Improving Performance and Applying Cascades in Visual Classification

Master Thesis

Michal Kiwkowitz

Contents

ABSTRACT

Computer-based object classification has improved consistently over the last decade, but the performance of current computational schemes is still significantly lower than human classification performance. In addition, the errors made by current classifiers are often unreasonable by human standards. Since improvements to performance become increasingly difficult to achieve when absolute performance is already high, it is currently unclear which future directions will be useful for reaching truly human level performance.

In this work I examined two general directions for future improvements in current classification scheme. One general direction assumes that current methods extract only a part of the available information for classification from the training set, and attempts to identify the main possible sources of additional information. I examined a number of plausible sources for possible improvements. Some of the methods under study, but we concluded that they are unlikely to be sufficient by themselves to reach human-level performance.

The second general direction assumes that a major limitation comes from the size of the training set: training sets used in practice may be inherently insufficient to capture all the necessary variations needed to learn a truly high-performance classifier. Our experiments suggested that classification performance indeed increase monotonically and without clear saturation as the training set increases in size. We also found that the set of features used for classification needs to be increased to capture the additional information in the increased training set.

These results have two implications to the classification scheme. First, they raise the advantage of constructing classifiers in an on-line manner, in which the classifier is continuously improved by new examples. Second, they raise the problem of increasing computational load as the number of features used for classification increases.

To deal with these problems we developed and tested two classification schemes. Both are multi-stage, or 'cascade' methods in which all images are analyzed in the first stage, and, depending on the results, only some of them are analyzed further by subsequent stages. The first of these methods was a so-called 3-way cascade, which is an extension of previously used cascade methods. In this method, a first-level classifier is first applied to the new image to be classified. If the response is high or low enough a decision is made, otherwise, a second-level classifier, which uses more features, is applied to the image to allow a more confident response. The second multi-stage scheme we developed is the so-called configuration cascade. In this approach, the decision to process an image in additional stages is based on the particular feature configuration discovered in the image. Using this scheme, erroneous configurations of the first-level classifier are processed by a second-level classifier which uses more features and achieves better performance. We showed that the number of possible erroneous configurations is bounded in practice, and that the error on the erroneous configurations is reduced after additional processing. This scheme leads to a continuous improvement in performance, with only a small increase in computational cost.

1 Introduction

The field of visual recognition has been making continues progress in the area of classification and recognition. However, up to date, no classifier for a real complex class has achieved classification without errors or fully human-level performance. In this work we examine the sources of these errors, and possible methods to improve performance without a large increase in computational cost.

We start by investigating the type of errors made, using as an example the fragment based approach [2], to classify faces and non faces, and evaluate in Section 2 the classifier's performance and the errors it makes. We then survey existing classifiers and their advantages and drawbacks, in Section 3 . In Section 4, we examine different ideas for improving the performance of the fragment based classifier. We continue by identifying the training set size as an inherent boundary to the performance, and the implications of this observation on classifier's computational cost in Section 4.2 . We suggest in Section 5 two methods of cascades, which can improve performance with relatively low additional computational cost. The first, a three-way cascade described in Section 5.1 , which performs further processing on images near the margin, and the second, a configuration cascade described in Section 5.2 which uses the configuration of the detected features to determine whether an image requires additional processing in the cascade.

In the next section we examine more closely the performance of current classifiers and our goals and motivations for the current work.

2 Evaluating Classification

The goals of our work are to answer the following questions:

- Can the performance of current state-of-the-art visual classifiers be considered satisfactory?
- Are we learning all we can learn from the training data?
- What are possible ways of improving performance further?

In this section we survey methods of evaluating classification performance. We then consider the performance for practical applications and examine the amount of errors

made by machine classifiers compared with human performance. The errors can be generally divided into two types: false negative errors, images that have been rejected falsely, and false positive errors, images in which the class has been falsely detected. We begin with an overview of the methods used to evaluate a classifier.

2.1 Evaluation Methods

Judging the performance of a classifier is not a trivial question. Many authors plot and compare the Receiver Operating Characteristic (ROC) curve (seen in Figure 1) of different classifiers, and although error rates may be small and satisfactory, the error instances themselves are quite alarming, Figure 2 and Figure 3 show some examples. This graph enables understanding of the relationship between the different errors made by the classifier. A threshold on the response of the classifier determines a desired hit rate and respectively a false positives rate that depends on the classifier's ROC curve. In an ideal world we would be able to set the threshold so that our classifier will get 100% hits and 0% false positives, a perfect classifier with no errors. In reality, there is no perfect threshold, and the possibility is to select some acceptable mixture of false positives and false negatives.

The ROC, although very useful, is not a simple parameter to compare. A measure that is easier to compare is the Equal Error Rate (EER) of a classifier. This is the location on the ROC curve where the percent of false positives equals the percent of false negatives. In a perfect classifier the value of the EER will be zero and therefore there will be a threshold that allows the classifier to make a perfect decision every time. A similar measure is the Minimum Error which provides the minimum total error (false positives and false negatives). The EER and the Minimum Error are usually found at close locations on the ROC curve. Another way to appraise the ROC is to measure the percent of the area below the ROC curve; the perfect ROC will have a value of one.

For most real world classification problems, there are no perfect classifiers; many classifiers improve the classification by improving the ROC curve at the top left corner. In many cases the changes are very subtle and the comparison of EER, area or even the different ROC curve misses a lot of information about the classifier. Given two classifiers with the same ROC or very close ROC how can they be compared? Consider a vendor that

is purchasing a classifier that uses images to alert him when a police car passes by his property. He is offered two classifiers with very good ROC curves but imperfect ones. Shouldn't he consider the type of false positives made by these as an additional method for rating their performance? Wouldn't a classifier that detects a cab as a police car every once in a while be better than one that detects a cat or just the wind in the trees? Similarly, when a child calls a cow a horse, his parents will happily just let him know that he is wrong, but shouldn't they pay a little more attention if their child is using the word cow when he encounters shoes or when staring at a clear wall? The error instances themselves are important and there is much to learn from them. We next examine the EER and the type of errors made by current classifiers.

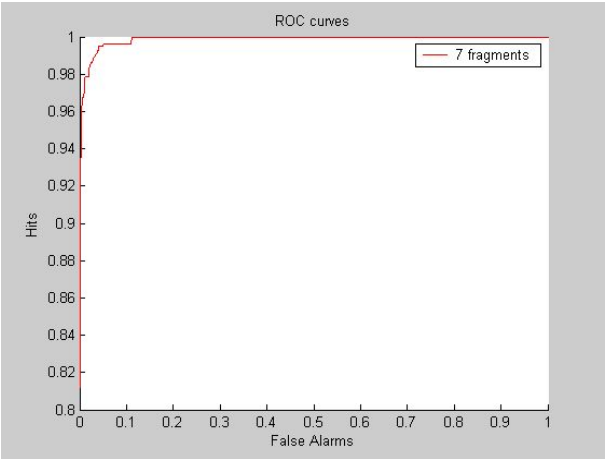


Figure 1: Classifier Performance. The Reciever Operating Characteristic curve of a face classifier using seven fragments as described in [3]. Although the ROC is good, it is imperfect.

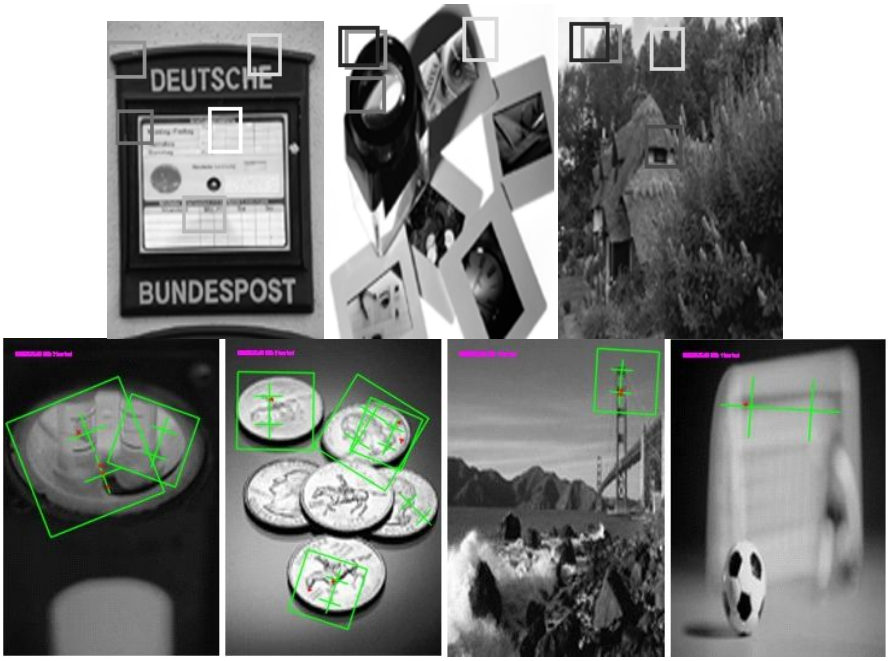


Figure 2: Faces Found by Face Classifiers. Images detected as faces by two different face classifiers with high performance. On the top row are results from a fragment classifier as in [3], the fragments detected are marked. On the bottom row are false faces detected by the Betaface classifier [8]. This classifier detects the eyes in an image and marks them in green.

