Exemplar Codes for Facial Attributes and Tattoo Recognition

[123]Michael J. Wilber, [12]Ethan Rudd, [1]Brian Heflin, [1]Yui-Man Lui, and [12]Terrance E. Boult
[1]: Securics, Inc [2]: University of Colorado, Colorado Springs [3]: Cornell University
{mwilber,erudd,bheflin,ylui,tboult}@securics.com

Abstract

When implementing real-world computer vision systems, researchers can use mid-level representations as a tool to adjust the trade-off between accuracy and efficiency. Unfortunately, existing mid-level representations that improve accuracy tend to decrease efficiency, or are specifically tailored to work well within one pipeline or vision problem at the exclusion of others. We introduce a novel, efficient mid-level representation that improves classification efficiency without sacrificing accuracy. Our Exemplar Codes *are based on linear classifiers and probability normalization from extreme value theory. We apply Exemplar Codes to two problems: facial attribute extraction and tattoo classification. In these settings, our Exemplar Codes are competitive with the state of the art and offer efficiency benefits, making it possible to achieve high accuracy even on commodity hardware with a low computational budget.*

1 Introduction

As computer vision researchers, we have a plethora of algorithms and pipelines available to solve many real-world problems from face detection to object recognition. And yet, those who actually implement these state-of-the-art systems on conventional hardware are faced with a problem: How can we deal with the inherent trade-off between efficiency and accuracy? Sometimes, we are genuinely stuck between a rock and a hard place: Running a state-of-the-art deep learning neural network pipeline on a mobile device or even a modern desktop simply isn't feasible yet, but a much simpler Histograms of Oriented Gradients (HOG) pipeline taped to a linear classifier isn't terribly impressive with regards to accuracy. From a pure applications perspective, if we do not wish to wait for tomorrow's hardware to catch up so it can run yesterday's state-of-the-art algorithms, we must find better ways of improving accuracy using what limited computational power we have today. This paper represents our ongoing efforts to create an efficient yet accurate mid-level representation that can help solve a variety of real-world computer vision tasks. Our aim is to build a lightweight, *efficient* classifier that can approximate the performance of other complicated, *high-accuracy* classifiers while being *flexible* enough to work well within many existing vision pipelines.

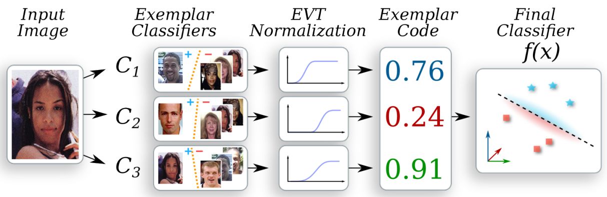

Figure 1. Overview of our Exemplar Codes pipeline. Each exemplar classifier is trained with one exemplar as positives and data from other labels as negatives. The EVT-normalized score for each exemplar becomes one element of the feature vector for the final classifier. Exemplar Codes are efficient, only requiring one dot product and one CDF evaluation per dimension.

How can we do this? Rather than choosing between the "better classifier" and "cheaper classifier," we argue that computer vision researchers can use mid-level representations to manipulate the performance $\leftrightarrow$ efficiency curve. Usually, adding another layer into the middle of a system with the aim of improving accuracy typically makes the system slower. Consider the "multitude of complex, nonlinear, multi-layer neuromorphic feature representations" proposed in [5], for example. Such features improve accuracy at the cost of increased processing time since they send the image through several layers of processing to calculate each feature dimension. Similarly, POOF features [24] require an affine transformation, a feature extraction step, and a classification decision for each dimension of the feature vector. On the other side of the spectrum, PiCoDeS [2] focus on efficiency and low memory use and are much simpler than the above methods, but since they quantize each dimension to a single bit to save time and memory, this method discards useful information.

We aim to address the gap left by existing representations. Here, we present a novel mid-level representation that approximates the accuracy of complicated radial basis function (RBF) classifiers while achieving the efficiency of linear classifiers. To this end, we introduce the *Exemplar Code* representation. Each dimension of an Exemplar Code is the decision score returned from a simple linear classifier re-normalized to a probability estimate via extreme value theory (EVT), which corresponds to the match probability between the sample and an individual training exemplar.

These Exemplar Codes serve as feature vectors for a linear/nonlinear second-stage classifier. This formulation is efficient: both training and classification complexities for Exemplar Code classifiers scale linearly with the number of training exemplars. Moreover, training an Exemplar Code classifier is an embarrassingly parallel task suited to efficient implementation on contemporary and future parallel architectures. Our pipeline is simple and generalizable to a wide variety of applications beyond those mentioned here. Our contributions are threefold:

- We introduce **Exemplar Codes**, a novel mid-level representation for images based on simple linear classifiers and a probability normalization technique from extreme value theory.
- We use exemplar codes to create a **face attribute extraction system**, compare the accuracy and efficiency of our work to existing state-of-the-art, and outline possible changes to improve efficiency without compromising accuracy. We demonstrate that our Exemplar Code system is at least as accurate as a heavyweight RBF classifier, but is much more flexible and efficient.
- We implement a state of the art pipeline for **unconstrained tattoo classification.** We demonstrate this pipeline works well on a challenging dataset of tattoo images.

