computer graphics & vision

Inffeldgasse 16/2, A-8010 Graz

www.icg.tu-graz.ac.at

TUG

Embeddings for Random Ferns Classification

Markus Oberweger, BSc

Institute for Computer Graphics and Vision
Graz University of Technology, Austria

Supervisor: Dipl.–Ing. Dr.techn. Michael Donoser

Master's in Telematics F 066 411
Master's Thesis
Graz, May 4, 2014

contact: Markus Oberweger markus.oberweger@student.tugraz.at

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)

Abstract

Efficient multi-class machine learning methods are a key component in many computer vision applications. In this area the Random Forest (RFo) classifier is considered state-of-the-art as it provides an efficient way to learn an inherently multi-class classifier, that naturally handles high dimensional, multi-modal data. Recent research focused on optimizing this classifier for different applications, by learning appropriate node split criteria. In this work we present a different scheme for acquiring these node splits. By selecting feature subsets and using an ensemble of weak classifiers, we propose different linear subspace and ordinal embeddings to derive flat classifiers for leaf assignment, similar to the popular Random Ferns (RFe) classifier. Therefore we use a generic framework into which we integrate all of our proposed embeddings.
We evaluate our classifiers on several machine learning benchmark datasets, as well as on well-known computer vision datasets. We show, that our classifiers can outperform conventional RFe and even RFo without significantly increasing computational costs. Further, we show the applicability of our classifier for the task of planar object tracking, as well as 3D interest point recognition.

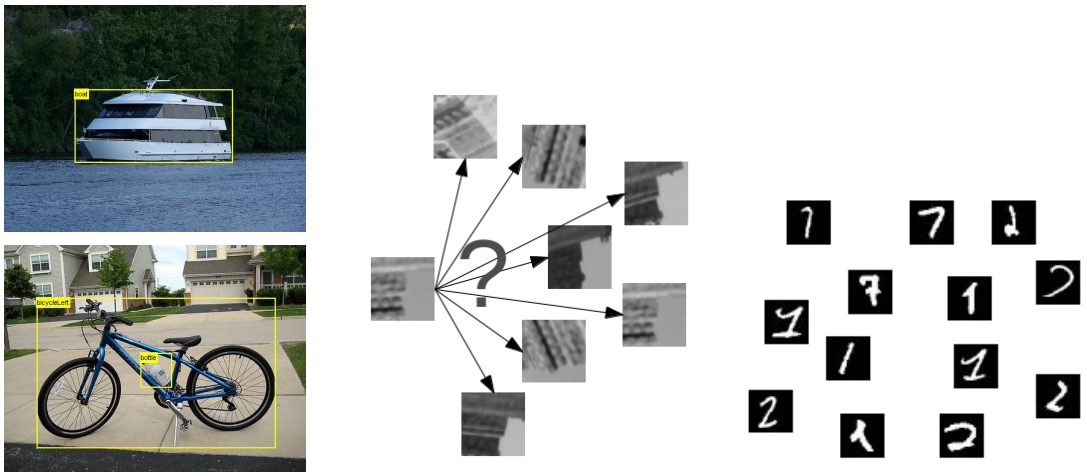
Keywords: *Random Ferns, Random Forest, classification, computer vision, machine learning, embedding, subspace, ordinal, WTA hash, CRT, oblique, PCA, CCA, LDA*

Contents

1 Introduction

Computer vision is a rapidly evolving field which is getting increasing attention in industrial and consumer applications. In the last years there has been a vast improvement in the imaging technology by increasing resolution, frame rates, and image quality, thus leading to higher data rates and the need for more sophisticated image processing algorithms that can handle this amount of data. Computer vision applications are also pushing into the wearable devices segment, which have considerably smaller computational resources than current personal computers. This makes it necessary to adapt and develop specialized methods that can cope with this restricted resources.

One of the tasks which is commonly required in computer vision applications, and also has a huge researching community, is classification [8, 17, 72, 81, 84]. The applications are wide spread and some examples are shown in Figure 1. Classifiers can be used, e.g. in object detection [111], where an image is given and one should name and localize the object(s) which are present (recognition), or to solve the correspondence problem by interpreting interest point recognition as classification [65], or in digit recognition [73]. In classification, we recognize the type or category y of some input data $\mathbf{x}$ and assign it to a specific discrete, categorical class label among H classes such that $g : \mathbf{x} \rightarrow y$ and $y \in \{1, \dots H\}$.



(a) Object detection. [21] (b) Interest point recognition. (c) Digit recognition: 1, 2, 7.

Figure 1: Different example applications of classifiers in computer vision.

The classification problem is getting increasingly complex with larger, more realistic datasets, e.g. Pascal VOC [21], which contain hundreds or thousands of object classes, and highly complex image or feature representations, such as the Scale-Invariant Feature Transform (SIFT) [71] descriptor with 128 dimensions or the Histogram of Oriented Gradients (HOG) [16] descriptor with 3780[1] dimensions. This increasing computational complexity demands more efficient classifiers, which on the one hand can cope with the more complex data, and on the other hand are computationally efficient such that they can run on mobile devices with restricted computational power.

[1]Example dimensionality calculated for 7 blocks, 15 cells per block, and 36 bins per cell, thus $7 \cdot 15 \cdot 36 = 3780$ dimensions. [16]

For such high dimensional features the so called *curse of dimensionality* makes it computationally difficult to select relevant features, representations or combinations for the classification. The optimal choice of features and their combination is a NP-hard problem [63, 6]. This problem can be circumvented by using an ensemble of base classifiers, where each classifier uses a different set of randomly selected features [41].

One of the most popular ensemble classifiers is the Random Forest (RFo) [9]. It trains an ensemble of decision trees by selecting subsets of features for each weak classifier and finding optimal node splits. The inference of the optimal node splits is a computationally expensive task, thus the so called Extremely Randomized Trees [33] were introduced, that allow faster training by selecting random node splits. Still, this lightweight implementation has the hierarchical structure of the decision trees that limits the classification speed. In order to mitigate this problem, the Random Ferns (RFe) principle was introduced, that allows even faster training and classification by using a flat structure instead of the hierarchical decision trees.

Proposed by Özuysal et al. [84] the RFe classifier is a simple, inherently multi-class and randomly trained classifier. Due to its relation to the well-known RFo classifier, the RFe share most of its advantages. They include inherent support of multiple classes, robustness against label noise, robust to overfitting to the training data, and natural handling of high dimensional features. Further, the RFe has a reduced computational complexity compared to the RFo classifier at a comparable classification performance, which enables faster classification. It has been shown to run efficiently on mobile devices, see e.g. [116]. RFo, as well as RFe, are of special interest for (near) real-time applications with complex inputs, as they can be easily parallelized and very fast implementations for Graphics Processing Unit (GPU) exist [101].

Since the introduction of the RFe, there have been many applications taken advantage of this classifier. The applications range from interest point recognition [84, 70, 116], over general machine learning [58, 20], to object detection [111, 113, 114], camera stabilization [49], action recognition [81], online learning [112, 89], etc.

Besides the big advantages of the RFo principle, these methods use feature representations, that only allow for axially parallel node splits. Thus the use of different embeddings for the RFo was proposed [74], in order to allow non-axially aligned splits and to archive better classification results. While these possibilities have been investigated for RFo extensively, it has not gained any interest in the RFe framework yet.

By employing a subspace projection of the original input data using a certain projection method, the multi-class data can be more effectively separated, and thus higher classification performance is achievable. In the RFo framework embeddings or subspaces can be determined by iteratively searching for optimal subspaces at each node. This does not integrate well in the RFe framework, as we want to determine the subspaces instantaneous to keep the runtime low. Although it would be possible to select optimal leaf assignments (instead of node splits due to the flat structure) this would considerably reduce the performance.

Thus we investigate different embedding methods for the RFe framework. Besides linear subspaces, we also evaluate ordinal embeddings, that do not only encode the feature values, but also the different feature relations with their spatial position within the feature vector. This approach promises invariance against different linear perturbations [18], as well as stronger base classifiers for a better classification.

In this work we propose highly efficient classifiers based on the random ensemble principle. Therefore we present and evaluate different embeddings for the RFe classifier. Our proposed subspace embeddings can adapt to the data properties in order to induce a higher commonality within the leaves and thus increase the classification accuracy. Further our ordinal embeddings only use the order of the input data and thus provide a high tolerance to a perturbation of the input data. We show, that all of our embeddings only increase the runtime of the classification marginally. Our classifiers provide the same advantageous properties of the RFe, as inherent multi-class capability, handling of high dimensional data, tolerance to noise and fast runtime.
We demonstrate the performance of our approach using several machine learning and computer vision datasets, which amongst others deal with specific computer vision problems, as e.g. image classification or face recognition. Further, we show the applicability of our classifiers for solving correspondence problems for the applications of planar object tracking and 3D interest point recognition.

Please note, that in machine learning the term *ordinal classification* is differently used for methods, where the objective is to predict labels in an ordinal scale, see e.g. [50, 12, 38] for an overview. Given an input vector of observations $\mathbf{x}$ the objective is to predict the label y such that $g : \mathbf{x} \rightarrow y$, where a natural order in the output label space exists. For example $y \in \{never, sometimes, often\}$ would imply such an order with $y_1 \prec y_2 \prec y_3$ where $\prec$ denotes the order. In contrast we do not infer the ordinal labels but we use the order of the input data for classification.

Throughout the thesis we use matrix and vector notations extensively, thus we define them first for clarity. We assume a D-dimensional vector to be a row-vector denoted by a small boldface letter $\mathbf{x} = [x_1, \dots x_D] \in \mathbb{R}^{1 \times D}$ and the subscript indicates the dimension. A matrix is denoted by a boldface capital letter $\mathbf{M} \in \mathbb{R}^{R \times C}$ where R is the number of rows and C the number of columns, respectively. We use superscripts to refer to the individual elements $\mathbf{M} = [\mathbf{M}^1, \dots \mathbf{M}^C]$.

2 Related work

The foundations of this work are provided by the Random Ferns (RFe) framework proposed by Özuysal et al. [84]. Primarily, the classifier was proposed for interest point recognition in tracking applications for which the correspondence problem of one interest point to the learned prototypes can by formulated as a classification problem [65]. They have shown, that the RFe can outperform the RFo in terms of accuracy and speed for planar tracking applications. For the RFe an ensemble of flat classifiers called *ferns* are trained by selecting randomly chosen features, and the consensus of the ensemble is used for predicting the class. A Semi-Naive Bayesian approach is used to estimate the posterior probabilities and to combine the predictions from the weak classifiers within the ensemble. For interest point recognition each fern is trained by a set of features, where each feature is derived from a comparison of two pixel values. The RFe are trained using a huge set of artificially created training samples. Since its publication many extensions and further applications have been proposed.

Next to the RFe, Williams et al. [119] simultaneously proposed a related classifier, the *Randomized Lists* for the purpose of interest point recognition in the context of Simultaneous Localization And Mapping (SLAM). Similarly, they use a sequence of binary tests to classify interest points, but they only store binarized leaf scores and for each interest point they evaluate multiple class hypotheses instead of the one with maximum posterior probability. Further, they use an intensity offset for the feature computation to generate more stable features in monotone areas where only noise is measured, and they explicitly handle noise during training by discarding features that are close to a noise threshold.

Villamizar et al. [111] proposed to apply boosting and bootstrapping, common principles borrowed from the RFo classifier, to extend the RFe in order to improve the performance for object classification, especially if the number of training samples is low. Therefore they train their boosted classifiers and evaluate them on a bootstrapped sample set. Then they retrain the classifiers with new warped samples of the classified and misclassified samples. This allows them to train their classifier with only a few samples at classification rates which are obtained by training with more samples, but their approach requires an extended training phase.