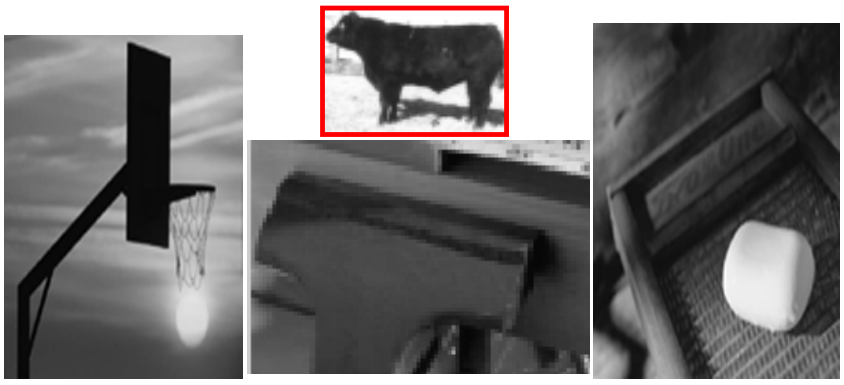


Figure 3: Cows Found using SIFT. These images recived very high scores and were classified as cows by a classifier that uses SIFT features [7], we show a sample cow in the red rectangle.

Figure 4: **Classifier Performance on Specific Images**. On the top row are some misses of the classifier, in the center are some false alarms or false positives. The bottom row shows some images that we believe may serve as more reasonable false alarms.

2.2 Practical Application of Classification

Although the equal error rate in some of the best classifiers can reach 2-6 %, this is still quite high. Consider searching a natural scene and trying to determine if there is a horse in the image. Suppose that the image is 1000 pixels high by 1000 pixels wide, and we expect the horse object to be found in a window of around 50 by 50 pixels. Searching the image serially with local windows with a 50% overlap, we will need to check 40x40 locations, which is 1600 sub windows. With an error rate of 2%, this means that 32 sub windows will be falsely detected as horses, which is not very practical. In the next section we contrast human performance with the performance of machine classifiers.

2.3 Machine Classification vs. Human Classification

If humans had the false alarm rate mentioned in the previous section, they would see horses everywhere, or in other words they would be prone to mistakes and visual hallucinations. Compared with State-of-the-art classifier's performance, humans perform better: for most databases or classification tasks there is no question whether machine performance is better or poorer than that of humans, humans outperform current classifiers. When comparing machine performance to human performance, the task of classification is harder on machines, given that the in-class variability is greater. Machine software is closer to human performance in somewhat simpler tasks like recognition, that are controlled for rotation and lighting. Not many real comparisons have been made between human and machine categorization, perhaps because it is common knowledge that humans are better. A comparison study by Adler and Schuckers (2006) compared automatic face recognition technologies available in 1999, 2001, 2003, 2005 and 2006 with human performance and showed that as automatic face detection improves, the average human performance is no longer better than the machine performance. This is impressive, yet we need to consider that some human performance is nevertheless better than the best automatic face recognition, and the task at hand is answering 'same' or 'different' to pair of images from the NIST Mugshots Identification Database, where the match is between images of the same person at different age points which may cause large facial differences, see Figure 5 and [1] for more details. The field of face recognition and classification is one of the better studied classes, on other classes (planes, motorbikes, dogs, etc.) performance is usually poorer. As mentioned in the previous section, humans do not make as many false positive errors as machine classifiers; this fact is important for understanding the human visual system and for improving machine classification. In the next section we take a closer view at some classification error dividing them into false negative and false positive errors.

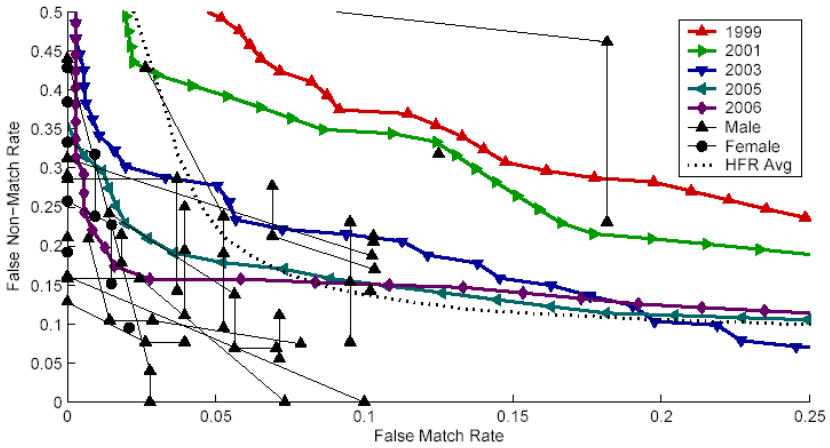


Figure 5: Comparison of face recognition software and human performance from [1]. The black circles and triangles indicate the human female and male performance respectively, and because they could respond on a scale of 1-5 conveying their confidence in the match, an curve was drawn for each human by adjusting the threshold on their confidence, the continues curves show the results for the highest performing software available to the authors of [1] in the specified years, and the dotted line is an average of the human performance, that is highly affected by the outliers. Most human performance remained at the bottom left corner with the lowest error rates, and since where the human curves do not completely outperform an automatic curve, the authors counted it as an intermidiate performance (33.3% comparing to the 2006 condition), they got a good score for the 2006 comparison. This may be challanged if the statistics where different.

2.4 False Negative Errors

When we examine the false negatives (misses) that current face classifiers make we can often understand them (see Figure 4): many times misses occur in challenging face images with long hair, high foreheads, glasses, or in images that contain non frontal faces, strong shadows or noise. This seems intuitive and is consistent with human performance: infants are known to sometimes burst in tears when seeing a familiar kin that has grown a beard or put on new glasses. We can assume that large changes in a face that were not seen before may not be encoded as part of the stored features and may cause unfamiliarity. These errors are usually understandable; furthermore, algorithms have been suggested to reduce the rate of false negatives by adding more training samples, selecting more features or boosting the classifier [13]. The false positive errors, discussed in the next section, are more interesting.

2.5 False Positive Errors

False positives errors are those images that the classifier falsely detects as class but they are not. They can be divided into two main types of errors, which I call multi-class errors and single-class errors. As multi-class errors we consider those errors that are caused by confusion between similar classes. For example, when trying to classify horses, we may recognize some cows as horses. Since both classes are four legged animals, we consider these errors reasonable; extending the classifier into a multi class classifier to learn the similar class, or adding the similar class to the training samples labeled as non-class would resolve this problem. Similar errors are also made by human infants before they learn the name of the new class.

The single-class errors are those images that do not resemble the class and are nevertheless classified as the class; see Figure 4 for some examples. Although the classifier shown has a fairly good ROC curve, it makes unreasonable mistakes that a two-year old would not make, failing to see for instance that a register is not a face. This problem of unreasonable false alarm (compared with human judgment) is not easily avoided, it persists in many different types of classifiers and when using different kind of features. Using a Naive Bayes classifier that extracts SIFT features as in [7] the problem persists, examples with their likelihood scores can be seen in Figure 6. In Figure 2 we show another algorithm, Betaface [8] that detects faces by finding the eye region, and fails as well when provided with queries of some non-class images that are hard for the fragments classifier.

The single-class false positives are interesting since it is clear they are unreasonable.

We continue to discuss some popular classification methods, their relative advantages and shortcomings in section 3 . In Section 4 we use our observations to suggest and compare possible general methods for potential improvements in performance of the classifiers.