2 Background and Related Work

Our Exemplar Code classifiers are similar in nature to three distinct branches of research. Classification by an ensemble of exemplar SVMs was originally pioneered for the problem of object detection. Malisiewicz *et al.* performed object detection using a sliding window approach in conjunction with hard-negative mining [16]. Whenever an exemplar decision exceeds a threshold, this approach assigns a category to that window. Our work is intended for classification, where we consider the full image rather than a single sliding window. Under this formulation, it makes more sense to intelligently combine the scores from many exemplars rather than simply using the label of the maximum Platt-calibrated score.

Bergamo *et al.*, by contrast, formulated SVM ensemble image descriptors for the purpose of novel category recognition by second-stage classifiers [2, 3]. Their approach learns basis classes, and uses the normalized decision scores returned from these classifiers as inputs to a second-stage SVM. This work differs from our approach in two ways: First, Bergamo *et al.* define several classifiers by explicitly optimizing for final-stage linear classifier performance while ours is applicable to any final-stage classifier. Second, to save memory, they quantize each classifier's output to a single bit which loses a great deal of information while we leave our Exemplar Codes as floating-point values. Consequently, the approach in [2] does not synthesize information corresponding to single training instances, and the descriptor discards valuable information that may be useful to the final stage.

Both Malisiewicz et al. and Bergamo et al. used Platt scaling to normalize distances from the decision boundaries of raw classifiers into signed probability estimates for their classifier ensembles. Platt normalization aims to improve classification by using a sigmoid fit to estimate probabilities. The sigmoid assumption behind Platt normalization, however, is based on empirical observations of tails of score distributions near SVM decision boundaries [19] with no firm statistical grounding. Our approach uses a W-score normalization, which unlike Platt normalization, is statistically grounded in extreme value theory (EVT) [20, 21] and thus is a more principled model for our system. We discuss this more in Sec. 3.2.

Representing the decision function as some comparison between the query samples and a held-out library of exemplars is a common way to perform face recognition/verification and several algorithms are based on this idea [25, 22, 26]. We emphasize that while we are not interested specifically in face recognition, all of the following systems provide interesting examples of mid-level representations that improve accuracy at some extra computational expense.

One-shot similarity [25], for example, verifies whether query images a and b have the same identity by training two classifiers at test time, each one with its query image as the lone positive and the entire library as negatives. The decision score for the pair is the average of the scores of the two classifiers. Decision scores for each descriptor are lifted into a multidimensional feature representation by sampling many random subsets of the library. Since several classifiers must be trained at test time, this method is not suitable for real-time applications. Our approach does not require learning classifiers at test time, and instead of taking the average of two sample classifiers, our final stage decision can take advantage of as many pre-learned "sample-vs-library comparison" classifiers as desired.

An approach based on *ranked lists of doppelgängers* [22] decides whether two faces have the same identity by computing similarity between a library of exemplars. They apply a threshold to the correlation coefficient between the ranked identity lists. Because their library contained many images of varying pose and lighting, comparing ranks of identities helps the classifier attain pose and illumination invariance. However, this relative ranking destroys the absolute measure of how similar objects are; for instance, this method cannot distinguish between samples that are very similar to many library exemplars from samples that are similar to only one or two—information that may be useful for the final stage decision. Further, this approach requires libraries that contain a wide variety of poses and lighting for each identity.

The *associate-predict model* of [26] associates the query images with images having similar pose and lighting within a set of multi-view identities. Then, only the similarly-posed images are compared to determine a verification result. Our method does not rely on an assumed association result and can compare using more samples than just the best-posed match within the exemplar database, and it does not require a held-out set of differently posed images.

2.1 Classification of Visual Facial Attributes

Visual attributes are commonly used in face recognition/verification for two reasons: Attributes carry semantic meanings which are useful to humans and they are open-set in nature, which means they can capture information about unknown persons of interest even if the person is not in the gallery.

In [13] and [12], Kumar et al. developed "AFS," a state-of-the-art visual attribute classification approach in which an RBF kernel SVM is trained for each attribute. The feature vector corresponding to a given attribute consists of a learned set of feature types extracted from a learned set of functional regions of the face, giving each attribute classifier the optimal attributes from the optimal regions of the face. This approach serves as the baseline for our comparison in Sec. 4.1. However, the final classification step is demanding in terms of speed and memory use and thus is not feasible for latency-sensitive real time applications which rely on facial attribute classification.