The memory consumption can be a bottleneck for the usage of RFe in mobile applications. Therefore Lee et al. [62] proposed a method which significantly reduces the memory footprint of the classifier. They do not store the full posterior probabilities in the leaves, but only binarized probabilities, which are derived by thresholding the original posterior distribution. Thus the memory usage is reduced by a factor of 32 compared to the original memory requirement, but with a loss of classification accuracy.

Fragoso et al. [29] and Gao et al. [31] proposed methods for efficiently selecting features. As the features for the RFe are derived from pixel comparison, they both work on the selection of locations for the feature comparison. Fragoso et al. use non-cooperative game theory to produce unique descriptors per patch. One player tries to maximize his payoff by selecting pixel pairs that exhibit high variations within the patches, while the other player wants to minimize the payoff by selecting different patch pairs. They show a higher recognition rate and better rotational invariance, at the cost of solving the game in a costly training phase. In contrast to this, Gao et al. first create a subset of feature

locations with high discrimination capability from a large, random candidate set. Among these discriminative features they estimate the dependencies. A group of features that show similar dependencies is then used to create multiple ferns. This procedure is repeated for several groups of features, thus forming multiple "layers". They achieve a more accurate and robust classifier without speed loss, though they require a more complex training.

Wagner et al. [116] used an adapted version of the RFe for natural feature tracking and pose estimation on mobile phones. In order to reduce the memory usage they reduce the number of features per fern and further discretize the posterior probabilities to bytes in the range of $[0, 255]$. As this two methods considerably limit the classification performance, they add rotational invariance to the ferns by using the dominant gradient orientation. Further, they do not use the correspondence with the maximum class probability, but the top ranked as putative matches for calculating the pose, and they apply a sophisticated outlier removal.

Liu et al. [70] proposed an extension to the RFe for homography estimation, where they use the interest points matched by the RFe and the calculated homography to alternatively refine each other. Therefore they use an assignment matrix from the detected to the learned interest points which is iteratively optimized using energy minimization techniques. They can improve the accuracy of the homography estimation but the iterative optimization is a computationally costly operation.

Although the RFe is commonly used for interest point recognition, Kursa [58] proposed the usage of RFe for general purpose machine learning. Therefore he generalized the feature representation and introduced a feature importance measure, similar to RFo. The classification results of his proposed ferns are comparable to RFo but at higher classification and training speed especially for large datasets.

Bosch et al. [8] introduced RFe for image classification. They combine shape and appearance representations with spatial pyramid matching in order to derive a descriptor for different regions of interest of an image. For classifying these descriptors they evaluated RFe and RFo classifiers, where they use random subspace projections with random thresholds for the node tests for both. The information gain criterion is used to select the best node test for each node from a set of random tests. They have shown, that for this task the RFe and RFo perform similar to Support Vector Machine (SVM) but at improved speed.

The RFe classifier is related to the RFo classifier of Breiman [9]. He trains an ensemble of decision trees and combines the result of each tree to form the final prediction. Each tree is trained on a bootstrapped subset of the training data and the rest of the samples is used to estimate the generalization error of the tree. A tree is trained top-down by choosing random feature dimensions for the node splits and the optimal split is derived by minimizing an impurity measure, e.g. Gini gain. For prediction a sample is pushed down each tree and the ensemble consensus is used as result. While Breiman uses bootstrapping and bagging, Ho [41] uses random subspaces to train each tree, which is more similar to the RFe principle.

For the related RFo classifier, different subspace methods have been intensively studied. In the context of RFo they are called *oblique* due to their non-axis aligned decision boundaries [74]. For the induction of optimal hyperplanes that define the oblique splits, there are many works that propose different learning algorithms to derive optimal sub-

space projections for oblique RFo. For example Murthy et al. [78] use a randomized hill-climbing algorithm that derives a hyperplane for each node split, that optimally separates the data. Their oblique trees are smaller and more accurate than the axially parallel node splits for domains with numeric attributes. Similarly, Menze et al. [74] proposed different subspaces to train RFo. They compare random subspace projections, Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and ridge regression to obtain a linear subspace for the node splits and analyze the different subspaces for the application on different data types. They identified that RFo with orthogonal (axially parallel) node splits work best for data with discrete features, RFo with random projections for numeric data and their proposed ridge regression and LDA subspace worked best for highly correlated, high dimensional data.

While subspace embeddings try to find efficient subspaces, that have certain properties that make classification easier, ordinal embeddings try to add additional invariance into the classification process. Thus several works were published that use rankings or ordinal information for interest point descriptors. Wang et al. [118] proposed a descriptor which is invariant to monotonic intensity changes. They first divide the image patches into regions based on the intensity order. Further, they extract the rank of the intensity values sampled from a pixel neighborhood to build a histogram of rankings to form an interest point descriptor which outperforms many other descriptors. Similarly, Tang et al. [108] proposed an ordinal preprocessing of the image patches. Therefore they group pixels with similar ordinal range intensities to achieve invariance to pixel noise and monotonic light changes.

In order to incorporate ordinal information, Yagnik et al. [123] proposed a hashing method, the so called Winner Take All (WTA) hash, which derives a hash code from a feature vector while correlating the Hamming distance to rank similarity measures. Therefore the sample vector is randomly permuted multiple times and the first K dimensions are used to determine the index with the largest value. This results in a hash code whose dimensionality can be parametrized. They used a binary version of the hash code with $K = 2$ for similarity search in high dimensional space and showed that they outperform conventional descriptors and machine learning methods. Further, the WTA hash was successfully used for embedding HOG filter responses in object detection [17] due to its invariance to numeric perturbations and its ability for efficient hash generation.

Schulter et al. [98] proposed a method related to the WTA hash, which uses ordinal node splits for RFo. For their splitting function they randomly sample the input vector and if the dimension of the maximum value matches a defined node-specific constant, the sample is assigned to the left child node, otherwise to the right child node. This makes the commonality of data in one branch higher, thus strengthening the splitting functions. All samples within the child node fulfill multiple inequalities instead of a single. They showed, that the Ordinal RFo can outperform conventional RFo for digit recognition and object detection.

Another method for incorporating ordinal features is to use the order of the input data directly. Therefore Demetz et al. [18] proposed the Complete Rank Transform (CRT), which is a specialized approach of partial order. The CRT transforms a pixel neighborhood into a vector, where each element encodes the relative order within the neighborhood. They used the invariance to grey-scale rescaling of the descriptor for robust optical flow calculation.

To sum up, there have been many improvements and adoptions published, where most of the related work focuses on optimization of the feature selection process, or on different representations of the posterior probabilities, e.g. to save memory. There could not be found any work that examines efficient embeddings and subspace methods for RFe, which is in contrast to RFo, for which these methods have been evaluated. This reason motivates for an evaluation of different embeddings in the context of RFe.

3 Embeddings for Random Ferns

In this work we focus on embeddings for Random Ferns (RFe) classifier and we propose different embedding methods. Therefore we first describe and generalize the underlying RFe framework which is the foundation of our embeddings, and further outline the evaluated embeddings. We formulate our proposed classifiers as embeddings for a generalized RFe classifier, although they could be described as a new kind of ensemble classifier based on a Semi-Naive Bayesian approach as well.

We propose different embeddings where we first present subspace embeddings derived from well-known linear dimension reduction methods, as e.g. Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), or Canonical Correlation Analysis (CCA). Further, we describe different ordinal embeddings which are based on the relative order of different input dimensions. At last we discuss a combination of subspace embedding and ordinal analysis, where we first project the data into a subspace in which an ordinal analysis is performed.

3.1 Random Ferns framework

In this work we use the RFe framework for multi-class classification. Formally, in classification we want to predict the discrete categorical class label among H classes $y \in Y = \{1, \dots H\}$ of an input sample $\mathbf{x} \in \mathbb{R}^D$ where D is the number of features (dimensionality) of the input sample, thus $g : \mathbf{x} \to y$. We perform supervised classification, where a training set with N samples and their class labels $(\mathbf{x}^i, y^i), i = 1 \dots N$ is provided in advance.

The RFe classifier is an ensemble classifier, that consists of multiple weak classifiers, the so called *ferns*. A fern is a flat structure, where at each level a binary test is performed on randomly chosen feature dimensions, in contrast to the related hierarchical binary decision tree. The RFe classifier can be derived from the well-known RFo classifier of Breiman [9] by using the same node split criterion at each tree level and thus transforming the hierarchical structure of the tree into the flat structure of the fern, which is depicted in Figure 2.

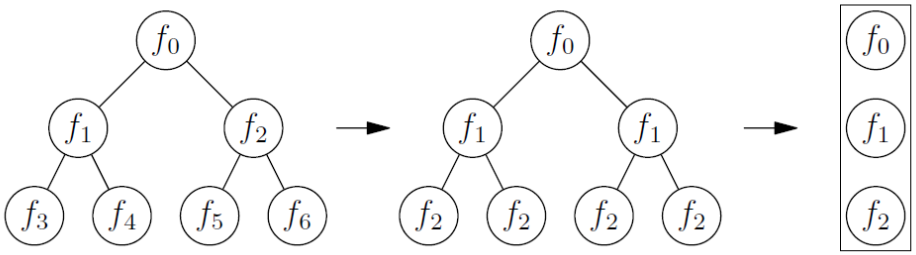


Figure 2: Correspondence of a decision tree to a fern. [83]

The classification using the RFe can be derived within a Naive Bayesian framework, which was shown by Özuysal et al. [83]. Given a set of C features f_c, which are derived from the input sample, we want to find the class $\hat{y}$ that maximizes the posterior probability

over all possible classes Y such that

$$\hat{y} = \arg\max_{y^i} p(Y = y^i | f_1, f_2, \ldots, f_C). \tag{1}$$

Using Bayes' Formula the conditional probability can be reformulated as

$$\begin{aligned} p(Y = y^i | f_i, f_2, \ldots, f_C) &= \frac{p(f_1, f_2, \ldots, f_C | Y = y^i) p(Y = y^i)}{p(f_1, f_2, \ldots, f_C)} \\ &\propto p(f_1, f_2, \ldots, f_C | Y = y^i). \end{aligned} \tag{2}$$

In this work we assume the class probability $p(Y)$ uniform, and the denominator is independent of the classes which resembles a scaling factor that can be neglected. Thus we can simplify

$$\hat{y} = \arg\max_{y^i} p(f_1, f_2, \ldots, f_C | Y = y^i). \tag{3}$$

Due to the possibly high number of features it is not feasible to store all 2^C joint probabilities, thus by assuming independence between features we can rewrite the joint probabilities as

$$p(f_1, f_2, \ldots, f_C | Y = y^i) = \prod_{j=1}^{C} p(f_j | Y = y^i), \tag{4}$$

which resembles a Naive Bayesian model [4].
However, complete independence is a very extreme formulation that can be relaxed by grouping features to $S = \frac{C}{M}$ groups, which are called *ferns* and model the joint probability between the features. M denotes the number of ferns $\mathbf{F}^m$ where the whole ensemble is formed by $\mathcal{F} = \{\mathbf{F}^1, \mathbf{F}^2, \ldots \mathbf{F}^M\}$. Each fern evaluates S binary features $\mathbf{F}^m = [f_1^m, f_2^m, \ldots f_S^m]$ leading to $L = 2^S$ leaf nodes $\mathcal{L}_l^m$ per fern and $l \in \{1, 2, \ldots L\}$. Thus using the joint probabilities of a fern $\mathbf{F}^m$ the conditional probabilities are

$$p(f_1, f_2, \ldots, f_C | Y = y^i) = p(\mathcal{F} | Y = y^i) = \prod_{m=1}^{M} p(\mathbf{F}^m | Y = y^i). \tag{5}$$

This is equivalent to a Semi-Naive Bayesian [126] approach, which models only some of the joint probabilities.
The final classification rule is then stated as

$$\hat{y} = \arg\max_{y^i} \prod_{m=1}^{M} p(\mathbf{F}^m | Y = y^i) \tag{6}$$

or more convenient by logarithmizing as

$$\hat{y} = \arg\max_{y^i} \sum_{m=1}^{M} \log\big(p(\mathbf{F}^m | Y = y^i)\big), \tag{7}$$

thus mitigating numerical problems in the multiplication of the individual probabilities.

