gains further in Section 6.2.

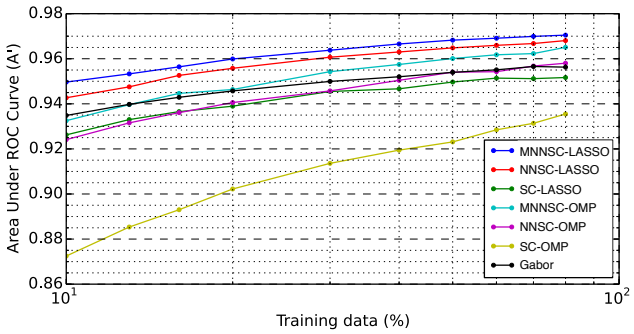

Figure 9. Performance evaluation of LASSO and OMP with respect to amount of training data (as a percent of the GENKI-4K dataset). The x-axis is plotted in log-scale.

6. ANALYSIS AND DISCUSSION

6.1. SC-tree Branching Rules Based on Expert Knowledge

Given the increasing amount of data available in the domain, our methods in this paper have been primarily data-driven. However, fundamentally the definition of emotions themselves are built upon models set forth by domain experts [6]. As such, in this section, we show that an SC-tree's branching rules can also be derived using expert domain knowledge, and then discuss the benefits of utilizing both data-driven and expert knowledge-based SC-trees in conjunction with one another.

In designing an SC-tree using expert knowledge, we look to the Facial Action Coding System, or FACS [6]. FACS codes facial expressions using combinations of Action Units (AUs), or small, localized muscle movements. An example AU is "cheek raiser", in which the cheek muscles are tensed and move upwards. Using combinations of

AUs, any facial expression can be encoded by hand.

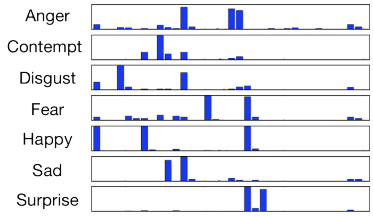

Figure 10. Histograms displaying the frequency of AU use within each emotion class for the CK+ data. The correlation between the frequency vectors represented by these histograms are used to incorporate the expert knowledge of Action Units and the FACS system in forming branching rules for the SC-trees.

To design the branching structure of the expert knowledge-based SC-tree, we utilize the correlation of AUs used in encoding different emotions, as coded by trained FACS coders. To do this, we use the CK+ dataset, which contains AU codings for each emotion sequence in the dataset. Using the expert knowledge embedded in the AU coding system, we find the frequency of AU usage among different classes of emotions, as shown in Figure 10. Using correlations between the frequency of use of AUs between classes, we discover groups very similar to those found through data driven methods. Namely, the groups (g_1:{"happiness", "fear", "surprise"}, g_2:{"anger", "contempt", "disgust","sadness"}), are the same as those found by data driven methods, shown in Figure 4, except that "surprise" is now grouped with "happiness" and "fear".

This underscores an important characteristic of SC-trees: even in a data-driven framework, there are additional benefits from incorporating model information to compliment the data-driven SC-tree. First, the model-based SC-tree structure can be used to inform and refine the structure of the data-driven SC-tree. Second, training a model-based SC-tree in addition to the data-driven SC-tree allows us to cross-validate each tree with one another. This can be particularly helpful in controlling for over-fitting when deriving the tree structure from the confusion matrix obtained through data-driven methods, as discussed in Section 3.2.

6.2. Efficiency

We now examine the efficiency of the methods presented in this paper. Two main considerations in this regard are:

1. *Dictionary Size*: A smaller dictionary reduces encoding cost and more easily fits within memory, an important consideration in memory-constrained devices.

2. *Choice of Encoding Algorithm*: The choice of encoding algorithm represents a trade-off between accuracy and computational efficiency. For example, greedy matching pursuit algorithms such as OMP are attractive for scenarios where efficiency is important. However, these greedy algorithms only find local minima, and often produce worse results for classification tasks when compared to computationally more intensive methods such as LASSO.

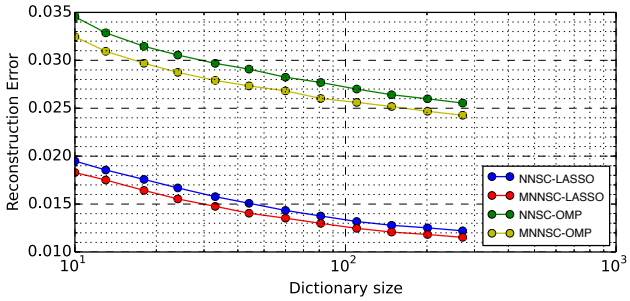

Figure 11. Mirroring allows the dictionary to be used more efficiently, and therefore has lower reconstruction error compared to the non-mirrored counterparts.

The two components of MNNSC (the mirroring procedure and the nonnegativity constraint) address these considerations. Mirroring enables more "efficient" dictionaries, as high level features on either side of the face can be represented by the same atom. Therefore, a smaller dictionary (almost half as many atoms) can be used to achieve the same level of reconstruction error when compared to the counterpart without mirroring, as shown in Figure 11. In addressing the efficiency of encoding algorithms, we find that the addition of the nonnegativity constraint greatly reduces the performance gap between OMP and LASSO given sufficient training data, as seen in Figure 9. Therefore, given enough training data, computationally-efficient encoding methods such as OMP may be used in favor of computationally expensive methods such as LASSO. We note that these considerations may have implications when using our pipeline on computationally- and power-constrained mobile devices.