2.2 Unconstrained Tattoo Recognition

Recognition of scars, marks, and tattoos has gained attention principally from the intelligence, law enforcement, and forensic communities as a means of recognizing persons of interest when hard biometrics, such as face or fingerprint samples, are not available or are not sufficiently discriminative. Several approaches to tattoo classification to date have offered promising results, but many are restricted in scope to closed-set, pre-cropped images [10, 14, 1, 11, 17, 15, 7], and are therefore predicated on the assumption that a tattoo is present in the probe image in question. For large scale content-based image retrieval (CBIR) applications, this assumption is often invalid. In [8], Heflin *et al.* approached the open-set problem by first using a GrabCut algorithm for image segmentation, then classifying SIFT feature matches via different one-class SVMs trained on positive sample images of scars, marks, and tattoos of various types. However, for law-enforcement systems, one important functional requirement is the ability to efficiently update classifiers. Unfortunately, one-class SVM training time complexities make this task expensive as the database size grows.

The flexibility inherent to Exemplar Code classifiers makes them promising candidates for the problem of tattoo classification. Since Exemplar Codes are comprised of ensembles of linear classifiers, their effectiveness does not depend on being able to fully characterize the geometry of tattoo classes in feature space. This is a challenging task for all classification systems due to the wide variability among tattoos of the same ground truth label; see Fig. 2 for examples. Instead, the effectiveness of Exemplar Codes depends merely on having similar tattoo exemplars in the training set. This way, the margin that separates tattoos of interest from tattoos of other classes becomes better characterized as the number of training exemplars grows. If the gallery contains several images of skull tattoos that are sufficiently similar to a given probe skull tattoo image, for example, Exemplar Code classifiers may register a match even though the precise geometry of the skull tattoo class boundary remains unknown. Further, since each Exemplar Code is the output of several linear classifiers, adding a new exemplar is as simple as training a new linear classifier. Consequently, gallery updates can be performed more often and with larger galleries than kernel based systems can feasibly accommodate. Finally, an additional advantage of using Exemplar Code classifiers for unconstrained tattoo classification is that efficient detection and cropping can be performed via exemplar ensembles [16]. This means that redundancies between object detection and classification can be largely eliminated unlike in [8].

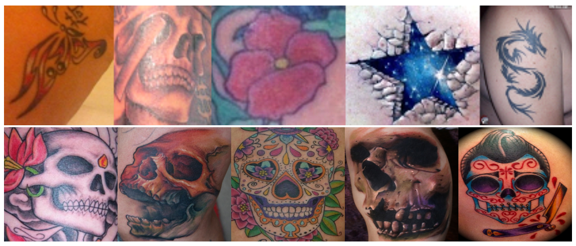
Figure 2. Samples of segmented tattoo images from each class (top), and samples showing the variation in the skull class (bottom). Intra-class variation often exceeds inter-class variation, complicating recognition.

3 Formulation

In this section, we describe our Exemplar Codes approach in detail. As discussed earlier, Exemplar Codes are composed of EVT-normalized scores from linear exemplar classifiers, but there are many steps to this process.

Rather than directly train a label classifier to each sample, we represent each sample as a measure of several similarities against a possibly separate library of *exemplar feature vectors*. To create a similarity comparison, we independently train several exemplar classifiers that each discriminate between that exemplar and sample vectors from other labels. Thus, if an exemplar classifier outputs a high score, then the test sample stands out from the negative data in a similar way that the exemplar itself does. It is important to note that we do not train exemplar classifiers with *all* other data as negatives. If we did, we run the risk of overfitting because only the very closest samples to each exemplar would become support vectors; each may induce a large tilt on the hyper-plane if they are extremely close, making the fitting unstable. Instead, we use only the differently-labeled samples as negatives.

Each exemplar classifier is a simple linear classifier for efficiency and because linear kernels are less likely to overfit than a neural network or RBF SVM. Though a linear classifier arguably is not very discriminative, each one does not need the complicated decision boundary required by the overall category classifier; rather, we only need a simple measure of similarity between the sample and its associated exemplar. In this sense, we wish to define only the negatives in a parametric way; the positive classifiers need only establish non-parametric affinity because the negative data influences the decision boundary the most [16].

3.1 Learning Exemplars

We consider the problem of creating a classifier that distinguishes between two or more application-specific labels. This might be an attribute classifier that distinguishes MALE from FEMALE labels or a tattoo classifier that distinguishes DOG from FLOWER from DRAGON tattoo labels. As input, we use two (possibly identical) training sets: a library of n exemplars $E = \{e_1, \ldots, e_n\}$ with corresponding labels $L_e = \{l_{e,1}, \ldots, l_{e,n}\}$ and a set of m training samples $T = \{t_1, \ldots, t_m\}$ with corresponding labels $L_t = \{l_{t,1}, \ldots, l_{t,m}\}$. Each exemplar and each training sample is assumed to be represented as a $1 \times d$-dimensional feature vector. Note that exemplars may be drawn from the training set where $m = n$, $E = T$, and $L_e = L_t$, or they may be drawn from a separate set of held-out examples. If $|E|$ is large, exemplars can be chosen using the method in Sec. 3.4.