An essential part of this formulation is the calculation of appropriate binary features f_i. In the original work [84] the feature evaluation method f_s^m is stated as

$$f_s^m = \begin{cases} 1, & \text{if } \mathbf{x}(d_{s,1}^m) < \mathbf{x}(d_{s,2}^m) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where $\mathbf{x}(d_s^m)$ is the sample value of the dimension d_s^m. By choosing two random dimensions d_1^m and d_2^m the comparison is invariant to any linear transformation of the sample $\mathbf{x}$.
A more general formulation for the feature evaluation [58, 20] can be stated as

$$f_s^m = \begin{cases} 1, & \text{if } \mathbf{x}(d_s^m) < \tau_s^m \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where the sample value $\mathbf{x}(d_s^m)$ is compared to a random feature-specific threshold τ_s^m. This formulation is more flexible as it does not require the input $\mathbf{x}$ to be drawn from a single feature scale, as e.g. an image with each pixel in the range $[0, 255]$. τ_s^m can be chosen in different way, as e.g. randomly, fixed, or the median value over all samples can be used to generate a threshold that splits the training set.
S such feature evaluations are used to form a binary vector $\mathbf{F}^m = [f_1^m, f_2^m, \dots f_S^m]$ which is then used to assign each sample to the corresponding leaf $\mathcal{L}_l^m$ with $l = \sum_{i=1}^{S} 2^{i-1} f_i^m$.

This feature generation and evaluation process can be seen in Figure 3. Based on the input vector $\mathbf{x}$ random dimensions are selected $\mathbb{R}^D \rightarrow \mathbb{R}^T$ and from them the features are calculated. In the original RFe framework [84] a random subspace $\mathbb{R}^T$ is not explicitly defined but implicitly by the feature selection vector $\mathbf{d}^m$. The features, i.e. the binary vector, are then used for leaf assignment. Each leaf stores a trained distribution $p(\mathbf{F}^m|Y)$ over $Y \in \{1, \dots H\}$ classes. Thus the required memory for storing the probabilities depends on both, the number of classes H and the number of leaves L, which is $\mathcal{O}(MHL)$[2] for M ferns.

We generalize this feature generation step further and do not limit the leaf assignment on the generation of binary features, but on an arbitrary assignment of the input data to the leaves. Therefore we formulate the leaf assignment as general mapping $\phi : \mathbf{x} \rightarrow \mathcal{L}$ that does not imply any input data properties. Thus not only binary features can be integrated into the RFe framework, but any arbitrary feature representation as we further show. The mapping ϕ has to be a bijective function (an injective non-surjective function would also work, but is non optimal in terms of memory requirements). Thus we use the RFe framework only for combining the base classifier results.

The training procedure for the RFe classifier is shown in Algorithm 1. Instead of training the probabilities, which is done by e.g. [20, 84], we only train the number of samples per leaf. Therefore we use a frequency matrix [57], which counts the occurrences of the samples for each leaf, in order to estimate the probabilities $p(\mathbf{F}^m|Y)$. For the frequency matrix we use a sparse matrix representation[3] to circumvent the memory limitation. Thus the new memory requirement is $\mathcal{O}(M \min(N, LH))$. This is especially

[2] $\mathcal{O}(\cdot)$ denotes the big O notation, which is used to express and compare the computational complexity of an algorithm. [56]

[3] Sparse matrices are supported by many software packages and can be created e.g. in Matlab using the `sparse` function or in OpenCV using the `SparseMat` class.

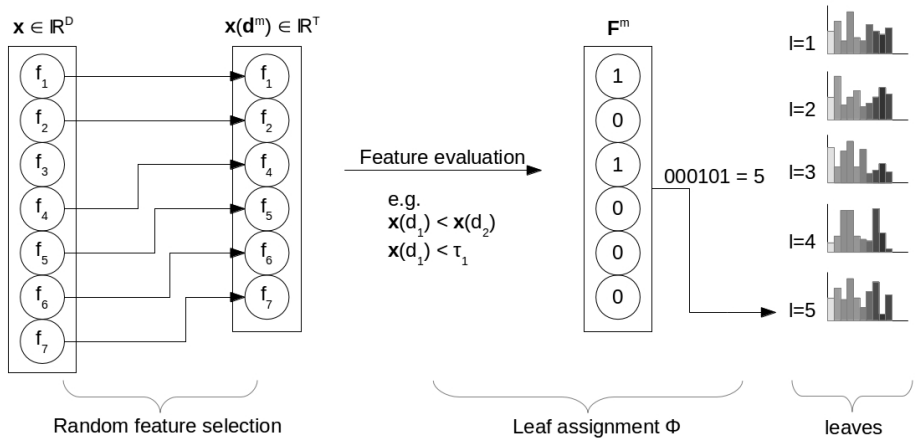


Figure 3: Feature evaluation of a single fern. By evaluating the input features of the fern, each sample can be assigned to a leaf, which stores the class probabilities.

advantageous if the number of training samples is low and the number of leaves L is high or there are many classes H.
The training procedure is straightforward. For each of the M ferns in the ensemble we first choose the random feature dimensions $\mathbf{d}^m$. The feature dimensions are randomly drawn from all possible features without repetition in order to hold the assumption for independent feature combinations. Then the frequency matrix $\mathcal{P}^m$ is initialized with 0 for each class and leaf. By using a sparse matrix this step might be obsolete. Then we iterate over all N training samples given as training data matrix $\mathbf{X} = \left[\mathbf{x}^1, \mathbf{x}^2, \dots \mathbf{x}^N\right] \in \mathbb{R}^{N \times D}$ and calculate the leaf assignment from each sample by evaluating the random features. Therefore ϕ denotes a general mapping of an input sample $\mathbf{x}$ to a fern leaf $\mathcal{L}$, whereas different mappings, i.e. embeddings, do not alter the training and classification process. The frequency matrix is incremented for each occurrence of a leaf for the given class y^n. The classification is then carried out based on this frequency matrix.

Algorithm 1 Training the RFe classifier.

Require: Training samples $(\mathbf{X}, \mathbf{Y})$ (data and class label)
Require: Number of ferns M
Require: Number of features S
Require: Leaf assignment mapping ϕ
  for all $m \in 1 \dots M$ **do**
    $\mathbf{d}^m \leftarrow$ randomly sampled subset of D ($\mathbf{d}^m \subset \{1 \dots D\}^S$)
    $\mathcal{P}^m = \{0\}^{L \times H}$
    for all $n \in 1 \dots N$ **do**
      $\phi : \mathbf{x}^n(\mathbf{d}^m) \rightarrow \mathcal{L}$
      $\mathcal{P}^m(\mathcal{L}, y^n)^{++}$
    end for
  end for

Given the trained frequency matrix $\mathcal{P}^m$ for fern m we classify a given test sample $\mathbf{x}$ as illustrated in Algorithm 2. Therefore the conditional probabilities of each fern are

combined. In order to derive the probabilities from the frequency matrix, the frequency matrix is normalized for the general multinomial case using

$$p(\mathbf{F}^m|Y) = \frac{\mathcal{P}^m(\mathcal{L}, y) + u}{\sum_{\mathcal{L}} \left(\mathcal{P}^m(\mathcal{L}, y) + u\right)} \tag{10}$$

such that $\sum_{\mathcal{L}} \mathcal{P}^m(\mathcal{L}, y) = 1$ where u is a Dirichlet prior with $u = 1$ [83]. The Dirichlet prior provides a smoothing of the probabilities among the classes and further handles the case if there are no counts in the frequency matrix for a leaf. In this case the probability would be zero and thus multiplying to zero and loosing the information from the other ferns, which is prevented by adding the Dirichlet prior. The input sample is assigned to a leaf providing $p(\mathbf{F}^m|Y)$, which is used to combine the conditional probabilities in a logarithmic manner. The maximum probability over all classes from all combined ferns is then used as predicted class $\hat{y}$.

Algorithm 2 Classification of a sample using the RFe classifier.

Require: Test sample $\mathbf{x}$
Require: Number of ferns M
Require: Chosen features $\mathbf{d}$
Require: Trained frequency matrix $\mathcal{P}$
Require: Leaf assignment mapping ϕ
  $p(\mathcal{F}|Y) = 0$
  for all $m \in 1 \dots M$ **do**
    $\phi : \mathbf{x}(\mathbf{d}^m) \rightarrow \mathcal{L}$
    $p(\mathcal{F}|Y)$ += $\log\left(p(\mathbf{F}^m|Y)\right)$
  end for
  $\hat{y} = \arg\max_{y^i} p(\mathcal{F}|Y = y^i)$

As we have introduced the framework, we propose different embeddings that implement the leaf assignment ϕ in the next sections. The embedding should be chosen in such a way that the different classes can be optimally separated. A higher commonality within the leaves induces more information thus providing a higher classification confidence. This is similar to RFo where e.g. information gain [41], i.e. information maximization or entropy minimization, is used for feature selection in order to derive possibly pure class leaves for better classification results. However, we do not apply this in an iterative node optimization manner as in the context of RFo, but through our different embeddings much faster.

3.2 Subspace embeddings

At first we introduce our subspace embeddings, which we name *Oblique Ferns* similar to the RFo. For this method we project the samples into a linear subspace of the original input space. The transformation of a feature representation to a subspace is a common method in machine learning. In the context of RFo subspace methods are successfully used to obtain smaller classification errors [78, 74], thus the application in our RFe framework is obvious.

A subspace embedding as dimensionality reduction is convenient as the ferns memory is exponential in the number of features S, i.e. $\mathcal{O}(2^S)$. By using such a projection we can consider a higher number of features (more than the practical limitation of $S = 12 \ldots 15$) per fern and thus derive more discriminative base classifiers within the ensemble. Further, our subspace embeddings can adapt to the input data in order to derive a higher commonality within leaves.

Next we describe our Oblique Ferns framework into which all different linear subspaces can be integrated. In the following sections we propose the usage of different methods to obtain the linear projections and describe their individual properties. We describe the linear projection methods, i.e. random orthonormal projections, PCA, LDA, and CCA, in detail.

3.2.1 Oblique Ferns framework

Our Oblique Ferns framework can be derived in a similar way as described in 3.1. Therefore we first provide a statistical formulation and then the integration into the RFe framework.

Again, we want to maximize the posterior probability

$$p(Y = y^i | f_1, f_2, \ldots, f_C) \propto p(f_1, f_2, \ldots, f_C | Y = y^i) \tag{11}$$

as shown before, in order to derive the most appropriate class label for a given set of input features f_c. In this case we derive the features from a subspace projection of the original data into a reduced space $\mathbb{R}^T \rightarrow \mathbb{R}^S$. The T feature dimensions in the input data space are randomly chosen for each fern and combined over multiple ferns in an ensemble manner [41]. One can think of this as a windowed subspace projection of the original high dimensional input space to a low dimensional space in which the features are calculated. This provides a local emphasis on the input features, that can extract information which otherwise might be neglected in a global approach due to the curse of dimensionality. This local selectivity was, e.g. shown for a random subspace LDA [117].