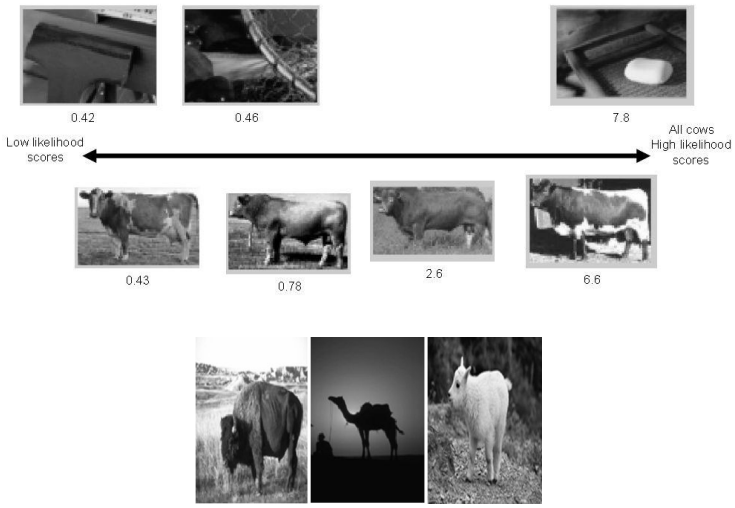


Figure 6: Cow Database Likelihood. On the top some examples of the likelihood scores produced by a cow classifier based on SIFT features [7]. The false positives are again irrelevant to the class of cows. On the bottom some examples of images that we consider more relevant as false positives**.**

3 Existing Classification Methods

In this section I survey the main relevant classification methods, the features they use, and how they combine features to reach a decision. The goal is to identify possible reasons for failure and possible methods for improvement.

Classification is the process of determining a label Y_i for each sample X_i for some unknown function $F(X_i) = Y_i$. In our discussion we focus on problems where the samples are 2D images and the labels are binary class vs. non-class. Most supervised methods work in a similar manner, as follows:

(i) Extract features that represent the sample data in some finite dimensional space.

(ii) Select a subset of features that can be used to separate between class and non-class and will not over fit the training data.

(iii) Combine the features found to a score that can predict or classify the sample correctly.

We begin by examining different types of features used and how a subset is selected in section 3.1 . Then we continue to discuss how the features are combined to form a decision in section 3.2 . In section 3.2.3 we view ensemble methods that combine classifiers in order to improve performance. We also discuss online learning methods in section 3.3 .

3.1 Feature Representation

Most state of the art classifiers represent an object or category using a set of local features rather than a single template, and then combine the features to a single score. Classifiers differ in the number of features they use, from a few to thousands, and in the type of information used such as appearance, location, or statistics. The challenge is finding feature representations that will represent intrinsic object variability within the class, whilst maintaining invariance with respect to reasonable changes in translation, rotation, scaling and illuminations. Features can be very simple or very complex; Vidal-Naquet and Ullman discuss in [3,4] the idea of intermediate features. A classification algorithm can use a small set of complex features combined in a simple way or an exhaustive set of simple features combined in a very complex classifier and achieve similar performance. In the following subsections we show a variety of features, from the fragment based approach which uses intermediate complexity features combined in a simple statistical method, to SVM method which combines simple features in a more complicated scheme.

3.1.1 Fragments

In [2, 3, 4] highly informative class image patches at different sizes are extracted to represent the class. The patches, referred to as fragments are matched to images using the Normalized Cross Correlation (NCC) method, although other similarity measures can work as well. For each fragment, a threshold is determined, that maximizes the fragment's mutual information with the class variable, and the detected fragments are combined using a generative Bayesian algorithm. In order for the fragment approach to be robust to scale, and rotation, variable scale images can be added to the training set, or the features can be extended to multi resolutions.

3.1.2 Gabor features and wavelets

Some approaches create features that are themselves more resistent to illuminations, rotations, scale and translations. The Gabor features [19] are one example. 2D Gabor filter are defined by harmonic functions modulated by a gaussian distribution. These are applied to any image using convolution to obtain a response. The Gabor features are closely related to processes in the mammalian primary visual cortex, and have been showed to approximate the response of simple cells in the visual cortex.

3.1.3 SIFT

Another approach that is resistant to transformations and rotations has been suggested by D. Lowe and is referred to as SIFT - Scale Invariant Feature Transform [7]. This approach uses complex keypoint descriptors as features applied to images preprocessed by a Difference of Gaussian filter at multiple resolutions. The features are created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location; these gradients are then weighted by a Gaussian window, and accumulated into orientation histograms summarizing the contents over cubic sub regions of a chosen size. The final feature is a vector the size of the sub regions with information about the orientation and magnitude at multiple resolutions.

3.1.4 Improving Features - Hierarchies and Semantics

Features of all types can be improved using various methods. Examining methods to generate more precise features, we encounter the hierarchical fragments [18], which are more complex features extracted by decomposing image fragments using a probabilistic hierarchical model and combining their inner informative subparts. The hierarchical decomposition generates a more robust fragment, allowing some flexibility between the fragment parts and defining the more important subparts of each fragment. Another way to increase the effectiveness of a fragment feature is to add an inherent variability to the feature, as in semantic fragments [17]. In this scheme, 'anchors', stable image parts in the training set, predict a geometric model and using this model other more variable parts and their semantic brothers are detected. The fragments used for classification can now be

semantic fragments with variable appearance (for example a mouth with teeth, smiling or frowning) creating a more resistant representation and improving performance.

3.1.5 Constellation Models

In the constellation model, Fergus et al. [10] models variability not only in appearance but also in the shape of objects. The features are image parts reduced to the first *k* (typically 10-15) principle components of the image patch, but combined not only using a probabilistic appearance model but also considering shape, occlusions and scale. In this model the features' relative location is used so that the shape and scale are represented by a joint Gaussian density of the locations of features. Another constellation method applied to face detection proposed in [20] uses gabor-filter based complex statistical features and adds a phase of registration using a transformation / constellation of three landmark features followed by an appearance verification stage that gives a probabilistic score to the geometrical model of the detected features.

3.1.6 Satellites for Similar Classes

We momentarily extend our discussion, limited to single class classification, to similar classes to survey satellite features, as we will refer to them later in section 4.1.6 . When necessary to differentiate between two similar classes, the problem becomes harder. Learning features that differentiate between faces and non-faces is different from learning the difference between women and men faces. Many of the previously discussed approaches will perform better on the class/non-class problem and need some extension in order to solve the similar class problems. However there are methods that deal specifically with this problem, we consider the satellite fragments classifier [16] by B. Epshtein. Assuming that the features that differentiate between similar classes are smaller and more localized, the typical method of finding high mutual information fragments will not succeed since small features can be found in many image locations. In the satellite approach, the classifier finds basic features that are shared by both classes in order to generate a geometric model and find more specific and localized satellite features that will represent the small differences between the similar classes, For example, in the case of

men and women, an earring on the ear is a highly informative feature or the existence of facial hair above the lip is another informative feature.

3.2 Learning and Classification

Once the type of features to use has been determined and the data is represented using these features, learning is performed to generate a classifier that can classify unseen examples (referred to as a testing set). Classification methods exist for unsupervised learning where the training data is not labeled. However, we will focus on the family of supervised classifiers that have at their dispose labeled training data. When given some representation of the data, classifiers select a smaller set of representative features that will not over-fit the training data and will generalize well to new instances of the class. The features are combined into, usually in a none rigid ways or using a probabilistic method, to generate a single score that determines the class labels and sometimes a predicted value in regression problems.

Some authors [11] separate the feature selection methods into filters and wrappers. Filter methods are those methods that select features without considering the specific classification method that is used, or considering it but not directly. Wrapper methods rely on the performance of the classifier on the training set to evaluate the quality of the features and to determine which features to select. The later are prone to over-fitting the training set. We start by describing the filter method used by the fragment based approach as in [2,3]. We then continue to describe SVM which is another filter method, and in the next section we describe some ensemble methods, among which AdaBoost functions as a wrapper avoiding over-fitting in a different way.

3.2.1 Max-Min Selection and Bayesian Classification

In the fragment classification described in [2] ,[3], features are selected by a greedy search to maximize their mutual information (for a detailed description see appendix 8.1). In short, the algorithm has a pool of fragments or image patches that have a similarity to all training samples; typically this is the maximum normalized cross correlation with the training sample. The Class variable C has binary value 0 or 1, representing non-class and

class, respectively. Each fragment *f* is changed to a binary feature using some threshold θ which maximizes the mutual information between the fragment and the class.

A smoother solution to the naïve Bayes decision is a soft naïve Bayes decision that models probabilistically the similarity expected allowing usage of the similarity measurement, thus explicitly avoiding information losses by changing the fragments into binary features. The soft naïve Bayes outperforms the simpler naïve Bayes. Figure 7 shows a comparison of the two.