To learn Exemplar Codes, we learn n linear classifiers $C_i : \mathbb{R}^d \to \mathbb{R}$ that map samples to a decision score. The ith classifier is trained with e_i as the *only positive sample* and $\{e_j \in E \mid l_{e,j} \neq l_{e,i}\}$ as the negative samples (*i.e.*, all exemplars with a different label). Other samples with the same label are not part of the ith exemplar's training set. Note that the outputs of these classifiers are normalized; see Sec. 3.2.

Our framework can use any linear classifier that represents each exemplar's classification score function as $C_i(x) = xw_i^T + b_i$ where $w_i \in \mathbb{R}^{1\times d}$ is the ith exemplar's weight and b_i is its bias. Representing exemplar classifiers by their coefficients has efficiency advantages. Let $\mathbf{W} = [w_1^T | w_2^T | \ldots | w_n^T]$ and $\mathbf{B} = [b_1, b_2, \ldots, b_n]$. This allows us to extract scores for several exemplars as a single matrix multiplication, $C^*(x) : \mathbb{R}^d \to \mathbb{R}^n = x\mathbf{W} + \mathbf{B}$, which can be easily parallelized as appropriate. We define C^* to be the *Exemplar Code extractor*, and its output as a sample's *Unnormalized Exemplar Code*. Exemplar Codes can be extracted in batch for several samples at once by stacking them vertically and broadcasting $\mathbf{B}$ to the necessary shape, leading to an extremely efficient extraction function.

3.2 Probability Normalization with Extreme Value Theory

Each column of an Exemplar Code corresponds to one exemplar's decision score which represents a notion of similarity between the sample and that exemplar. Since that exemplar classifier was trained on samples with other labels as negatives, the classifier will learn characteristics that distinguish it from samples with other labels. Intuitively, in a high-enough dimensional feature space where each exemplar lies on a spatial boundary, a sample that scores high on $C_i(x)$ is an outlier with respect to the rest of the dataset.

Unfortunately, one problem with treating raw SVM decision scores as a feature vector is that each classifier is trained independently with no joint regularization. This means the dimensions of a raw Exemplar Code will be inherently incomparable. To address this, we wish to define a calibration that maps $C_i(x)$ to the probability that x should be associated with exemplar i. Unfortunately, in a probabilistic sense, we cannot assume anything about the distribution of scores for a particular exemplar. In particular, we should not assume that score distributions are Gaussian as many existing normalization systems do. However, the extreme value theory (EVT) [21, 20] allows us to reason about the *tail* of the distribution. When decision scores are bounded—and SVM scores are bounded by the margin—EVT states that the score distribution's tail *must* follow a Weibull distribution [20]. This gives us a principled way to estimate probabilities for each exemplar's scores: by computing $C_i(t) \forall t \in T$, taking the top N_i scores, and fitting a Weibull distribution to these, we can define $P_i(C_i(x))$ as the Weibull CDF according to [20] that computes the probability that $C_i(x)$ is an outlier with respect to the rest of the training dataset. This is also the probability that x is associated with the exemplar. We set our tail size N_i to be 1.5 times the number of support vectors in C_i.

It is important to distinguish between normalizing with respect to the *exemplar* and normalizing with respect to the *category label*. Platt normalization used by the Exemplar-Ensemble approach [16] as well as W-score fitting [21] work differently. These systems make the simplifying assumption that samples from equal labels are alike, which allows them to normalize with respect to the distribution of *all scores in a particular label*. However, since different positive examples might be different for different reasons from samples in other labels, we wish for a more fine-grained formulation: we want the normalization to reflect the probability that a sample is associated with the ith exemplar, not just the set of all similarly-labeled samples. We cannot assume that all positives are drawn from the same continuous set and we wish to retain the individual granularity of each Exemplar Code independently. This is why we perform the fitting to the tail of the score distribution for *all samples* regardless of label. For brevity, let $P^*(x) = [P_1(C_1(x)), P_2(C_2(x)), \ldots, P_n(C_n(x))]$ be the sample's *Normalized Exemplar Code*.

3.3 Classification Decision

Normalized Exemplar Codes make an excellent feature representation for many kinds of final-stage classifiers. One could imagine approaches based on SVMs, random forests, or neural networks. In this paper, our focus is on efficiency; thus, we typically use linear classifiers. In this case, let the final decision score $f(x) = P^*(x)w^T + b$, where w and b are learned as part of the final classifier via SVM or some other linear classifier.

Note that even if linear classifiers are used for both individual exemplar classifiers and the final stage decision, the resulting classifier is not linear because of the probability calibration. However, it is still efficient – a classification decision takes n dot products, n Weibull CDF evaluations, and one last n-dimensional dot product. Further, the final model is compact, requiring $n \cdot (d+1)$ floats for the Exemplar Code classifiers, $2n$ floats for the Weibull scale and shape parameters, and n floats for the final-stage classifier. This is completely linear with respect to the number of exemplars in the dataset. Below, we show that the accuracy obtained using these models is statistically indistinguishable from that obtained from more complicated RBF classifiers used in other approaches.