A fundamental statistical assumption of our subspace methods (CCA, LDA, PCA) is, that the input data is normally distributed [51, 86]. For CCA and LDA the input data features should be independent [86]. Both requirements are not guaranteed to be fulfilled in the general case. This makes the projections suboptimal and thus only approximative [51, 86], but still accurate enough as we use them only for dimensionality reduction. All these subspace methods provide a new orthogonal subspace, which is generally regarded independent [92]. But for the general case of an arbitrary input distribution, the independence cannot be guaranteed. Thus we use a Semi-Naive Bayesian approach in order to model the joint probabilities of the possibly dependent variables. Further, the Semi-Naive Bayesian approach combines the features in the subspace. Depending on the relation of the number of features S to the subspace dimensionality T, the most important dimensions of the projections can be selected, that e.g. provide most variance in case of PCA. However, the subspace dimensions are mutually dependent from the common input dimensions, which is modeled through combining the features as well.

By assuming independence between features we can rewrite the joint probabilities as

$$p(f_1, f_2, \ldots, f_C | Y = y^i) = \prod_{j=1}^{C} p(f_j | Y = y^i), \tag{12}$$

which resembles a Naive Bayesian model. But as discussed this assumption does not hold for the general case. Therefore we formulate a trade-off between the independence of the individual input variables and the jointly modeled probabilities by grouping features to $S = \frac{C}{M}$ groups, similar as described in Section 3.1. We apply a dimensionality reduction in each fern thus the choice of S is limited by the subspace dimensionality T and for the CCA and LDA by the number of classes, as we discuss later. The chosen input data dimensionality T depends on the underlying data properties, but in all cases $S \leq T$. Each fern evaluates S binary features $\mathbf{F}^m = [f_1^m, f_2^m, \dots f_S^m]$ which are calculated in the subspace. This leads to $L = 2^S$ leaf nodes $\mathcal{L}^m$ per fern. Thus we formulate the joint probabilities of a fern $\mathbf{F}^m$ similar to Equation 5 as

$$p(f_1, f_2, \dots, f_C | Y = y^i) = p(\mathcal{F} | Y = y^i) = \prod_{m=1}^{M} p(\mathbf{F}^m | Y = y^i). \tag{13}$$

The final classification rule is then stated equivalently to Equation 7 as

$$\hat{y} = \arg\max_{y^i} \sum_{m=1}^{M} \log \big(p(\mathbf{F}^m | Y = y^i) \big). \tag{14}$$

Although we consider a high number of features per fern for constructing the ferns (possibly up to $T = \lceil \frac{D}{2} \rceil$ in the evaluation), the Semi-Naive Bayesian principle can still be used by assuming independence between the jointly modeled probabilities. For each fern we select randomly chosen feature dimensions without repetition. Further, we use M independently trained ferns, where the number of different combinations of input dimensions is $\frac{D!}{\lceil D/2 \rceil !}$. Thus it is very unlikely that multiple ferns contain the same feature combination. Each new feature combination creates another projection of the data using the subspace methods thus creating independent features per fern. However, we show in the evaluation that these assumptions do not hold in all cases.

Our proposed subspace embedding is shown in Figure 4 for a single fern. We apply a projection ψ on the data by using T randomly selected features per fern to obtain a S-dimensional subspace, which is $\psi : \mathbb{R}^T \rightarrow \mathbb{R}^S$ with $T \geq S$. We define an explicit subspace, similar to [41], which is used to derive the linear projection. The dimensionality S can either be chosen freely, as e.g. when using PCA, or it is limited by the number of classes as we discuss later for the LDA and CCA subspaces. The mapping ψ, our linear projection, can be implemented as matrix multiplication. We only consider linear projections as subspace projection methods. The samples are projected into the fern-specific subspace $\mathbf{x}'^m = \mathbf{x}(\mathbf{d}^m)\mathbf{W}^m \in \mathbb{R}^S$ where $\mathbf{W}^m \in \mathbb{R}^{T \times S}$ is a projection matrix for fern m. In this subspace the assignment vector from data to the leaves is obtained by the binary vector $\mathbf{F}^m = [f_1^m, \dots f_S^m] = (\mathbf{x}'^m > \mathbf{c}^m)$, which can be derived for each dimension independently. The individual features are calculated by thresholding the individual dimensions by

$$f_s^m = \begin{cases} 1, & \text{if } x_s'^m > c_s^m \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

The threshold $\mathbf{c}^m$ can be derived from the median value over all samples for each dimension, in order to bisect the data. A more sophisticated, but computationally more complex approach for calculating the *optimal* threshold is provided by Strecha et al. [105].

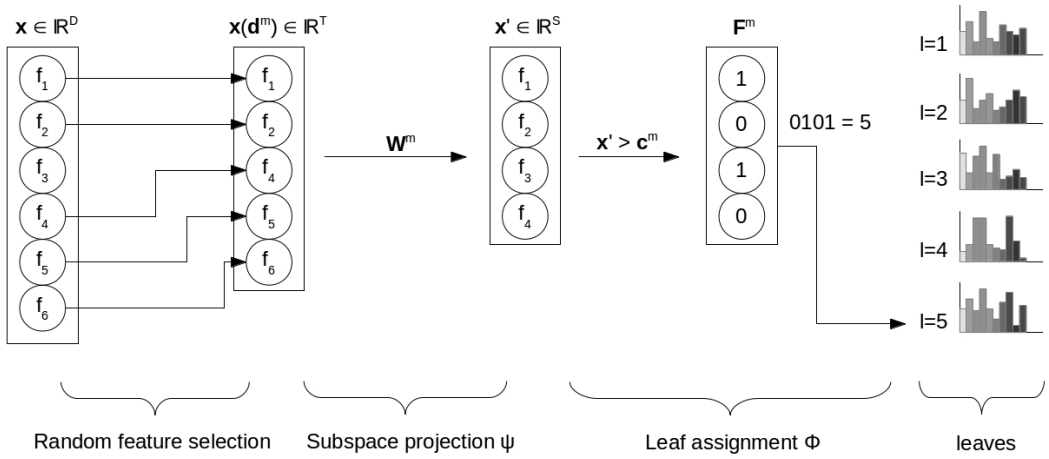


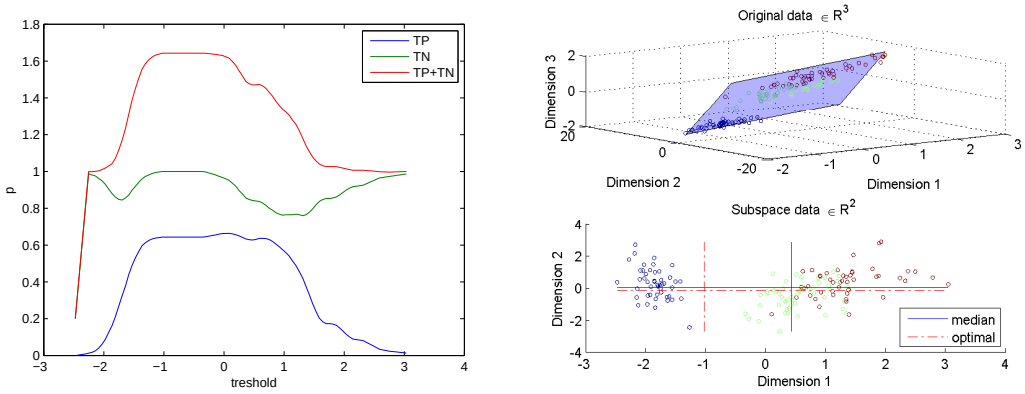
Figure 4: Illustration of a single Oblique Fern. The input data of randomly selected input dimensions per fern is used to obtain the subspace projection $\mathbf{W}^m$ and the threshold $\mathbf{c}^m$. Samples can be assigned to a leaf by projecting them into the subspace and thresholding.

Therefore they project two input vectors $\mathbf{x}^1$ and $\mathbf{x}^2$ into the subspace $\mathbf{y}^1 = \mathbf{x}^1\mathbf{W}$ and $\mathbf{y}^2 = \mathbf{x}^2\mathbf{W}$ and calculate the individual threshold for each dimension using a linear search over possible thresholds $c_s \in \{\min(y_s^1, y_s^2), \dots \max(y_s^1, y_s^2)\}$. If the threshold $c_s < \min(y_s^1, y_s^2)$ or $c_s > \max(y_s^1, y_s^2)$ the s-th assignment bits coincide and thus assign the two samples to the same leaf. This optimal assignment can be formulated as a maximization problem [105]

$$\begin{aligned}\max_{c_s}\{\mathrm{TP}(c_s) + \mathrm{TN}(c_s)\} = \max_{c_s}\{\ & p(\min(y_s^1, y_s^2) \geq c_s \vee \max(y_s^1, y_s^2) < c_s|\mathcal{P}) \\ & + 1 - p(\min(y_s^1, y_s^2) \geq c_s \vee \max(y_s^1, y_s^2) < c_s|\mathcal{N})\}\end{aligned} \tag{16}$$

where TN denotes the true negative and TP the true positive rate. The probabilities $p(\cdot)$ are estimated on randomly chosen pairs of input samples.
As this optimal threshold is only defined between two classes, i.e. the positive $\mathcal{P}$ and negative $\mathcal{N}$ pairs, we adapt this for our multi-class framework. Therefore we generalize this approach to multiple classes by applying a one-vs-all scheme, familiar from e.g. SVM classifiers [91]. We maximize the problem formulation of Equation 16 by setting matches between the class samples as positive pairs and all matches between class and non-class samples as negative pairs. This is done for each class and we average the true positive and true negative rates over all classes, i.e. the threshold that separates one or more classes best from the rest. Further the true positive and true negative rates are smoothed using a mean filter in order to mitigate fluctuations due to the random pair selection. An example of this threshold method is shown in Figure 5. Figure 5a shows the function of the true negative and true positive rate over the different thresholds and Figure 5b shows the selected thresholds for the median and the optimal method with the underlying data. If there is a good separation possible, as for dimension 1, the optimal threshold clearly selects the better threshold, but if the data is not separable, as for dimension 2, both thresholds are similar and the median provides a good and fast approximation of the optimal threshold.

The computational complexity of the median is $\mathcal{O}(SN)$ [55] for S subspace dimensions and N training samples. The optimal threshold has a computational complexity of



(a) True positive and true negative rate for different thresholds.

(b) PCA subspace with thresholds.

Figure 5: Different thresholds shown for sample data, illustrated by using the first 3 dimensions of the Iris dataset [27].

$\mathcal{O}(tHSN)$ with t denoting the number of thresholds used in the linear search and H the number of classes.

During training all training samples are used to determine the subspace projection $(\mathbf{W}^m, \mathbf{c}^m)$ and further the training samples are used to estimate the conditional probabilities in each leaf. For constructing each fern, T different randomly selected feature dimensions are used and thus different subspaces are derived for each fern. Although some of the subspace methods are unsupervised, the training of our classifier is still supervised, but only the calculation of the subspace projection is unsupervised.
For testing, the data sample is subsampled using the chosen feature dimensions per fern and further projected into the subspace. The binary assignment vector is obtained by thresholding the data using the trained threshold and then assigned to a leaf for each fern. The class probabilities of all ferns are then combined to obtain the classification result equivalent to Algorithm 2.

The linear projections used can be derived in several ways, where each method has its specific properties. They can be chosen

(a) in a random matter, similar to oblique splits at random orientations [9, 8],

(b) unsupervised, e.g. using PCA which captures most of the variance of the original data, or

(c) supervised by considering the class labels and thus deriving more discriminative leaf assignments as e.g. LDA maximizes inter-class distances, or CCA obtains an orthogonal subspace in which the data maximally correlates to the class labels.