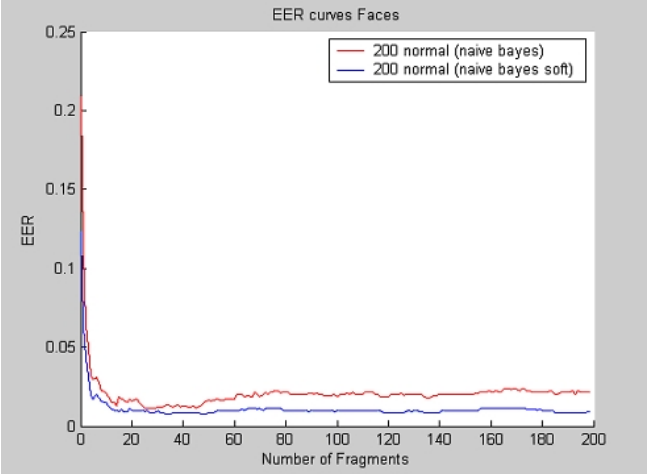


Figure 7: Comparing Naive Bayes with a soft Naive Bayes. The EER when using up to 200 fragments on a faces classification task and combining them as binary features using simple naive Bayes or using a softer version that uses a gaussian distribution on the similarity strength.

3.2.2 SVM

SVM or Support Vector Machines is sometimes considered the best performance classification method today, but other methods perform close to SVM [21]. Given a set of features and class labels, support vector machines minimize the empirical error while maximizing the geometrical margin; they are remarkably resistant to over-fitting. SVM uses kernels to map data into a high dimensional space. Some common kernels are linear, radial, or polynomial kernels, but others can be defined as well. The decision surface is defined by support vectors: some mapped training samples that define the margin as well as possible. Using SVM is equivalent to solving a quadratic minimization problem with

linear constraints; there are many works of optimizing SVM algorithms [22]. SVM works close to other methods that maximize the margin. One known method is boosting, a type of ensemble algorithm which we discuss in the next section.

3.2.3 Ensemble Methods

Ensemble methods are methods that combine several classifiers into a single classifier with better performance. They can use various classifiers like Bayesian classifiers, neural networks, decision trees or others as the basic classifiers they combine [23]. We will discuss shortly bagging and then boosting which we will elaborate on since it will be used later in our work.

~~3.2.4~~ 3.2.3.1 Bagging

In bagging, a set of classifiers is trained on the training set multiple times by re-sampling from the training set with repetitions and training each time a different classifier, with different parameters according to the random training set selected [23]. The scores are then combined into a single classifier by averaging over the scores or by voting - labeling according to the class that more classifiers voted for. Bagging creates a more stable algorithm by averaging while reusing the same data.

~~3.2.5~~ 3.2.3.2 Boosting

Boosting is a general term that refers to combining several rough classifiers into a single very accurate classifier with better performance. Many boosting algorithms have been suggested, one common method is AdaBoost [13]. AdaBoost linearly combines weak hypothesis h by weighting the training samples and concentrating on the errors. AdaBoost is described in detail in appendix **Error! Reference source not found.**.

AdaBoost has a theoretical bound on the generalization error. However, empirically it is known to drive down the generalization error long after the training error has reached zero, contradicting the spirit of the bound and avoiding over-fitting. Many explanations have been offered to explain this behavior [24], [25]. The idea is that similar to SVM, AdaBoost effectively increases the margin on the training set which results in an improved bound on the generalization error. Another type of combining features which we will refer to in our work is the cascade classifier described in the next section.

~~3.2.6~~ 3.2.3.3 Cascades

In the cascade method suggested by Viola and Jones [5] complex classifiers are combined in a computationally efficient way. The classifiers are applied in successive stages of a cascade allowing the initial fast stages to filter out many sample images that are non-class with high probability. Since images that are rejected in the initial stages will not be processed again, the cascade uses complex classifiers and chooses a threshold that can ensure a very low false negative rate. Cascades allow for rapid processing and applying computationally demanding classifiers that can achieve very low error rates without paying the cost of low speed performance. The method also has extensions, for example, in [6] a method that deals with asymmetric data sets is suggested. We will return later to discuss the advantages of the cascades in the Section 5 . Until now we described batch classifiers we turn now to discuss classification in an online setting.

3.3 Online Classification

Online classification methods consider the problems that arise when dealing with real world problems. Many times the training set is too large to process directly or not available when initially learning. A good classifier is one that can adapt easily when more data is available, use additional features that were not available when learning initiated, combine additional data into its decision scheme simply, use a small amount of memory when learning, learn from single examples as they arrive or from blocks of examples and all this while achieving performance that is close to that of a batch algorithm that has all the samples and the memory available at all times. Online algorithms have been suggested for many classification schemes, for the fragment based approach [14], and for boosting [12]. These methods keep in memory the minimum necessary to explain samples seen so far and extract features and statistics out of incoming samples. These methods are encouraging when considering that many times errors are caused by new class appearances. The possibility to easily train a classifier on new samples allows reaching ever lower error rates. Another advantage is that the online scheme seems closer to how human beings learn. In the next section we discuss our ideas for improving classification.

4 Improving Classification

In previous sections we discussed current classification schemes, and the quality and rate of their errors. In this section we discuss the role of a number of possible sources to the limitations of current classifiers and continue to propose some ways to improve classification. The ideas for improvement can be viewed in two main contexts (Figure 8): (i) improving classification by learning more from the available training data, (ii) since the training data may be an inherent limitation to the performance and a major difference between humans and machines, we propose ways of dealing with ever increasing training data and class features not only for learning, but also for achieving accurate and fast response times considering these limits. In Section 4.1 we discuss the first possibility, and in Section 4.2 we demonstrate the training set limit.

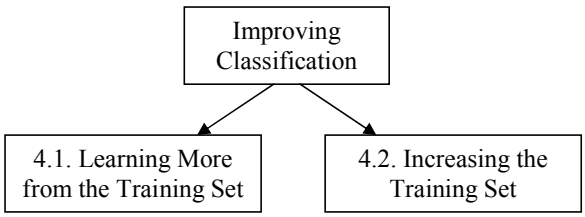


Figure 8: Two main learning paths.

4.1 Learning More from the Training Set

In this section we discuss limitations of current classifiers and ask: are we extracting all we can from the training set? Are the feature detectors good enough? Are the classifiers combining the features in the best possible way? Is there more information that is overlooked, the geometry for example? We begin by discussing possibilities to improve classification, and then propose three different schemes that try to exhaust the training set data in the quest to achieve better performance. In current classification schemes decision is often reached based on restricted and partial information, finding a small number of local fragments which are far from covering the object, and may have a too limited view of the object. Figure 9 demonstrates this. To avoid this limitation we introduce the perspective fragments that try to add context to each fragment in section 4.1.4 . The anti

fragments in Section 4.1.5 try to generate more accurate features by learning more from the false positive errors the classifier makes and defining the similarity to a fragment better. In Section 4.1.6 the satellite fragments are introduced as a second stage of classification. The approach uses a first quick stage as a mean of extracting better localized features for the second stage.

4.1.1 Improving the Similarity Measure

Does the type of feature make a difference? Some methods excel on some problem sets but are outperformed by other methods on other sets. Can a human mistake an arbitrary object for a face when given the features the classifier is using? Figure 9 shows the fragment features detected in different images by the Normalized Cross Correlation (NCC) similarity measure. Do the detected features in the false positives seem sufficient to determine the image is a face? Do the detected features in the false negatives seem insufficient to determine that the image is not a face? One simple way to add more information to the classifier is to use more features that cover the image more thoroughly. This indeed helps but there is a limit to adding features as well. No matter how many features are added the error rate remains above 2%, as Figure 7. There are other methods to add information to the features which can be considered, but none make the errors disappear.

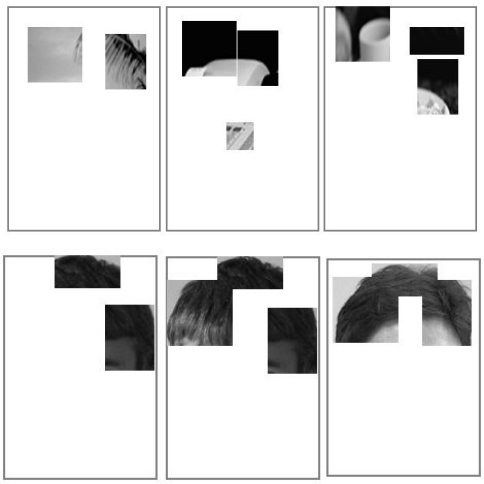


Figure 9: Detected features. On the top row - features found in non-class images that were classified as class. On the bottom row – we show features detected in class images that were ultimately rejected by the classifier. The eye fragments that are highly informative are missing, but would you fail to classify these images?

In Figure 10 we show the same features detected in the non-class images shown in Figure 9, but with the full image. Are the detected features too dissimilar from the fragments they represent? Note that when the full image is exposed, it is more prominent that the features detected are not very similar. One argument is that the similarity measure of the features is not good enough. However changing the feature similarity measure from Normalized Cross Correlation to SIFT [7], for example, changes the false positives a bit but doesn't change the problem (see Figure 6).