3.4 Extreme Exemplar Codes

We find that the scores from many exemplar classifiers tend to be strongly correlated and not all are needed to produce a good classification decision. For efficiency, one can select a subset of the "extreme" exemplars from E. The resulting Exemplar Codes have dimensionality equal to the cardinality of the subset. Since decision time scales linearly with the number of exemplars, one can expect a $20\times$ speedup by selecting 5% of the exemplar classifiers as the "extreme" exemplars. This can be done naïvely by selecting a random fraction, or it can be done using iterated forward feature selection where we greedily add the exemplar that minimizes held-out validation error to our considered subset.

3.5 Cascade Exemplar-Code Classifiers

Finally, we fuse the extreme and full Exemplar Code decision functions for a small accuracy boost. When a sample's decision score with respect to the "Extreme Exemplar Code" linear classifier is *confident* (*i.e.* $|f_{\text{Extreme}}(x)| \geq \tau$), this classifier is often a better predictor than f. Thus, in training, we learn the threshold τ that maximizes held-out accuracy as decided by

$$f_{\text{Cascaded}} = \begin{cases} f_{\text{Extreme}}(x) & \text{if } |f_{\text{Extreme}}(x)| \geq \tau \\ f(x) & \text{otherwise.} \end{cases} \tag{1}$$

In this case, both speed and accuracy of f_{Cascaded} vary based on τ: if τ is too low, f_{Extreme} will always be used, leading to worse accuracy. However, if τ is too high, f will always determine the decision score, which means the system must always extract the full-dimensional exemplar code, decreasing speed.

4 Exemplar Codes for Efficient Facial Attribute Classification

In this section, we demonstrate Exemplar Codes within a face attribute extractor similar to the RBF attribute classifiers as described by Kumar *et al.* [13]. Our overall goal is to make face attribute extraction feasible in a real-time setting.

Each of the 73 facial attributes is a binary decision score that describes the presence or absence of an attribute; for example, a positive "glasses" score means the subject likely has glasses while a negative "male" score means the subject is likely female. We consider inherently multiclass attributes by splitting them into binary ones; for example, "hair color" becomes "has blonde hair," "has brown hair," etc., for each possible hair color. For a fair comparison, we retain the same formulation and consider each binary attribute as separate and independent from the others in our experiments.

The attribute extraction pipeline is quite similar to that in [13]. First, we extract face fiducial coordinates using the Everingham feature point extractor [6]. We use an affine transformation to warp the images to a common reference and use the same boosted feature extraction with the same attribute-specific learned features used in [13].

From there, we learn Exemplar Code classifiers as discussed in Sec. 3. Here, E and T are both drawn from the same training set and $L_e = L_t = \{1, -1\}^n$ are the positive and negative labels for the considered attribute. After probability normalization, a final linear classifier takes the sample's Exemplar Code as input and decides the single attribute label.

Since $n = m$ and $E = T$, each sample in the training set becomes both one exemplar/one dimension of the Exemplar Code feature vector *and* one training sample used to train the final classifier. Even in this "restricted" scenario, Exemplar Codes do not over-fit to the training/exemplar data. Note that this scenario differs from other approaches that require a separate "held-out" library of exemplar samples [24, 22, 23, 26, 2], which greatly simplifies data collection.

Once we learn Normalized Exemplar Code Transformer P^* from E, we collect Exemplar Codes for the training set $V = \{P^*(t_1), \ldots, P^*(t_m)\}$ and train the final stage classifier using V and L_t. For these experiments, both the final classifier and each exemplar classifier are linear SVM classifiers.

Dataset. To evaluate our attribute extractor, we use a private dataset of 197,933 face images split into 73 partitions for 73 different semantic attributes. Since each attribute contains a subset of the data, we consider each "attribute" as a completely separate independent experiment. Each image has one binary label that indicates the presence/absence of its

	System	Mean%	Max%	Min%	Better	Worse
A	Cascaded Exemplar Codes	0.0	1.6	-2.5	7	10
B	Extreme Exemplar Codes	-0.6	1.4	-3.9	1	21
C	Linear Classifier	-2.2	0.7	-8.5	0	43
D	Exemplar SVM Ensemble	-27.7	-0.9	-75.9	0	73
E	Platt Exemplar Codes	-0.5	1.2	-3.6	2	13
F	Unnormalized Exemplar Codes	-1.1	1.6	-6.2	1	33
G	LFW-Cascaded Exemplar Codes	-0.6	2.4	-5.1	2	16

Table 1. Performance of several systems. For each system, we show mean/max/min accuracy improvement compared to the RBF pipeline in [13] over all 73 attributes, how many attributes are statistically significantly better, and how many are worse. Cascaded Exemplar Codes (A) perform identically to the RBF pipeline and Extreme Exemplar Codes have minimal accuracy impact, but other systems perform noticeably worse.

attribute. The number of exemplars/training samples $n = m$ varies between 1244 to 5656 with a mean of 2711 images per attribute. Images were collected from around the web, filtered with a face detector, and groundtruth labels were collected with Amazon Mechanical Turk. We calculate each attribute's average accuracy with 5-fold cross validation on the attribute's partition. To decide whether the difference between two systems is statistically significant with respect to a given attribute, we compare the 5 scores of each system by calculating the p-value using a paired two-tailed T-test. If $p < 0.1$, we say the performance differs in a statistically significant manner.