Subspace projections are sensitive to feature scaling. Therefore we preprocess the input data and normalize each feature dimension to zero mean and unit variance, which is a common normalization method [2, 52]. The mean and variance are estimated from the training set. Although more sophisticated feature scaling methods exist, which derive better results by weighting the feature scales, as e.g. in [7], we do not apply such domain specific methods.

In the next sections we describe different methods for obtaining the required linear projections.

3.2.2 Random Oblique Ferns

The first and most straightforward method for obtaining the projection matrix $\mathbf{W}^m$ is to choose it in a random matter. In this case we use a random, orthonormal matrix $\mathbf{W}^m \in \mathbb{R}^{T\times S}$ for each fern. An orthonormal matrix can be derived from a random matrix $\mathbf{A} \in \mathbb{R}^{T\times S}$, whose values are drawn from an uniform distribution $\mathbf{A} \sim \mathcal{U}(0,1)^{T\times S}$ by using the QR decomposition [32] such that

$$\mathbf{A} = \mathbf{QR}. \tag{17}$$

$\mathbf{R} \in \mathbb{R}^{T\times S}$ is an upper triangular matrix and $\mathbf{Q} \in \mathbb{R}^{T\times T}$ is an orthonormal matrix, whose columns are orthogonal unit vectors such that $\mathbf{Q}^\intercal\mathbf{Q} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. In case of dimension reduction a non-square matrix $\mathbf{W}^m$ is needed, thus we take the S first columns of the square matrix $\mathbf{Q}$.

The random subspace is unsupervised and data independent, hence it does not incorporate any of the input data's properties. This might be derogatory for the classification result, but we state it here as baseline and for completeness. However, each projection is randomized and can provide a different "view" on the data, even if the features are the same. Especially if the dataset has a low dimensionality, it decreases the correlation between the ferns. If all ferns were trained with the same features, the ferns ensemble would overfit the training data as all ferns would be highly correlated. The random projections have a regularization effect on the decision boundaries, as the different projections create new classification hyperplanes for each fern that smooth the overall decision boundary of the ensemble.

3.2.3 PCA Ferns

An unsupervised projection method is PCA [45], which calculates an orthogonal subspace whose basis-vectors correspond to the directions of highest variance of the data. By increasing the variance in the orthogonal subspace the data gets "stretched", thus increasing the margin between the decision boundaries. PCA is a well-known method for dimensionality reduction. Usually it is used to reduce the dimensionality of a dataset, where only the dimensions containing most variance are kept. Instead of applying PCA to the whole dataset, we use PCA to generate a subspace for each fern individually. This provides a more feature-specific emphasis on the variances of the data, which increases the specificity on variances and thus the strength of the base classifiers instead of a global representation.
The PCA is calculated from the covariance matrix $\mathbf{C} \in \mathbb{R}^{T\times T}$ using

$$\mathbf{C} = \frac{1}{N-1}\mathbf{X}^\top\mathbf{X} \tag{18}$$

where $\mathbf{X} \in \mathbb{R}^{N\times T}$ is the subsampled and normalized data matrix and N is the number of data samples.

Further, the eigenvalues and eigenvectors are computed from $\mathbf{C}$ by using a simple eigenvalue decomposition, such that

$$\mathbf{CV} = \mathbf{VD} \tag{19}$$

where $\mathbf{D} = diag(\lambda_1, \dots \lambda_T)$ is a diagonal matrix with the eigenvalues of $\mathbf{C}$ on the main diagonal and the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{C}$. The eigenvectors, which define a new orthogonal coordinate system, indicate the directions of the highest variances, i.e. principal components, within the dataset. The corresponding eigenvalues account for their statistical significance. By discarding the minor principal components, i.e. the eigenvectors with smaller eigenvalues, the projection matrix $\mathbf{W}^m \in \mathbb{R}^{T \times S}$ can be formed from the S retained eigenvectors thus achieving dimension reduction.
Alternatively the principal components can be acquired using the Singular Value Decomposition, see [51] for details.

There are no restrictions on the dimensionality of the subspace, unless there are more data samples than features, thus $N > T$ and the rank of $\mathbf{C}$ is full [43].

3.2.4 LDA Ferns

In contrast to the unsupervised PCA, LDA provides a supervised subspace method thus it incorporates the class labels. LDA can be used to find a linear, discriminant subspace for multi-class problems by minimizing the intra-class distances and maximizing the inter-class distances. The classification is then performed in the more discriminative subspace, which improves the separation between the classes and thus increases the commonality of the data within the leaves.

For each fern we calculate the LDA subspace based on the fern-specific feature dimensions. LDA was initially introduced for two-class problems [27]. Therefore the scatter matrix $\mathbf{S}^i \in \mathbb{R}^{T \times T}$ for class y^i, which is an estimate of the covariance matrix, is given by

$$\mathbf{S}^i = \sum_{\mathbf{x} \in y^i} (\mathbf{x} - \boldsymbol{\mu}^i)^\top (\mathbf{x} - \boldsymbol{\mu}^i) \tag{20}$$

where $\boldsymbol{\mu}^i = \frac{1}{N_i} \sum_{i=1}^{N_i} \mathbf{x}^i$ is the mean over the class samples and $\mathbf{x} \in \mathbb{R}^T$ is a subsampled input sample. Thus the full intra-class scatter matrix $\hat{\boldsymbol{\Sigma}}_w$ is calculated by

$$\hat{\boldsymbol{\Sigma}}_w = \mathbf{S}^1 + \mathbf{S}^2 = \sum_{i=1}^{2} \sum_{\mathbf{x} \in y^i} (\mathbf{x} - \boldsymbol{\mu}^i)^\top (\mathbf{x} - \boldsymbol{\mu}^i). \tag{21}$$

Further, the inter-class scatter matrix $\hat{\boldsymbol{\Sigma}}_b \in \mathbb{R}^{T \times T}$ is given by

$$\hat{\boldsymbol{\Sigma}}_b = (\boldsymbol{\mu}^1 - \boldsymbol{\mu}^2)^\top (\boldsymbol{\mu}^1 - \boldsymbol{\mu}^2) \tag{22}$$

where $\boldsymbol{\mu}^1$ and $\boldsymbol{\mu}^2$ are the means over the class samples for class y^1 and y^2, respectively. From these two matrices the projection $\mathbf{W}$ is determined by maximizing the so called *Rayleigh* coefficient

$$\mathcal{J}(\mathbf{W}) = \frac{|\mathbf{W}^\top \hat{\boldsymbol{\Sigma}}_b \mathbf{W}|}{|\mathbf{W}^\top \hat{\boldsymbol{\Sigma}}_w \mathbf{W}|}. \tag{23}$$

The projection matrix $\mathbf{W}$ can be derived from the eigenvectors of $\hat{\mathbf{\Sigma}}_w^{-1}\hat{\mathbf{\Sigma}}_b$ if $\hat{\mathbf{\Sigma}}_w$ is non-singular, thus requiring $N \geq T + H$ [66].

This approach of calculating the LDA would considerably limit our classifier to only two-class problems, therefore [90] introduced a multi-class extension. Similarly to the two-class problem, the intra-class scatter matrix $\hat{\mathbf{\Sigma}}_w$ is calculated by accumulating the scatter matrices over all H classes $Y \in \{1, \dots H\}$ by

$$\hat{\mathbf{\Sigma}}_w = \sum_{i=1}^{H} \mathbf{S}^i = \sum_{i=1}^{H} \sum_{\mathbf{x} \in y^i} (\mathbf{x} - \boldsymbol{\mu}^i)^\top (\mathbf{x} - \boldsymbol{\mu}^i). \tag{24}$$

Further, the inter-class scatter matrix $\hat{\mathbf{\Sigma}}_b$ is given by

$$\hat{\mathbf{\Sigma}}_b = \sum_{i=1}^{H} N_i (\boldsymbol{\mu}^i - \boldsymbol{\mu})^\top (\boldsymbol{\mu}^i - \boldsymbol{\mu}) \tag{25}$$

where N_i is the number of samples of class y^i such that $N = \sum_{i=1}^{H} N_i$ and

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}^i \tag{26}$$

is the mean over all samples. Note, that for the multi-class extension different weighting schemes exist for incorporating the number of samples per class, see e.g. [63, 66]. Hence, we follow the notation of [66].
The projection vectors $\mathbf{w}$ are then obtained by solving the generalized eigenvalue problem

$$\hat{\mathbf{\Sigma}}_b \mathbf{w} = \lambda \hat{\mathbf{\Sigma}}_w \mathbf{w} \tag{27}$$

where λ is the eigenvalue to the corresponding eigenvector $\mathbf{w}$. Again, we choose the S eigenvectors, with the corresponding highest eigenvalues for assembling the projection matrix $\mathbf{W}^m \in \mathbb{R}^{T \times S}$.

The LDA subspace dimensionality is limited by $\hat{\mathbf{\Sigma}}_b$, whose rank is $\mathrm{rk}(\hat{\mathbf{\Sigma}}_b) = H - 1$ at most and $\mathrm{rk}(\hat{\mathbf{\Sigma}}_w) = N - H$ [66]. By assuming that $N \gg H$, only the rank for $\hat{\mathbf{\Sigma}}_b$ limits the subspace dimensionality.

3.2.5 CCA Ferns

A different supervised approach is given by the CCA [46]. Instead of maximizing the inter-class distance like LDA, or maximizing the variance of a dimension like PCA, CCA is used to find a lower-dimensional subspace, in which two sets of variables are maximally correlated. One set of variables is given by the training data and the other set is created from the class labels [107].

Again, the first set is our subsampled training data $\mathbf{X} \in \mathbb{R}^{N \times T}$ and the second set is the class label matrix $\mathbf{Y} \in \{0, 1\}^{N \times H}$, which encodes the class label that corresponds to the data sample. Therefore the corresponding matrix value is set to $\mathbf{Y}(i, y^i) = 1$ using a 1-of-K binary coding [4]. Although the training data matrix incorporates different feature dimensions per fern and thus is different for each fern, the class label matrix is the same for all ferns.

We use the CCA to obtain an orthogonal subspace with maximum correlation of data to class labels. Two projection vectors $\mathbf{w}_x \in \mathbb{R}^T$ and $\mathbf{w}_y \in \mathbb{R}^H$ are calculated from the correlation coefficient

$$\rho = \frac{\mathbf{w}_x \mathbf{X}^\top \mathbf{Y} \mathbf{w}_y^\top}{\sqrt{\mathbf{w}_x \mathbf{X}^\top \mathbf{X} \mathbf{w}_x^\top \mathbf{w}_y \mathbf{Y}^\top \mathbf{Y} \mathbf{w}_y^\top}} \tag{28}$$

by maximizing ρ. This equation can be reformulated as maximization problem since ρ is invariant to the scaling of $\mathbf{w}_x$ and $\mathbf{w}_y$ [107]

$$\max_{\mathbf{w}_x, \mathbf{w}_y} \mathbf{w}_x \mathbf{X}^\top \mathbf{Y} \mathbf{w}_y^\top \quad \text{s.t.} \quad \mathbf{w}_x \mathbf{X}^\top \mathbf{X} \mathbf{w}_x^\top = 1,\ \mathbf{w}_y \mathbf{Y}^\top \mathbf{Y} \mathbf{w}_y^\top = 1. \tag{29}$$

It is equivalent to the generalized eigenvalue problem

$$\left[\mathbf{X}^\top \mathbf{Y} (\mathbf{Y}^\top \mathbf{Y})^{-1} \mathbf{Y}^\top \mathbf{X}\right] \mathbf{w}_x^\top = \lambda \left[\mathbf{X}^\top \mathbf{X}\right] \mathbf{w}_x^\top \tag{30}$$

where we determine the eigenvectors that resemble the projection vector $\mathbf{w}_x$. For our orthogonal projection matrix we only consider $\mathbf{w}_x$ that provides the projection from data to the subspace. We choose the S eigenvectors that correspond to the major eigenvalues to assemble $\mathbf{W}^m \in \mathbb{R}^{T \times S}$.