Looking into results of different methods like SIFT, Constellations, and others, we see that the error rate is approximately similar once it reaches the levels below 5%. All similarity measures have their shortcomings. The problem is inherent and may be connected to the reduction of the image dimensions, the locality of the features, the combination of the features into a single score and the size of the training set. In the following sections we propose some ideas of how the similarity can be changed to improve the final classification. Before that we consider how the features are combined into a score.

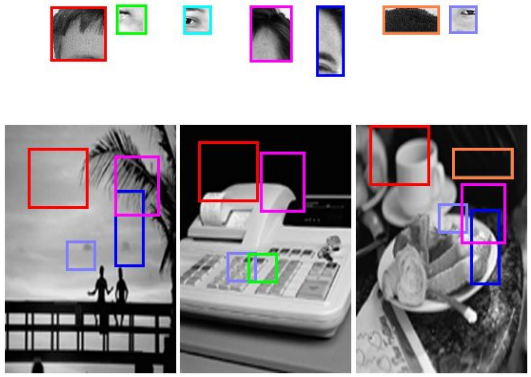


Figure 10: Detected Features in Context. The top row shows the features used for classification. The bottom row shows the features found in the misclassified images using the NCC similarity measure; the border color marks which feature was detected. Is the similarity sufficient? When seeing the 'full picture' the detected fragments seem quite wrong.

4.1.2 Improving the Combination of Features into a Score

The different classifiers we surveyed have been compared on many tasks and there is no algorithm that achieves better performance on all tasks and all settings; each task and algorithm need to be tailored to the necessary restrictions [23]. In a paper by Fleuret [11], empirical comparisons of face classification show that max-min feature selection (also referred to as CMIM - Conditional Mutual Information Maximization criterion) combined with SVM gives the best performance, a test error of 1.12%; however AdaBoost and max-min with Naive Bayesian methods are not very far behind with test errors of 1.45% and 1.52%, respectively. For our purposes, the main observation is that the basic difficulty of the unreasonable errors remains when varying the classification method or the feature selection method. We ran some comparisons on our data sets using fragments as binary features and observed similar results to Fleuret's. The work described in the following sections, does not try to create a new classification scheme, but rather combine existing classification methods to achieve better performance.

4.1.3 Using the Configuration to Improve Classification

Most classifiers we studied search for some finite set of features in a sample image and combine them into a score. Few, of which the constellation model by Fergus [10] is one, also generate some value for the configuration or relative location of the detected features. Although the configuration is another source of information, and in many false positive images the configuration is incorrect and can be used to reject the image, there are also false positives that have a reasonable configuration, and on the other hand, there is a variability in the configurations in the class images (see Figure 11). Since we assume the location is a good feature, and can add to the simple similarity, we tested this idea using SVM. We provided the SVM classifier with a vector of features that was composed from similarity and location of possible fragments, rather than just the similarity. We show a sample result in Figure 11 that does not display a large advantage to the location information. We did not proceed to check this direction, as it is similar in a way to Fergus' work [10]. We do believe there are other ways to integrate the location information into a classifier, and that it is useful to add information of more than just similarity. The receptive field of cells in visual brain areas is definitely an important component in the

processes that help the human visual perception achieve its goals. We leave this for future work section, and continue to discuss some competing ideas that we attempted.

4.1.4 Perspective Fragments

In this subsection I describe briefly the motivation for the perspective fragments and some experiments.

When taking a look at the fragments found in false alarm images it seems that the detected patches, when taken out of the context of the full image, may seem similar to the fragments (Figure 12 shows some examples). However, expanding the region in the vicinity of the detected fragment often reveals that it is not embedded in the expected local context. Therefore, the idea we tested was to add a 'perspective' to the fragments which adds some information that depends on the local context.

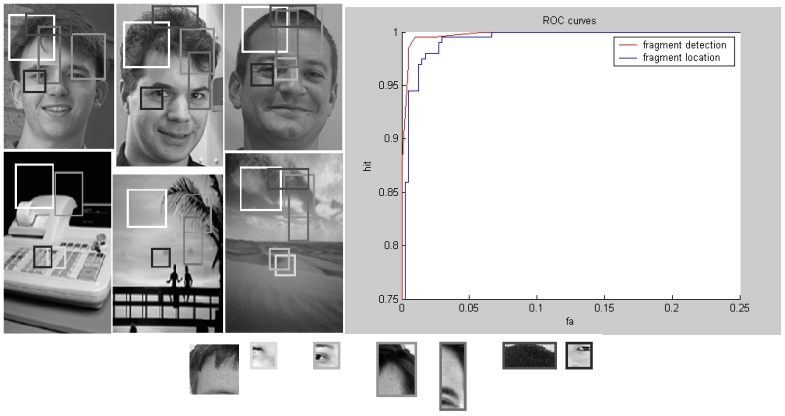


Figure 11: Using the Configuration. On the bottom we show 7 fragments used in this test. The left image shows some example of configurations detected in class and in non-class images. The configurations in the class are with high variability. On the right we compare using the fragment detection only for classification to using the fragment location. The data was combined here using an SVM linear classifier.

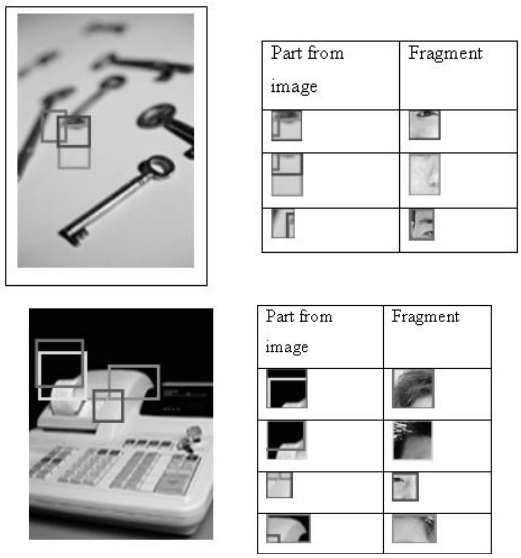


Figure 12: Features detected out of perspective. When looking at the fragments detected out of their image context the similarity is much more prominent to the naked eye.

Perspective fragments are fragments that *"put things in perspective"*. The algorithm tries to find for each feature, a larger encompassing feature that would give a context for the smaller fragment allowing verification of the smaller feature. For example, we verify a patch is really an eye by looking for the 'nose and eye' larger fragment with a lower threshold. Since searching for pairs of highly informative features is inefficient for combinatory reasons, we searched for informative features and then tried to improve them by finding a perspective for them. An extension, considering that sometimes the fragment itself is large, and may be a good perspective of some smaller feature inside, is to look for inner as well as outer perspective. For example, to verify an eye fragment is really an eye, find a smaller inner fragment with the iris. Figure 13 shows some fragment pairs.

The perspective fragments achieve better performance than normal fragments (see Figure 14). In the next section we present another method to improve performance using a different scheme - the anti-fragments.

Figure 13: Perspective Fragment Pairs.

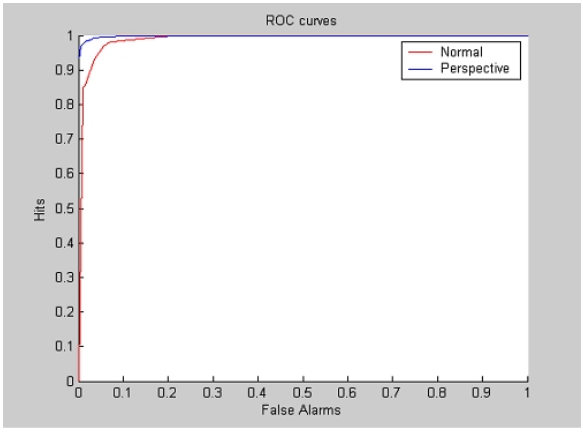


Figure 14: Perspective Fragments ROC

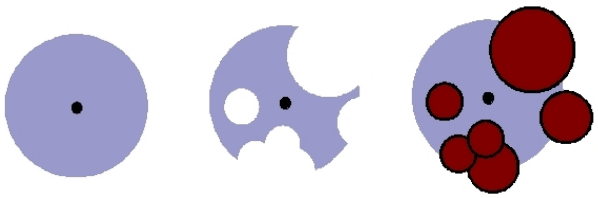
4.1.5 Anti-Fragments

We use the term 'anti-fragments' for features which are used as counter-evidence for the presence of a class, as opposed to supporting it. In this section we describe some of the motivation, biological and conceptual, for the anti-fragments idea, and then an algorithm for extracting and using them. We then show some results on the class of motorbikes.