4.1 Face Experiments

Using the above protocol, we compare Exemplar Codes to the RBF classifier pipeline described by Kumar *et al* [13], which uses an RBF SVM trained on learned features as described above. To reduce variation, we only compare the classification performance, ignoring the identical alignment and feature extraction steps. On our training data, the RBF pipeline achieves accuracies between 61.17% ("wearing necklace") and 97.55% ("color photo") across all 73 attributes. AFS' mean accuracy is 84.25%. Results for our systems are shown in Tab. 1. We discuss each row:

(Row A): First, **Cascaded Exemplar Codes' accuracy is statistically indistinguishable from [13] on most attributes** but it does have efficiency improvements in terms of memory use and classification speed. Our Cascade Exemplar Codes have no average accuracy change across all attributes, and accuracy is always within 2.5% of the RBF pipeline. Further, Cascade Exemplar Codes perform statistically significantly (*s.s.*) better on 7 attributes and *s.s.* worse on 10. Thus, the two systems are quite comparable. In return, we gain efficiency improvements. Storing all exemplar coefficients for the entire exemplar set along the final-stage classifier takes 32.6% of the disk space of storing the RBF classifier models. Classification throughput is also faster, with a 1.31 speedup factor over the RBF classifier. [1] Our classifier can calculate 65.43 image attributes per second on commodity hardware (ignoring feature extraction), which is just under 1FPS if we wish to extract all 73 attributes from each frame.

[1]Our exemplar codes and our RBF pipeline both use the same library to emphasize that the performance difference is not due to implementation. Cascaded Exemplar Codes perform 2.74 times faster than the original AFS system; Extreme Exemplar Codes perform a factor of 43.2 faster.

(Row B): We find that **using a small set of exemplars gives acceptable accuracy but an order of magnitude speed increase.** Using the greedy feature selection in Sec. 3.4, we keep only 5% of the exemplars in E, keeping all of T for training. The resulting *Extreme Exemplar Codes Classifier* is still always within 3.9% of the RBF pipeline. However, the extreme models take up $\frac{1}{60}$th of the disk space and are 21.9 times faster than the RBF pipeline. Our classifier can classify 1089 attributes per second, making it possible to extract all 73 attributes at 14.9 frames per second. At this speed, feature extraction (which we ignore) becomes the bottleneck.

(Rows C,D): Our Exemplar Codes perform much better than simple classifiers on this dataset. When we replace the original RBF classifier with a raw *Linear Classifier* (Row C) on raw image features, the linear classifier does *s.s.* worse on 43 attributes, sometimes up to 8.5% worse. The *Exemplar SVM Ensemble* method of [16] (Row D) where the label is chosen by the Platt-calibrated decision score from the maximum-scoring exemplar performs *s.s.* worse on *all* attributes. While it is better than chance most of the time, it fails on inherently multiclass attributes like hair color. In these cases, lifting exemplar decisions to our higher-dimensional feature space is necessary for good performance.

(Rows E,F): **EVT probability normalization is a better model than Platt probabilities**. Performance of our cascaded Exemplar Codes drops when replacing the EVT-based Weibull normalization step with raw Platt normalization to create *Platt Exemplar Codes* (Row E) or removing normalization completely to create *Unnormalized Exemplar Codes* (Row F), showing that W-score normalization is the best model for our scenario.

(Row G): Finally, we show that **Exemplar Codes can be learned on a completely separate dataset** without significantly hurting performance, even without using any groundtruth attribute labels. We set E to be 5,000 random images from LFW [9] and $l_{e,i} = i$ to be garbage labels so each exemplar is trained with all $n - 1$ other examples as negatives. These *LFW-Cascaded Exemplar Codes* (Row G) perform only 0.6% worse on average and the maximum drop is not more than 5.1%. Using properly labeled exemplars could help here, but even without any human effort, performance is still comparable and is still much better than simpler classifiers.

5 Exemplar Codes for Tattoo Classification

Tattoos, unlike faces, have no pre-defined shape and are not well suited to detection by a cascade classifier. In our tattoo extraction and classification pipeline, we therefore use an exemplar SVM ensemble object detector to detect the region of interest surrounding the tattoo. The maximum similarity score tells us whether the contents of the sliding window are similar enough to *any* of the tattoo exemplars to plausibly be a tattoo. We then categorize the tattoo via a two-stage Exemplar Code classifier. The task of tattoo extraction uses the methodology of [16], in which each ensemble of exemplar SVMs is trained, with one positive manually cropped image and many negatives hard-mined from thousands of images that do not contain the tattoo category. Exemplars are trained on a multi-scale pyramid of HOG features which serve as base features for classification.