The CCA is sensitive to the ranks of $\mathbf{X}$ and $\mathbf{Y}$. The rank of $\mathbf{Y}$ is full, thus $\mathrm{rk}(\mathbf{Y}) = H$, if samples are present for each class. Otherwise there would be a zero column due to our 1-of-K coding resulting in a non-full matrix. The rank of $\mathbf{X}$ is sensitive to the feature selection and $\mathrm{rk}(\mathbf{X}) = T$ at most. If one or multiple feature dimensions are constant, the matrix is not of full rank thus $\mathrm{rk}(\mathbf{X}) < T$. In order to mitigate this problem we apply a regularization to the data matrix [107]. The maximum subspace dimensionality is therefore $\min(H, T)$. However, only the limitation of H is relevant, as T can be chosen depending on data properties.

3.2.6 Connection to binary hashing

There exists a connection of our Oblique Ferns to different binary hashing methods. For example our PCA Ferns are related to Iterative Quantization (ITQ) [35] or our LDA Ferns to the LDAHash [105]. Similarly, they use an intermediate subspace embedding in which they binarize the data in order to generate a binary hash. The binary codes are then used to find nearest neighbors by calculating the Hamming distance from a query code to the database codes. However, our Oblique Ferns naturally include a classification that renders a subsequent nearest neighbor search unnecessary. Özuysal [82] has already shown, that the RFe acts as an approximate nearest neighbor classifier, that resembles a randomized locality-sensitive hash function in the feature space. Thus each fern calculates a hash from the input data, which is then used for classification. Further, we calculate the embedding on multiple random subspaces of the input feature space, that provide more emphasis on local data structures compared to a global subspace representation.

3.2.7 Comparison of subspace embeddings

Each of our proposed subspace methods exhibits its own characteristics. Therefore we provide a comparison in Table 1, which shows an overview over the subspace methods used. All but one of our proposed subspace methods depend on the data, thus incorporate the data properties for the subspace generation. The random oblique method does not incorporate any data properties. The two unsupervised subspace methods, random oblique and PCA, have no considerable limitation of the subspace dimension. Though, the subspace dimensionality of the LDA and CCA methods are limited by the number of classes which are present in the problem. For problems with a high number of classes, e.g. > 10, this is no serious obstruction, but for two-class problems this limitation should be considered.

The time complexity for acquiring a subspace of a single fern is similar for all subspace methods, as they are based on the calculation of covariance matrices and eigenvalue decomposition. For calculating the covariance matrix the complexity is $\mathcal{O}(NT^2)$ and for a general eigenvalue decomposition $\mathcal{O}(T^3)$ [85]. For simplicity we assume $T, H \ll N$. This leads to a complexity of $\mathcal{O}(NT^2 + T^3)$ [85] for the PCA, $\mathcal{O}(N \max(T, H)^2 + \max(T, H)^3)$ for the CCA, and $\mathcal{O}(NT \min(N, T))$ [66] for the LDA, where N denotes the number of training samples, H the number of classes and T the number of input features. The random oblique method has the lowest complexity of $\mathcal{O}(T^3)$ [32] due to the QR decomposition.
The memory complexity is unchanged with $\mathcal{O}(M \min(N, LH))$ with $L = 2^S$, but note that for constructing each fern $T \geq S$ features are used. Thus we can incorporate T features into a fern, at reduced memory requirements of S.

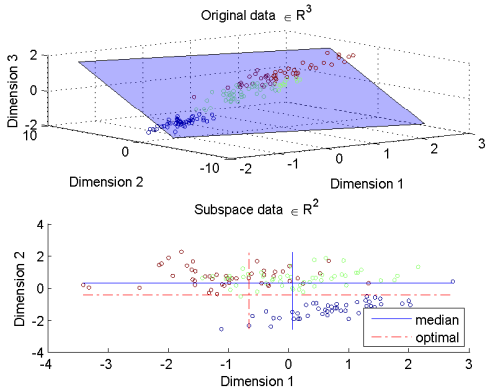
The runtime of the RFe for training is $\mathcal{O}(N \log L)$ where $L = 2^S$. For the related RFo the training complexity is $\mathcal{O}(KNL)$ where K is the number of node tests. The runtime for classification is $\mathcal{O}(\log L)$ for both the RFe and RFo, and $\mathcal{O}(T \log L)$ for the Oblique Ferns where T is the subspace dimensionality.

Method	Max. subspace dimension	Computational complexity training	Supervised	Data dependent
Random Oblique	T	$\mathcal{O}(T^3)$	×	×
PCA	T	$\mathcal{O}(NT^2 + T^3)$	×	✓
LDA	$H - 1$	$\mathcal{O}(NT \min(N, T))$	✓	✓
CCA	H	$\mathcal{O}(N \max(T, H)^2 + \max(T, H)^3)$	✓	✓

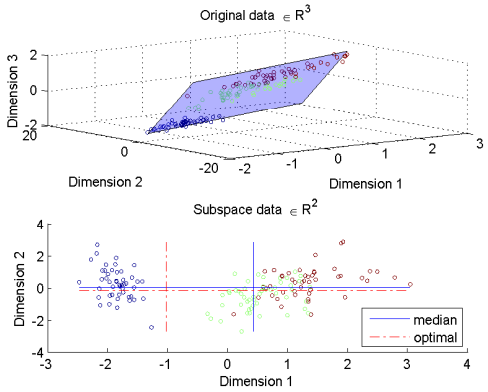
Table 1: Comparison of different subspace methods. The complexity is stated for a single fern.

Figure 6 illustrates the properties of different subspace method for a simple example of $\psi : \mathbb{R}^3 \to \mathbb{R}^2$. Nevertheless the essential characteristics of the subspace projections can be observed. The random oblique projections does not generate any meaningful subspace, thus there is no pattern visible. Solely the samples are more scattered in $\mathbb{R}^2$ compared to the case of truncating the third dimension. For the PCA it can be clearly seen that the projection plane fits the direction of the highest variance and thus the $\mathbb{R}^2$ subspace offers the biggest variance along its feature dimensions, i.e. the most scattered

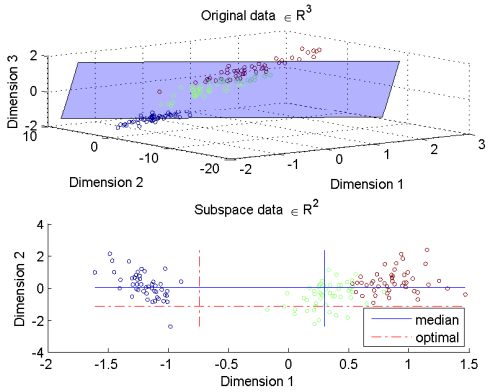
data representation. In the LDA subspace the three classes can be best separated, thus the three clusters are most discriminative as theory implied. Although CCA finds a subspace that correlates the data classes most, it does not separate the data as well as LDA. The samples are denser clustered but the clusters overlap.
The figures also show our two proposed threshold methods for assignment bit generation. While the median threshold simply bisects the data, the optimal threshold finds the threshold for each dimension that optimally separates the class samples.



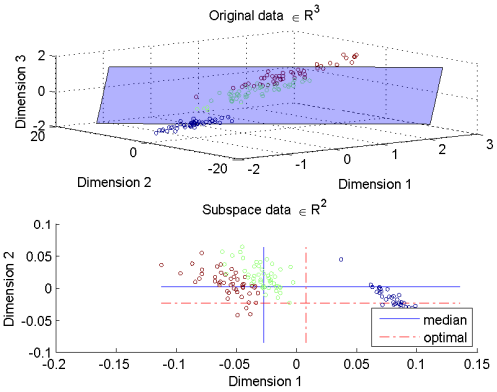
(a) Random oblique.



(b) PCA.



(c) LDA.



(d) CCA.

Figure 6: Different subspaces and thresholds illustrated by using the first 3 dimensions of the Iris dataset [27].

As we have only covered linear subspaces in this section, one might think of using non-linear subspace methods, were e.g. kernel methods can be used. But as we deal with large datasets, the kernel methods are infeasible to compute with limitations in memory and runtime [13]. The kernel space complexity is quadratic in the number of samples [122, 96], thus for our datasets with a high number of samples kernel methods are infeasible to compute. The linear subspace methods are only polynomial in the subspace dimensionality T, which is low due to our design of subsampling.

3.3 Ordinal embeddings

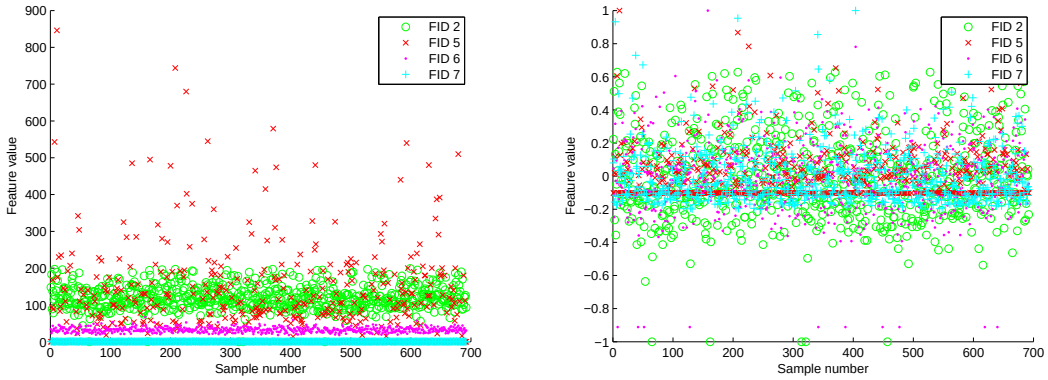
Next to the subspace embeddings we introduce different ordinal embeddings, that use the order of the features within an input sample as information. This information can be obtained by using the order of the values directly [18] or by using an extremal value [98, 123], i.e. the maximum or minimum value, of the input data. Ordinal information has recently gained popularity for feature descriptors [118, 108] and was also proposed as embedding for a RFo [98]. The advantage of ordinal embeddings is, that they are invariant to any linear perturbation and additional spatial information about the feature location is added into the model, i.e. not only that one feature is larger than another, but also the position of the feature within the feature vector.
This invariance property can be illustrated in a small example. Given the vector $\mathbf{x} = [10, 4, 14, 6, 1]$ we apply different perturbations to the vector.

1. We apply a function $f(x) = 2 \cdot x + 2$ thus $\mathbf{x}' = f(\mathbf{x}) = [22, 10, 30, 14, 4]$.
2. We add ± 1 in alternation thus $\mathbf{x}' = [11, 3, 15, 5, 2]$.
3. We add Gaussian noise $n \sim \mathcal{N}(-1, 1)$ thus $\mathbf{x}' = \mathbf{x} + n = [10.4, 3.8, 13.2, 6.8, 1.3]$.