4.1.5.1 Motivation and Algorithm

In most classification schemes, the features used are features that are found with high probability in the class and with less probability in the non-class. This is usually the case, since the possibilities for non-class are endless. However, even the best features are detected sometimes in non-class images in objects that resemble the feature, but are known to be very different from the feature. For example, a feature that is the corner of a motorbike wheel can be sometimes detected at the corner of a coffee mug. The round

element of the wheel is indeed similar; however, the coffee mug has a handle that distinguishes it from the wheel. Anti-fragments try to define the positive features, by using the errors made when the fragments are detected in the non-class images. For every fragment, we define a set of anti-patches that are combined with the fragment as a logical 'not', which can override the detection of the positive fragment. Figure 15 illustrates schematically what an anti-fragment is. A 'normal' positive fragment is a feature with a threshold, which means that all points above the threshold will be considered as detected instances of the feature. Adding inhibition patches with their own threshold can be viewed as creating a more flexible surrounding depending on the fragment as in Figure 15(b). Thus an anti-fragment is one that has a close similarity measure to some positive class feature and is far from similar from those non-class patches that are similar to the fragment. The algorithm uses false positive detections to generate the inhibition patches for a positive fragment and is detailed in appendix 8.3 ; we continue to show some results.



(a) (b) (c)

Figure 15: Illustrating an Anti-Fragment. (a) Depicts a normal fragment with some threshold, any patch that falls in the gray region is considered similar to the fragment. (b) Illustrates how the gray 'accepting' region is changed into some random, and possibly more accurate shape using anti-patches. (c) Shows the anti-patches (in red) that created the anti-fragment.

4.1.5.2 Experiment and Results

We run tests of the anti-fragments approach on the class of motorbikes. The motorbikes database is a hard data sets. It includes 800 images of motorbikes of size 220x180 pixels mostly cropped and scaled, and 500 non-class images of similar dimensions. The best fragments selected for the motorbikes are usually the wheels, since there is large class variability in all other details like chair, exhaust pipe, etc. This, of course, would not be the

case if we were learning to differentiate motorbikes from cars, but is the case in the described setting. Figure 16 shows the variability in the class. Some anti-fragments selected can be seen in Figure 17. Using the anti-fragments we found that we could get an improvement. However, a drawback is the over fitting to the non-class samples which means the classifier may not generalize too well. We show more results in Figure 18.

Figure 16: Motor Bike Database Variability. Some examples of different motorbikes in the database

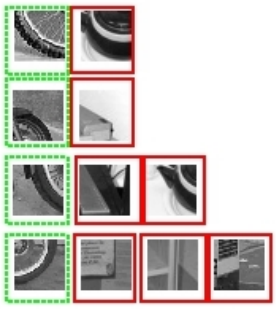


Figure 17: Sample Anti-Fragments from the Motor Bike database. Each line is an anti-fragment, the patches on the left are the fragment and the patches to right with the red border are the inhibitory anti-patches

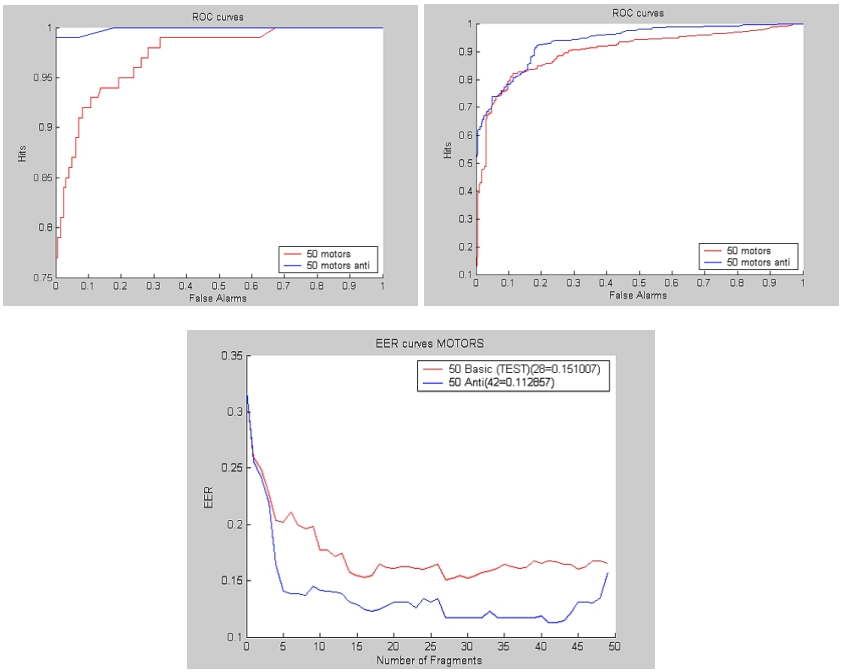


Figure 18: Anti Fragments Results. The ROC curves comparing performance on the motorbike task of normal fragments (red) to anti-fragments (blue). Top right is in the training set, top left is in the test set, and the bottom is the EER in the test set when adding fragments. We show the minimum EER and its location for each scheme.

The motivation for anti-fragments is derived from inhibition in the human visual system. The current scheme is implemented as a rigid 'all or nothing' inhibition, but smoother variations of inhibition can also be tested. Another attempt to improve classification is the satellite fragments as a second stage described in the next section.

4.1.6 Satellite Fragments as a Second Stage

Considering the false positives when classifying with a small set of fragments, we believe that searching for more specific features, that are relevant to the class, may help.

4.1.6.1 Algorithm and Motivation

The more specific features can rely on the information found at the initial stage, following the work of Epshtein and Ullman [16]. These additional features were chosen to be validation satellites. Similar to [16] the satellites are dependant on the detection of the

basic features, and are used only for images that are classified correctly in the first stage. For example, Figure 19 shows a detected patch of forehead and a patch of nose in two images, one is a real face and the other is not. The false image can be rejected if additional support at specific locations would be sought, since it is missing crucial parts - for example an eye below the forehead and adjacent to the nose.

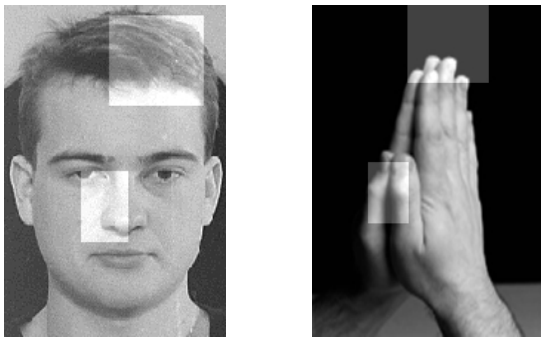


Figure 19: unreasonable false positives. Using the available features the following two images were classified as faces. If we could add a validation step to validate that below the forehead and a bit to the right of the nose there needs to be an eye, we could avoid making the mistake of classifying the hands as a face**.**

Similar to the satellite approach, a probabilistic geometric model is used that enables extracting more specific features (like the iris of the eye), that without a specific location would not be meaningful. Since the satellite algorithm works for similar classes, it was adapted for the general classification problem, a detailed description of the algorithm is provided in appendix 8.4 . In a sense the satellites are used to validate the initial classification.

4.1.6.2 Experiment and Results

The following results are from a database of eastern and western faces. The faces data set includes 1000 faces of size approximately 150x200 pixels roughly the same scale and orientation. In Figure 20 the model with the anchor fragments can be seen. Figure 21 shows the best satellite locations. It is interesting to note that the validation locations are selected in the eyes, mouth and nose ridge locations which are prominent and localized

features of the class. Each satellite was composed of a number of semantically equivalent fragments, some examples are shown in Figure 22.

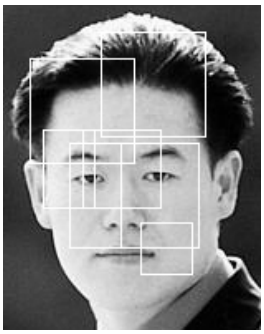


Figure 20: The Model Image. The white rectangles are the anchor fragments used for the first stage and for computing the model**.**

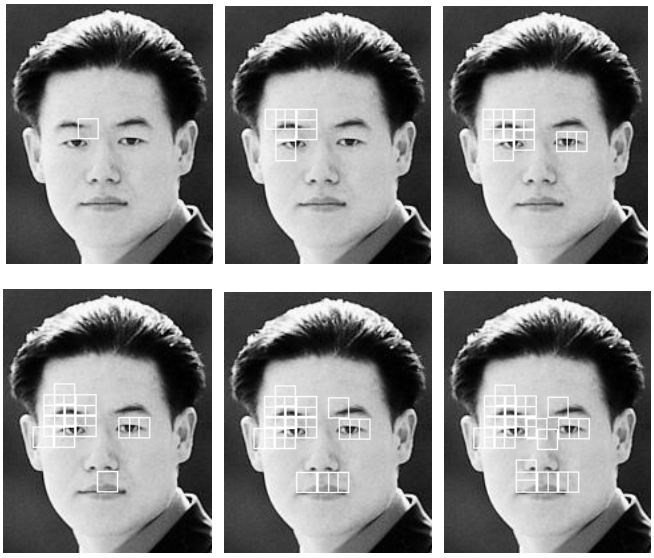


Figure 21: Validation Satellites. The images show the best satellites selected for validation by order.