During training of the exemplar ensemble, a linear SVM is trained for each exemplar using a single positive sample and a set of negative samples. The negative samples are divided into a number of subsets and mined one subset at a time; as a result, the SVM parameters are updated iteratively. This training process is repeated for each exemplar SVM until an ensemble of classifiers is obtained for each class. Because different exemplars may produce similarity scores in different ranges, normalization of the ensemble is performed prior to training the second stage classifier.

Following the methodology of [16], a validation set of negatives, equal in cardinality but disjoint from the negative training set, is used for normalization. We determined match and non-match labels as windows having at least 50% overlap with the ground truth bounding box and at most 20% overlap with the ground truth bounding box respectively. We discard overlaps falling between 20% and 50%, then learn normalization parameters using the similarity scores and determined labels. Once we have acquired normalization parameters, the similarity scores of the training set are normalized via these parameters and the resulting Exemplar Codes are submitted to the second stage classifier. Similarity scores are computed for all classes in order to determine a multi-class label for the given region of interest.

We used random forests for our second stage classifier. Recently, random forests have gained attention for pattern classification due to their effectiveness on nonlinear problems [4]. Our second stage random forest classifier takes a feature vector consisting of Exemplar Codes for all tattoo classes and classifies it through several decision trees, constructed via training on randomly sampled feature vectors with replacement. Each node is split based on an out-of-bag subset of variables. Every tree performs classification and provides a vote. The class label is determined by the majority vote of all the trees. One of the advantages of random forests is that there is no need for variable selection, which makes them suitable for features characterized by a large set of Exemplar Codes. We used the OpenCV implementation to construct our second stage classifiers.

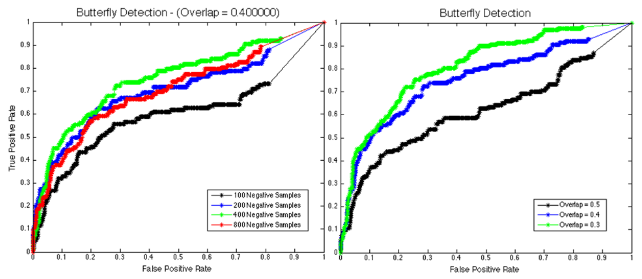

Figure 3. An evaluation of the ensemble detector. The ROC curves on the left indicate that 400 negative examples (green) perform best for when used with 50 positive exemplars. The ROC curves on the right show that 40% overlap (blue) provides a good compromise between detection precision and recall.

5.1 Tattoo Experiments

We evaluated the tattoo detection algorithm using 50 segmented positive examples of butterfly tattoos for training and 100 non-segmented images for testing. We determined the detection rate in terms of the amount of detected bounding box overlap with ground truth. Intuitively, requiring a high percentage of overlap lowers the detection rate, because only a very similar exemplar can yield a precise bounding box. In practice, we found that 40% overlap offered a good trade-off between detection rate and precision. Since exemplar SVMs are parameterized principally by their negative training data, we also examined how the number of negative examples impacts detection performance. Interestingly, we found that using a negative set of size 400 yielded better performance than using either a negative set of size 800 or a negative set of size 200, which we suspect is due to a need to obtain some optimal level of generality for each exemplar SVM. Our results are summarized in Figure 3.

To evaluate the classification performance of our Exemplar Code classifiers, we performed tests using images of five classes of tattoos including butterfly, skull, flower, star, and dragon; class names are taken from [18] where applicable. Examples of these tattoo images are given in Figure 2. We collected a test set of 50 butterfly, 50 skull, 46 flower, 50 star, and 42 dragon pre-segmented tattoo images from Google Images. The disjoint training set consisted of 50 butterfly, 60 skull, 80 flower, 64 star, and 50 dragon pre-segmented tattoo images and 400 negative samples. We evaluated classification performance under both Platt normalization [19] and W-score normalization [20] of the ensemble. In both cases random forests were used as a second stage classifier. Confusion matrices are shown in Fig. 4. Platt normalization yielded 60.5% classification accuracy and W-score normalization yielded 63.8% accuracy. This result provides additional evidence that W-score normalization offers superior score calibration performance to Platt normalization.

We also performed a preliminary evaluation of Exemplar Codes versus PiCoDeS [2] and the Exemplar SVM ensemble [16] alone. Our preliminary results show that for closed-set recognition performance, PiCoDeS perform similarly to the Exemplar SVM ensemble, but Exemplar Codes outperform both systems on most classes.

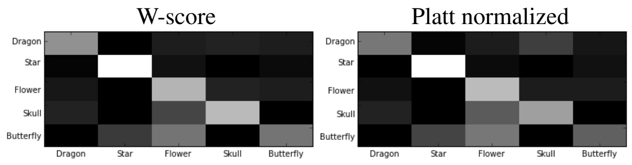

Figure 4. Confusion matrices for Platt normalization (left) and W-score normalization (right) Exemplar Code tattoo classifiers. While the two matrices are similar, Platt scores confuse Dragons/Skulls. Platt's overall accuracy is 60.5%, and W-scores is 63.8%.