Given the example of an image as input sample, the first function might resemble a non-linear light change and the latter two perturbations can be different noise effects. In all cases the order of the values stays the same, as well as the position of the maximum and minimum, respectively. While in linear space the similarity of the different data vectors is questionable, all different examples are congruent in ordinal space, i.e. they have the same order $x_5 < x_2 < x_4 < x_1 < x_3$.
The original feature evaluation method, presented in Section 3.1, can already be regarded as a very simple ordinal embedding, as it relies on the comparison of two input values. Thus each feature is based on a partial order of two input values, where the first is smaller or larger than the second one. However, this is not a sophisticated embedding and we show different ordinal embeddings that outperform the original one.
Therefore we use partial order statistics on the input data to derive a more sophisticated leaf assignment for each fern. We propose and evaluate several approaches which incorporate ordinal information into the RFe framework in different ways.

Although all ordinal embeddings are invariant to linear feature scaling per definition, this assumption holds only if the data is derived from one feature scale, as e.g. from an image where each feature (pixel) value is the range of $[0, 255]$ or from physical measurements where each feature value (measurement in e.g. meters) is given in the range of $[0, \infty)$. Given different feature scales that cannot be compared in a reasonable way, as metaphorically speaking comparing apples and oranges, this invariance vanishes and even restrains an application. The scale of one feature might be smaller by magnitudes compared to another feature thus prohibiting a reasonable comparison.
An illustration of this problem is given in Figure 7 where the raw feature values are shown for the Diabetes dataset [104]. In this specific dataset the features are derived from different feature scales. If we use the raw data for ordinal feature comparison, feature *FID 7* (diabetes pedigree function) will always be smaller than feature *FID 5* (2-hour serum insulin $\mu U/ml$) which renders a comparison useless, i.e. a comparison will not contain

any discriminative information that can be used for classification. Therefore we convert all features to a common scale by normalizing the data to zero mean and to a range of $[-1, 1]$, if the individual features are derived from different scales. If we use pixel values as features, this normalization is already done in the process of image formation and thus not necessary. Nevertheless a problem could arise in computer vision for example, if different descriptor types are combined that have different feature scales. In such a case feature normalization has to be performed.



(a) Raw feature values.

(b) Normalized feature values.

Figure 7: Comparison of different feature scales on selected feature dimensions of the Diabetes dataset [104].

In the following sections we introduce several methods of incorporating ordinal information into our RFe framework. We start with our so called *WTA Ferns*, that incorporate the Winner Take All (WTA) hash [123] for generating a non-linear feature transformation for leaf assignment. This hashing function derives a hash code from a feature vector by using ordinal information such that the distance between hash codes correlates to rank similarity measures. Then we propose the *Maximum Order Ferns*, which incorporate a relaxed formulation of [98] that uses the position of the maximum value within the feature vector. This embedding enforces multiple order relations instead of a single one used in the original feature evaluation function within a single leaf assignment bit. Further we introduce the *Direct Ordinal Ferns*. This embedding directly integrates the order of the input features by using the Complete Rank Transform (CRT) [18]. Each possible input order is then assigned to a different leaf, thus enforcing a high commonality in each leaf. At last we formulate an extension to the Direct Ordinal Ferns, the *Soft Assignment Ferns*, which try to mitigate noise problems in the Direct Ordinal Ferns. Therefore we use a soft assignment of the samples to the leaves by using rank correlation measures as feature weights.

3.3.1 WTA Ferns

The first of the proposed Ordinal Ferns are our so called WTA Ferns, which implement the WTA hash for feature calculation and leaf assignment. This hashing method was proposed by Yagnik et al. [123] and provides a non-linear feature transformation which we use for leaf assignment. The WTA hash has several advantages that support the usage

in our RFe framework. It provides a learning-free, non-linear mapping from the input data to the hash code, which is based on partial orders. This ordinal representation adds tolerance to numeric perturbations and noise in the input values. The dimensionality of the hash code can be tuned using two hyperparameters that integrate well into the RFe framework. Further, the algorithm can be efficiently implemented which favors the RFe principle.

In the following paragraphs we introduce our WTA Ferns. We first describe the hashing function itself and then how it is incorporated into our RFe framework.
The WTA hashing function derives a hash code from an input sample $\mathbf{x}$ by permuting the sample S times using random permutations $\boldsymbol{\Theta} = \left[\boldsymbol{\theta}^1, \dots \boldsymbol{\theta}^S\right]$. The permuted input sample is denoted $\mathbf{x}' = \mathbf{x}(\boldsymbol{\theta}^i)$ and $\boldsymbol{\theta}^i$ is a random permutation vector that reorders the input sample. S such vectors can be used to generate S different permutations of the original input vector. From each of these permutations $\mathbf{x}'$ we use the K first dimensions to determine the index with the largest value, thus

$$c_i = \underset{d=\{1,\dots K\}}{\arg\max}\, \mathbf{x}'(d). \tag{31}$$

This results in a S-dimensional hash vector $\mathbf{c} = [c_1, c_2, \dots c_S] \in K^S$, where each hash code encodes the position of the maximum value within the window size thus $c_i \in \{1, \dots K\}$. For a window size of $K = 2$ the hash code is binary.
The algorithm for calculating the full WTA hash is illustrated in Algorithm 3. First the hash $\mathbf{c}$ is initialized with zeros. Then for each permutation the index of the maximum within the window size K is determined and added to the hash.
The computational complexity for calculating the hash of an input sample is $\mathcal{O}(SK)$ where S is the number of permutation and K the window size.

Algorithm 3 WTA hash calculation.

Require: Input sample $\mathbf{x}$
Require: S random permutations $\boldsymbol{\Theta} = \left[\boldsymbol{\theta}^1, \dots \boldsymbol{\theta}^S\right]$
Require: Window size K
  $\mathbf{c} = [0, \dots 0]$
  for all $s \in 1 \dots S$ **do**
    $\mathbf{x}' = \mathbf{x}(\boldsymbol{\theta}^s)$
    $c_s = \arg\max_{d=\{1,\dots K\}} \mathbf{x}'(d)$
  end for

We have described the WTA hashing algorithm and we further introduce the hash codes as ordinal features in the RFe framework. Therefore we formulate a probabilistic framework for our WTA Ferns.
The initial classification rule that maximizes the posterior probability is

$$\hat{y} = \underset{y^i}{\arg\max}\, p(Y = y^i | c_1, c_2, \dots c_C). \tag{32}$$

The posterior probability now depends on the individual hash codes c_i and not on binary features as described in Section 3.1.

By using Bayes' Formula we rewrite the probability as

$$p(Y = y^i|c_1, c_2, \dots c_C) = \frac{p(c_1, c_2, \dots c_C|Y = y^i)p(Y = y^i)}{p(c_1, c_2, \dots c_C)} \propto p(c_1, c_2, \dots c_C|Y = y^i). \tag{33}$$

Again, the class probability $p(Y)$ is assumed uniform and the denominator is neglected as it is independent of the class. In order to estimate the posterior probability we would have to model all joint probabilities, which is not feasible, as a high number of codes $C > 1000$ is necessary for good results [123]. Therefore we rewrite the conditional probabilities in a Naive Bayesian model as

$$p(c_1, c_2, \dots c_C|Y = y^i) = \prod_{j=1}^{C} p(c_j|Y = y^i), \tag{34}$$

thus assuming independence between the individual input features. Although our original input features are permuted for the generation of the codes, the random permutations of the input features[4] do not affect the statistical properties of the underlying joint probability distribution [14] as we assume independence between the input features. Therefore we relax the independence assumption and model some of the joint probabilities in a Semi-Naive Bayesian manner, thus forming $S = \frac{C}{M}$ groups equivalently to the original RFe formulation. Our ferns ensemble $\mathcal{C} = \{\mathbf{c}^1, \mathbf{c}^2, \dots \mathbf{c}^M\}$ is formed by M individual ferns, of which each fern calculates its specific WTA hash $\mathbf{c}^m = [c_1^m, c_2^m, \dots c_S^m]$ using the fern-specific permutation matrix thus providing independence between the individual codes. Each hash code is in $c_i \in \{1, \dots K\}$ which leads to $L = K^S$ leaves per fern. By using the joint probabilities of a fern $\mathbf{c}^m$ the conditional probabilities are integrated as

$$p(c_1, c_2, \dots c_C|Y = y^i) = p(\mathcal{C}|Y = y^i) = \prod_{m=1}^{M} p(\mathbf{c}^m|Y = y^i). \tag{35}$$

The final classification rule can be stated equivalently to Equation 7 as

$$\hat{y} = \arg\max_{y^i} \sum_{m=1}^{M} \log\big(p(\mathbf{c}^m|Y = y^i)\big). \tag{36}$$

Therefore each leaf stores the probability $p(\mathbf{c}^m|Y = y^i)$ or equivalently the number of training samples in the frequency matrix $\mathcal{P}(\mathbf{c}^m, y^i)$, which are then combined according to Equation 36.

The integration of the hashing method into our RFe framework is shown in Figure 8. The original input vector $\mathbf{x}$ is first subsampled, where $\mathbb{R}^D \rightarrow \mathbb{R}^T$ is the selection of the fern-specific feature subset $\mathbf{d}^m$. For this example the subsampled vector is permuted $S = 3$ times using the random permutation vectors $\boldsymbol{\theta}_{1,2,3}$. The permutation vectors are randomly sampled from this $T \geq K$ dimensional feature subspace. The random

[4]Formally, this type of variable is called exchangeable sequence of random variables. An infinite sequence of random variables is exchangeable, if a rearrangement with a finite permutation does not alter the underlying joint probability distribution. [14]

permutation matrix $\mathbf{\Theta}^m$ is specific for each fern. From these permutations we calculate the WTA hash, where the index of the largest value is determined from the first $K = 4$ dimensions in this example. The position index is used for the leaf assignment. Therefore it is encoded as binary vector. For an arbitrary window size K, the number of bits necessary for leaf assignment is $\lceil \log_2(K) \rceil$ using this binary vector procedure. For the given example with a window size of $K = 4$ we would need 2 bits to encode the index for each permutation. The binary vector is then used as leaf index by simply converting its numeric base. Formally the WTA hash is a non-linear mapping from the subspace to the hash space, thus $\mathbb{R}^T \rightarrow K^S$.

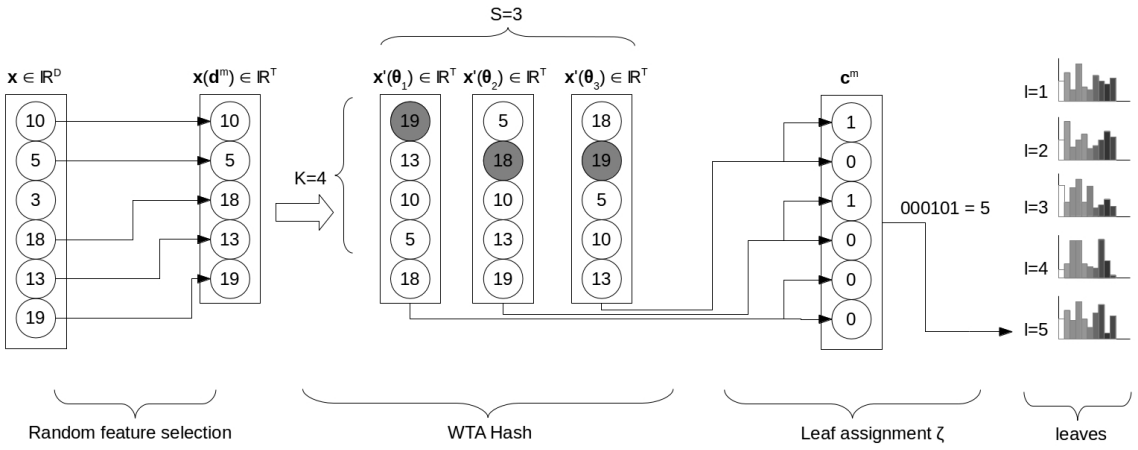


Figure 8: Illustration of our WTA Ferns depicting a single fern. In this example we use the WTA parametrization with $K = 4$, $S = 3$, $\boldsymbol{\theta}_1 = [5, 4, 1, 2, 3]$, $\boldsymbol{\theta}_2 = [2, 3, 1, 4, 5]$ and $\boldsymbol{\theta}_3 = [3, 5, 2, 1, 4]$.