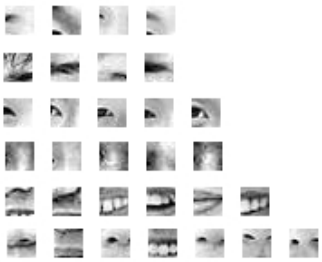


Figure 22: Satellite Semantics. Each row is a validation satellite with all the appearances selected. From top to bottom we have: end of left eye, right eyebrow, part of right eye close to nose, nose ridge, right side of mouth, nostrils**.**

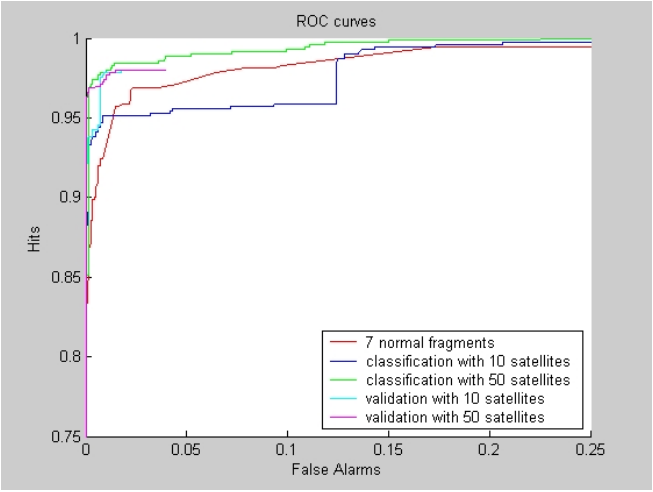


Figure 23: ROC curve comparing satellite classification. Validation here refers to applying the stages one after the other in a cascade.

In the ROCs displayed in Figure 23 an improvement for the satellites at low false alarm rates can be seen. However, since the satellite fragments are composed of many fragments, the naive comparison to only the anchor fragments is not enough. When comparing with many more fragments (approximately 7 for each satellite) the validation scheme looses its advantage. Moreover, a shortcoming of the scheme is that the initial phase uses a geometric model that has a large computational cost which also discourages using more than 7 fragments as anchors since the computational cost grows exponentially. However,

the idea of building a validation phase and using the data from previous stages remains as inspiration and will be discussed thoroughly in the next section. We now turn to discussing classification when the training set is not limited.

4.2 Classifying in the Setting of a Growing Training Set

In this section we focus on the notion that humans' superior classification performance is due in large to the fact that they continually learn over their life span. Unlike current computer systems, it is likely that humans have mechanisms that achieve quick and precise categorization in the framework of an ever growing training set. From the former part of our work, we conclude that any classification system that wishes to achieve human-level results at reasonable response rates needs to apply more elaborated computations to a part of the data, and to learn online from new examples. In section 4.2.1 below we show that adding training samples and enhancing the class representation by adding more features helps, as expected, to further lower the error rate. We conclude that the main avenue to improve performance to a human level is based primarily not on improving the learning from a fixed, small data set, but to continuously learn from a large data set, and to continuously extract useful features from the new examples. The main problems that arise are therefore, first, the ability to learn from a constantly growing set of samples, and second, to classify efficiently using a very rich set of features. We deal with these problems in subsequent sections.

4.2.1 The Training set Limit

As shown in the first part of this section, no matter how good the classifier is, when generalizing, the training set places a limit on the performance. Do humans have a limited set to learn from? Not really, and even though after a certain age, no one will tell a child, for example, that a pincher is a dog and not a mouse, humans still get many clues as to the fact that what they observe is a dog, even if they don't initially recognize it as a dog. Can the training set be one of the main differences between human classification performance and machine classification? It has been shown [23] that the training set theoretically limits the performance of any classifier. If this is the case, then perhaps the quest for human-

level performance should simulate a continuously growing training set, using appropriate online algorithms.

In this section we verify the effect of the training set size, by using fragments selected by either the max-min algorithm or by AdaBoost. As expected, adding images to the training set gives rise to a monotonic improvement in the learning and better performance. Additionally, when continuously augmenting the features selected as the training set grows, the performance improves as well, compared with using a fixed number of features but improving the selection over time. We can assume that the class representation therefore is enriched as the person gains experience with the class in question. We first describe some empirical tests and in the next section we suggest validation and online learning as means to cope with the training set limit.

4.2.1.1 Training set Experiments

In this experiment we tested the effect of the training set size on the classification performance. We ran the experiment on a database of horse images; we had 323 cropped horses in different poses and 450 non horse images of patterns and various scenes. We used max-min to select a set of up to 100 fragments that maximize the mutual information between the fragment and the class labels. At each stage we increased the number of images used for training. The test set was kept constant. In the results below, we used 40% of the images for the test set, and the other 60% were grouped randomly and added in six parts. We tested the effect of the training set size, as well as the maximal number of fragments extracted during training. For each training set, we added fragments and tested for the smallest equal error rate (EER) on the test set. The tests showed that the performance improves monotonically with the size of the training set, provided that the scheme is also allowed to increase the number of fragments used. We show the results averaged over 5 random training sets, and also a single representative run to see the number of images and fragments that the classifier learned from. When the pool of fragment is kept small, the effect of adding more training set images is limited, since the ability to select features that differentiate between the class and the non-class is limited by the set of features available at each step (Figure 24). When we allow the available fragment set to grow as the training set grows, the EER drops at each step, as shown in

Figure 25. Both the size of the pool that the features are selected from, and the final total number of fragments, increased the final performance.

We tested whether the effect of the additional train samples may be only due to the additional fragments, and not from the additional train images by themselves, by using a very large set of fragments (10,000) from the start, and keeping it constant at each step, see Figure 26. The experiments showed that the increase in the training set has an effect when the number of fragments is large and fixed.

We also tested the same scheme using AdaBoost, where each possible fragment was considered a weak classifier. The results were quite similar, except that AdaBoost, being a wrapper type classifier [15] had a zero error rate on the training set and a lower error rate on the test set. The effect of adding train samples and possible features remained the same as when using Naïve Bayes, see Figure 27.

We conclude that the error declines as a function of using more training set samples, and also as a function of having a larger pool of fragments to choose from, and using an increased number of fragments as the training set increases. We suggest that a method that attempts to imitate the human brain should be an online method, as the human brain continuously meets new examples and probably continuously improves class representations. We next turn to discuss validation and online learning; two methods that may help improve performance further, in the context of a growing set of features and a growing training set respectively.

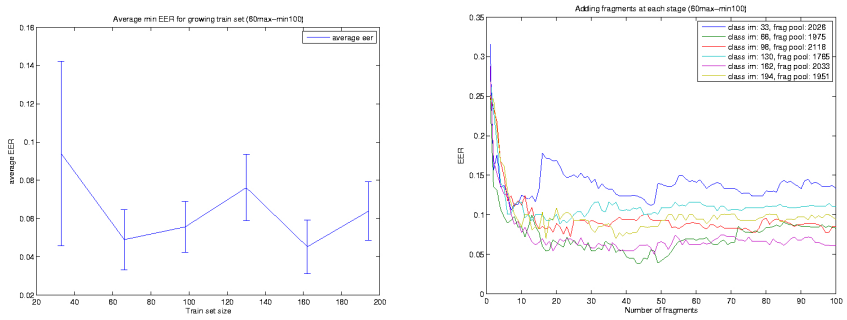


Figure 24: Increasing the training set size. As we increase the training set size, but limit the initial pool of features available for the classifier (around 2000 fragments) we can

hardly see any improvement. Averaging over 5 runs on the left, on the right we show single runs.