7. CONCLUSION

In this paper, we present SC-trees, a novel type of classification tree that splits based on node-specific dictionaries and classifiers. Although SC-trees can use a number of different sparse coding algorithms, we find that for classifying facial expressions, best results are achieved with Mirrored Nonnegative Sparse Coding (MNNSC), a novel sparse coding algorithm that supplements a traditional sparse coding pipeline with a *nonnegativity constraint* and *mirroring* in order to increase the average recall of the method. Further, SC-trees are a natural avenue through which to incorporate expert knowledge and models. We apply SC-trees to the emotion classification problem, and validate SC-trees on standard community benchmark datasets. and achieving results comparable with or exceeding the state of the art.

8. ACKNOLEDGEMENTS

This research was supported in part by gifts from the Intel Corporation. The authors would like to thank Stephen Tarsa for his thoughtful suggestions in improving the paper.

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 6

[2] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006. 3

[3] Abhinav Dhall, Roland Goecke, Jyoti Joshi, Karan Sikka, and Tom Gedeon. Emotion recognition in the wild challenge 2014: Baseline, data and protocol. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 461–466. ACM, 2014. 6, 7

[4] Abhinav Dhall, Roland Goecke, Simon Lucey, and Tom Gedeon. Collecting large, richly annotated facial-expression databases from movies. *IEEE MultiMedia*, (3):34–41, 2012. 6

[5] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004. 3

[6] Paul Ekman and Wallace V Friesen. Facial action coding system. 1977. 7

[7] Ian Fasel, Bret Fortenberry, and Javier Movellan. A generative framework for real time object detection and classification. *Computer Vision and Image Understanding*, 98(1):182–210, 2005. 2

[8] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009. 2

[9] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004. 3, 5

[10] The MPLab GENKI-4K Dataset. http://mplab.ucsd.edu. 6

[11] Raffi Khatchadourian. We know how you feel, January 2015. 1

[12] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. 3

[13] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009. 2

[14] Tsung-Han Lin and H. T. Kung. Stable and efficient representation learning with nonnegativity constraints. *Proceedings of the 31st International Conference on Machine Learning*, 32, 2014. 2, 3

[15] Ping Liu, Shizhong Han, and Yan Tong. Improving facial expression analysis using histograms of log-transformed nonnegative sparse representation with a spatial pyramid structure. In *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, pages 1–7. IEEE, 2013. 3

[16] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 94–101. IEEE, 2010. 6

[17] Mohammad H Mahoor, Mu Zhou, Kevin L Veon, Seyed Mohammad Mavadati, and Jeffrey F Cohn. Facial action unit recognition with sparse representation. In *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pages 336–342. IEEE, 2011. 3

[18] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689–696. ACM, 2009. 2, 3

[19] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010. 3

[20] MJ McDonnell. Box-filtering techniques. *Computer Graphics and Image Processing*, 17(1):65–70, 1981. 2

[21] Daniel McDuff, Rana El Kaliouby, Thibaud Senechal, May Amr, Jeffrey F Cohn, and Rosalind Picard. Affectiva-mit facial expression dataset (am-fed): Naturalistic and spontaneous facial expressions collected" in-the-wild". In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pages 881–888. IEEE, 2013. 6, 7

[22] Deanna Needell and Joel A Tropp. Cosamp: iterative signal recovery from incomplete and inaccurate samples. *Communications of the ACM*, 53(12):93–100, 2010. 3

[23] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004. 3

[24] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002. 4

[25] Raymond Ptucha and Andreas Savakis. Lge-ksvd: Flexible dictionary learning for optimized sparse representation classification. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pages 854–861. IEEE, 2013. 6

[26] Elias Rentzeperis, Andreas Stergiou, Aristodemos Pnevmatikakis, and Lazaros Polymenakos. Impact of face registration errors on recognition. In *Artificial Intelligence Applications and Innovations*, pages 187–194. Springer, 2006. 2

[27] Thibaud Sénéchal, Jay Turcot, and Rana El Kaliouby. Smile or smirk? automatic detection of spontaneous asymmetric smiles to understand viewer experience. In *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, pages 1–8. IEEE, 2013. 2

[28] Caifeng Shan. Smile detection by boosting pixel differences. *Image Processing, IEEE Transactions on*, 21(1):431–436, 2012. 2

[29] Sareh Shirazi, Mehrtash T Harandi, Brian C Lovell, and Conrad Sanderson. Object tracking via non-euclidean geometry: A grassmann approach. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 901–908. IEEE, 2014. 4

[30] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. Technical report, 2012. 6

[31] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666, 2007. 3

[32] http://mplab.ucsd.edu. The MPLab GENKI Database, GENKI-4K Subset. 6

[33] Jacob Whitehill, Gwen Littlewort, Ian Fasel, Marian Bartlett, and Javier Movellan. Developing a practical smile detector. In *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, 2008. 2, 7

[34] Jacob Whitehill, Gwen Littlewort, Ian Fasel, Marian Bartlett, and Javier Movellan. Toward practical smile detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(11):2106–2111, 2009. 6

[35] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE, 2009. 2