6 Conclusion

We introduced a flexible, scalable mid-level representation that achieves accuracy on par with state of the art heavy-weight classifiers at significantly less computational cost. We demonstrated this capability on the task of facial attribute classification, and we have shown that efficiency can be further increased with little loss in accuracy by intelligently selecting subsets of exemplars. For comparison, we released Exemplar Code attribute scores for the LFW data set.[2] Implementation of both a facial attribute classification pipeline and a tattoo classification pipeline demonstrate that our Exemplar Code representation is flexible enough to be applied to diverse problem domains. Moreover, Exemplar Code representations are particularly suited to classification problems, such as tattoo recognition, in which intra-class diversity and non-linearity are too great to be easily characterized by conventional classifiers. We also provided empirical evidence that demonstrates the superiority of statistical extreme value theory over Platt normalization for the purpose of calibrating ensemble scores. In future work, additional applications of Exemplar Code classifiers shall be investigated, as will better techniques for selecting subsets of exemplars.

Acknowledgments. This work was supported by ONR SBIR Award N00014-11-C-024, Army SBIR W15P7T-12-C-A210, and ONR MURI N00014-08-1-0638. We thank our SBIR partners, Automatic Face Systems, for the access to their attribute dataset and feature vector parameters.

References

[1] S. T. Acton and A. Rossi. Matching and retrieval of tattoo images: Active contour CBIR and glocal image features. In *SSAI*, 2008.

[2] Alessandro Bergamo, Lorenzo Torresani, and A. Fitzgibbon. PiCoDeS: Learning a compact code for novel-category recognition. In *NIPS*, 2011.

[3] A. Bergamo and L. Torresani. Meta-class features for large-scale object categorization on a budget. In *CVPR*, 2012.

[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] D. Cox and N. Pinto. Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *FG*, 2011.

[6] M. Everingham, J. Sivic, and A. Zisserman. Hello! my name is... Buffy — automatic naming of characters in TV video. In *BMVC*, 2006.

[7] H. Han and A. Jain. Tattoo based identification: Sketch to image matching. In *ICB*, 2013.

[8] B. Heflin, W. Scheirer, and T. Boult. Detecting and classifying scars, marks, and tattoos found in the wild. In *BTAS*, 2012.

[9] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, University of Massachusetts, Amherst, October 2007.

[10] A. K. Jain, J.-E. Lee, and R. Jin. Tattoo-ID: automatic tattoo image retrieval for suspect and victim identification. In *Adv. in Mult. Inf. Proc.-PCM*, 2007.

[11] A. K. Jain, J.-E. Lee, R. Jin, and N. Gregg. Content-based image retrieval: an application to tattoo images. In *ICIP*, 2009.

[12] N. Kumar, A. Berg, P. N. Belhumeur, and S. Nayar. Describable visual attributes for face verification and image search. *TPAMI*, 2011.

[13] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *ICCV*, 2009.

[14] J.-E. Lee, A. K. Jain, and R. Jin. Scars, marks and tattoos (SMT): soft biometric for suspect and victim identification. In *Biometrics Symposium BSYM*, 2008.

[15] J.-E. Lee, R. Jin, A. K. Jain, and W. Tong. Image retrieval in forensics: tattoo image database application. *Multimedia*, 19(1), 2012.

[16] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *ICCV*, 2011.

[17] D. Manger. Large-scale tattoo image retrieval. In *Computer and Robot Vision (CRV)*, 2012.

[18] E. Newton, G. Coleman, and Y. Patrice. Data format for the interchange of fingerprint, facial, & other biometric information part 2. Technical report, NIST.

[19] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. in large margin classifiers*, 1999.

[20] W. Scheirer, A. Rocha, R. Micheals, and T. Boult. Robust fusion: Extreme value theory for recognition score normalization. In *ECCV*, 2010.

[21] W. J. Scheirer, N. Kumar, P. N. Belhumeur, and T. E. Boult. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *CVPR*, 2012.

[22] F. Schroff, T. Treibitz, D. Kriegman, and S. Belongie. Pose, illumination and expression invariant pairwise face-similarity measure via doppelgnger list comparison. In *ICCV*, 2011.

[23] Y. Taigman, L. Wolf, and T. Hassner. Multiple one-shots for utilizing class label information. In *BMVC*, 2009.

[24] Thomas Berg and Peter N. Belhumeur. POOF: part-based one-vs-one features for fine-grained categorization, face verification, and attribute estimation. In *CVPR*, 2013.

[25] L. Wolf, T. Hassner, and Y. Taigman. Similarity scores based on background samples. In *ACCV*, 2010.

[26] Q. Yin, X. Tang, and J. Sun. An associate-predict model for face recognition. In *CVPR*, 2011.

[2] `http://www.metarecognition.com/exemplarcodes/`