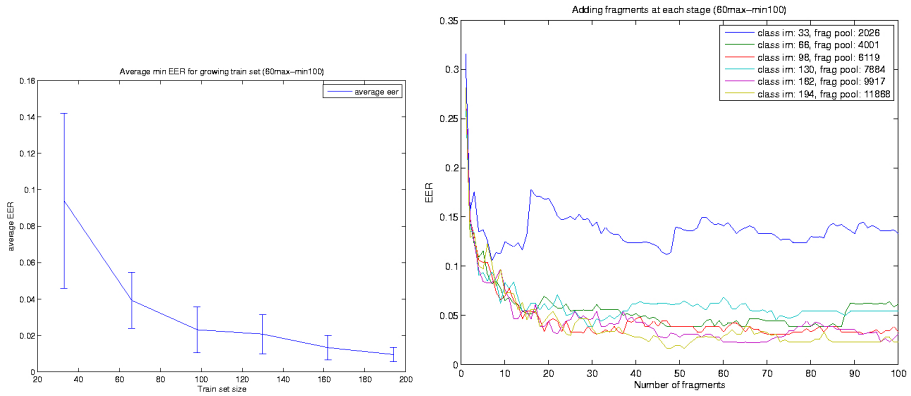


Figure 25: Increasing the training set and the fragment pool size. Naive Bayes classifier, as the training set grows, so does the pool of fragments. In (a) we can see the average EER decreases. In (b) we can see the results of single runs.

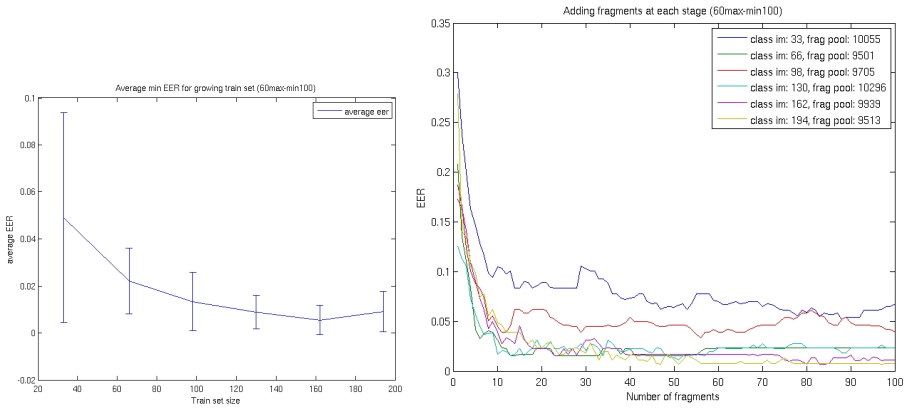


Figure 26: Increasing the training set with a large pool of fragments. The increase in performance is not only due to the large pool of fragments, since if we use a large pool of fragment to begin with, adding training set images improves the performance as well.

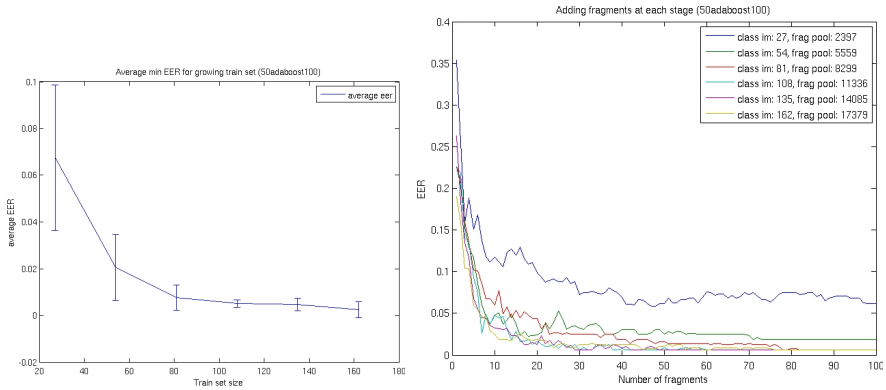


Figure 27: Increasing the training set and the fragment pool using AdaBoost. This shows that when the classifier used is AdaBoost the final performance is better, yet the effect of using more fragments and more train samples remains.

5 Cascades in Classification

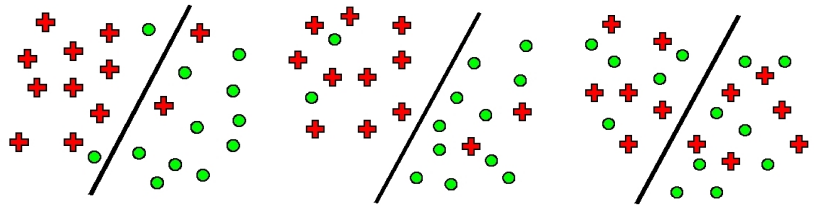
In the previous section we concluded that to achieve high performance, the learning state should include a large training set and be allowed to increase the number of features used in classification. This raises two main requirements from the classification scheme. First, to deal with a large number of training examples. The natural approach is to use on-line learning, which continues to learn from new examples, rather than use a limited training set in an initial training phase. Second, because the feature size grows, the computational load increases. To deal with this problem we develop multi-stage schemes. In such schemes, the number of features used in the first stage is limited, and additional features are applied only to a limited number of images, based on the results of previous stages.

We developed for this purpose two alternative multi-stage methods. The first is called a 3-way cascade. This is an extension of standard 2-way cascades used in the past and will be described in section 5.1 . The second method is called ‘configuration cascade’ and will be discussed in section 5.2 . It identifies specific feature configurations which are the source of errors and deals with them in subsequent stages.

5.1 Three-Way Cascades

Humans tend to make quick responses in physiological experiments to typical class instances, but typically show a longer response as well as higher error rate to harder examples. One hypothesis is that difficult instances require more processing than simple ones. If we consider the fragment approach, we can see that for horses, a ten-fragment classifier already achieves pretty good performance, at a cost of searching for only ten fragments. The benefit of adding fragments is initially very large, whereas the benefit of adding more fragments when the classifier error is already low is relatively low, and the cost (the number of fragments to search) becomes much higher. To deal with this difficulty we suggest classifying in stages, similar to previous cascade schemes, but extending the approach to a so-called 3-way cascade. In a 2-way cascade, the first stage is used only to reject some of the inputs. We use the first stage to either reject or accept some of the input.

To explain the use of a 3-way cascade, we define below a *strong* classifier as a classifier whose errors are close to the margin, Figure 28 shows schematically what a strong classifier is, and Figure 29 shows a sample response for a strong classifier.



(a) (b) (c)

Figure 28: The Strong Classifier Concept. The definition of a strong classifier is schematically shown here. In (a) we see a strong classifier, where the errors are close to the border. In (b) we see a classifier that makes the same number of errors but is not a strong classifier, since the errors are not close to the border; (c) shows a weak classifier.

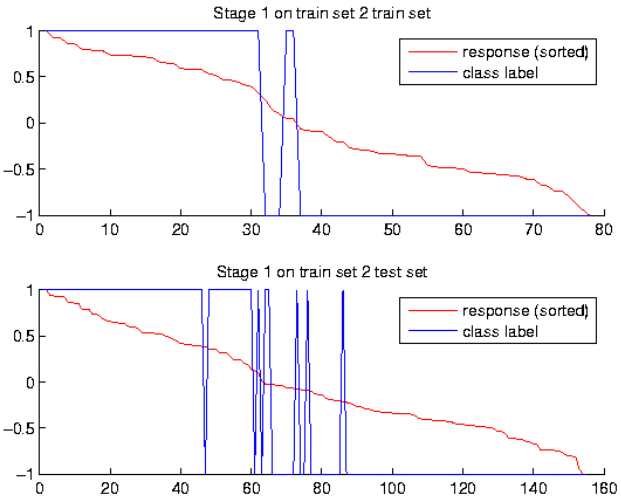


Figure 29: Strong Classifier Response. The red line is the response of the classifier sorted from highest to lowest, the top part is in the training set and the bottom is in the test set. The blue line is 1 when the image is class and -1 when it is non-class so the vertical lines are the errors. The errors are distributed around the classifiers medium response, and we can see that the classifier is a strong classifier as there are no errors in the region of highest and lowest responses.

The three-way cascade classifier works in stages similar to a standard 2-way cascade [5]. At each stage a strong classifier is applied to all samples that require further processing. Initially, all images require processing, the improvement is that if the classifier we use is a strong classifier, those samples that had a sufficiently high response are classified as class and will not be processed in additional stages, and those samples that had a very low response will be classified as non-class and will also not require any more processing. Only those images that are in the middle zone in terms of response strength will be further processed in the next stage.

In some cases the three way cascade scheme, not only enables cutting computational costs but also allows reaching lower error rates. We discussed in previous section that to improve the performance of a classifier the classifier needs to learn more and to process many features. Cascades allow adding more features online, while keeping reasonable computation cost on the average. The three-way cascade scheme may achieve better