If the window size K is not a power of 2, the leaf assignment is not optimal because the bit space is not fully used and several leaves would be empty. In order to save memory we do not use such a binary assignment as depicted here for simplicity, but a general mapping that directly assigns each hash code to a leaf. Thus a mapping function $\zeta : \mathbf{c} \rightarrow \mathcal{L}$ is used for leaf assignment which maps each hash code $\mathbf{c}$ to a leaf $\mathcal{L}_l$ and the leaf index l is calculated by

$$l = 1 + \sum_{i=1}^{S} (c_i - 1) K^{i-1} \tag{37}$$

with $l \in \{1, \dots K^S\}$.

If the subspace dimensionality $T = D$, thus the permutation matrix is calculated from all features in each fern, it directly maps the input sample $\mathbf{x} \in \mathbb{R}^D \rightarrow \mathcal{L} \in K^S$ without a fern-specific subspace limitation. The selection of the input feature dimensions is fully controlled by the random permutation matrix and thus dynamically sampled from all possible dimensions for each fern. For this case we can still assume independence between the features if the window size K is much smaller than the feature dimensionality D. The number of combinations for different windows is denoted by $\binom{D}{K}$ and if $K \ll D$ the probability that we sample the same combination multiple times is sufficiently small.

Training and testing works in the same manner as described in Section 3.1. For training, the random permutation vectors are created for each fern and from each sample the leaf assignment is calculated using the WTA hash. The occurrence of the training samples is counted for each leaf. For testing, the leaf assignment is calculated again for the test sample, but the same permutation matrix as used for training must be applied.

There exists a connection between our WTA Ferns and the original feature evaluation of [84] presented in Section 3.1. The original RFe can be seen as a specialized version of our WTA Ferns with a window size $K = 2$ and the random permutation vector $\boldsymbol{\theta}^i = \left[d^m_{i,1}, d^m_{i,2}\right], i = 1 \dots S$ with $\mathbf{d}^m$ being the fern-specific random feature dimensions. But instead of only encoding one inequality, each WTA code encodes $K-1$ inequalities. For example the index encoding of the maximum of the first permutation in Figure 8 satisfies $4-1=3$ inequalities, namely $x'_1 > x'_2$, $x'_1 > x'_3$, and $x'_1 > x'_4$. So for example by encoding 7 inequalities using the original feature formulation we would need 7 bits for the leaf assignment. For our WTA approach we would require a window size of $K = 8$ and thus only need $\log_2(K) = 3$ bits at most to encode the inequalities.

3.3.2 Maximum Order Ferns

Next we present our so called Maximum Order Ferns which provide a feature evaluation function that integrates well into the RFe framework as it provides binary features and encodes the position of the maximum within the subsampled input data. Therefore we first present the related feature evaluation function from the RFo and our adaptations, and then how it integrates into the RFe framework.

A method related to the WTA hash which uses ordinal node splits for a RFo was recently proposed by Schulter et al. [98]. They introduced a splitting function Ψ that incorporates partial order statistics as

$$\Psi_{\mathbf{D},\delta}(\mathbf{x}) = \begin{cases} 0, & \text{if } \left(\arg\max_{d=\{1\dots K\}} \mathbf{x}(\mathbf{D})_d\right) = \delta \\ 1, & \text{otherwise} \end{cases} \tag{38}$$

where $\mathbf{D}$ is a K-dimensional vector, which randomly samples the input $\mathbf{x}$. This leads to a K-dimensional vector $\mathbf{x}(\mathbf{D})$ whose index of the maximum is compared to a reference value $\delta \in \{1, 2, \dots K\}$ within a window of size K. If the dimension of the maximum value of $\mathbf{x}(\mathbf{D})$ matches δ, the sample is assigned to the left child node, otherwise to the right child node. During training the best splitting function is selected from a set of randomly sampled functions, i.e. random $\mathbf{D}$ and δ, by minimizing an impurity measure. Using this splitting function the commonality of the split data is higher, thus strengthening the splitting functions, where all samples within the child node fulfill $K-1$ inequalities.

This splitting function is too strict for our RFe as we do not sample the optimal splitting function from a set of possible ones. We only use a randomly chosen function in order to adhere to the RFe principle and to keep the runtime low. Especially for a large window size of $K \gg 2$ the function would evaluate to 0 for most samples. However, the criterion can be relaxed to fit. Instead of using the strict comparison $= \delta$ we use the inequality $\leq \delta$. Thus each leaf fulfills $K - \delta$ inequalities instead of $K-1$.
We construct our feature evaluation function f^m_s for fern m and feature s as

$$f^m_s = \begin{cases} 0, & \text{if } \left(\arg\max_{d=\{1\dots K\}} \mathbf{x}'^m_s(d)\right) \leq \delta^m_s \\ 1, & \text{otherwise} \end{cases} \tag{39}$$

where $\mathbf{x}'^m_s = \mathbf{x}(\boldsymbol{\theta}^m_s)$ a randomly permuted input sample with the feature-specific random permutation vector $\boldsymbol{\theta}^m_s$, and $\delta^m_s \sim \mathcal{U}(1, K)$ is a randomly chosen reference value within the window size K. For the order we assume total order that satisfies the binary relation $x_j < x_k \iff (x_j < x_k) \vee (x_j = x_k \wedge j < k)$. This method encodes more than one comparison into a single bit of the leaf index. Thus each bit of the leaf index is derived from a specific partial order.
This feature evaluation function integrates well into the Semi-Naive Bayesian approach presented in Section 3.1 because our ordinal feature evaluation function also results in a binary value. The Semi-Naive Bayesian feature combination in Equation 5 is still valid for this embedding; however, with our ordinal feature evaluation function. If $K = 2$ each feature encodes $2 - \delta$ inequalities, with $\delta \sim \mathcal{U}(0, 2)$, which corresponds to the original RFe [84] in average.

In order to illustrate the feature evaluation function we state an example. Given $\mathbf{x}(\boldsymbol{\theta}^m_s) = [1, 3, 5, 4, 2]$ with $K = 5$ the index of the maximum is 3. Assume $\delta = 3$ for the original splitting function Ψ which fulfills the $K - 1 = 4$ inequalities $x_{1,2,4,5} < x_3$. Our adapted function with $\delta \leq 3$ fulfills the inequalities $x_{2,3,4,5} < x_1$ or $x_{1,3,4,5} < x_2$ or $x_{1,2,4,5} < x_3$. Thus the inequality $x_{4,5} < x_{1,2,3}$ is fulfilled in all cases which make up $K-\delta = 5-3 = 2$ inequalities. Although we fulfill only $K-\delta$ inequalities the commonality of the samples within each leaf is still higher than using a single comparison as proposed for the original feature evaluation function. However, samples that do not fulfill this inequality constraint fulfill $\delta - 1$ other constraints, which are $x_4 > x_{1,2,3}$ or $x_5 > x_{1,2,3}$ and combined $x_{4,5} > x_{1,2,3}$. Thus the overall commonality is in average still higher than in the original RFe method.

The integration into our RFe framework is shown in Figure 9. First the input sample is subsampled from $\mathbb{R}^D \rightarrow \mathbb{R}^T$ using a random fern-specific subspace. Then we introduce randomness in two different ways. We generate S different random permutations using the permutation vectors $\boldsymbol{\theta}^m_s$ and we compare the index position of the maximum to the predefined random reference value $\delta_s \sim \mathcal{U}(1, K)$. Thus by evaluating f^m_s for different $\boldsymbol{\theta}^m_s$ and δ^m_s per assignment bit, a binary feature vector $\mathbf{F}^m = [f^m_1, f^m_2, \dots f^m_S]$ can be constructed. The binary vector is then used for leaf assignment by simply converting it into the leaf index. This feature evaluation function provides a non-linear mapping from $\mathbb{R}^T \rightarrow \mathcal{L}$. The number of leaves is $L = 2^S$.
The runtime for the feature evaluation method is $\mathcal{O}(SK)$ in order to determine the maximum within the first K dimensions once for each of the S permutations.

Although it is similar to the previously introduced WTA Ferns, it provides a different and more convenient parametrization, thus leading to different memory requirements. Similar to WTA Ferns different permutations are used, but the number of leaves is independent of the window size K and only depends on the number of permutations S. The window size K does not correlate with the number of leaves $L = 2^S$, thus K can be selected independent of S. This is in contrast to the WTA Ferns, where the window size K directly influences the number of leaves $L = K^S$.

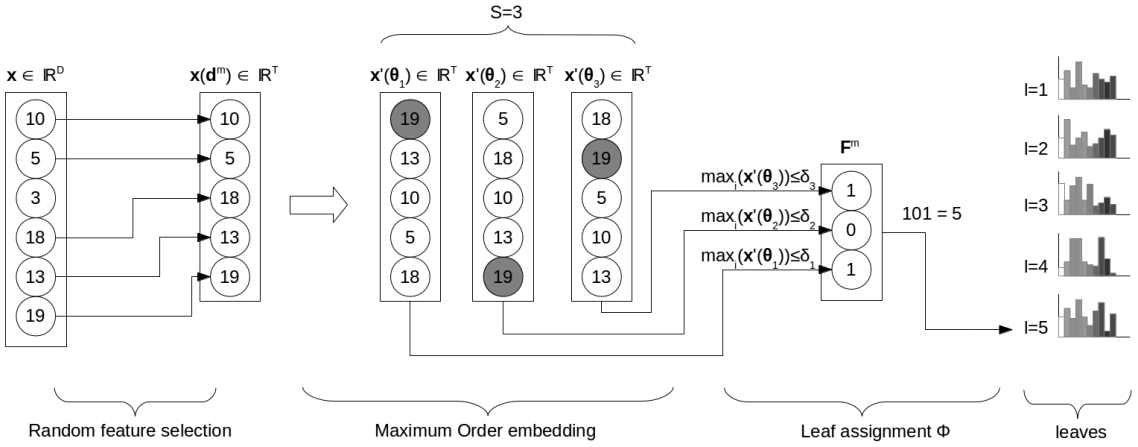


Figure 9: Illustration of our Maximum Order Ferns depicting a single fern. In this figure $\max_i$ denotes a function that returns the index of the maximum. The parametrization is $S = 3$, $K = 5$, $\delta_1 = 1$, $\delta_2 = 4$, $\delta_3 = 3$, and $\boldsymbol{\theta}_1 = [5, 4, 1, 2, 3]$, $\boldsymbol{\theta}_2 = [2, 3, 1, 4, 5]$ and $\boldsymbol{\theta}_3 = [3, 5, 2, 1, 4]$ for this example.

3.3.3 Direct Ordinal Ferns

Another method for incorporating ordinal information into the RFe framework is to use the order of the input data directly. Therefore we modify the Complete Rank Transform (CRT) proposed by Demetz et al. [18], which is a specialized approach of partial order, and integrate this feature transformation into our RFe framework. This ordinal feature transformation provides a non-linear mapping of the input data to the leaves, where each input feature set encodes multiple inequalities that uniquely define the order of the values in the set. This order is then used to assign samples to leaves. We first introduce some used mathematical definitions and then we show the integration of our modified CRT into the RFe framework.

For this type of Ferns classifier we use certain mathematical definitions that we introduce first. We assume, that we have given an ordinal space $\mathbb{S} = \{S_1, S_2, \dots S_Q\}$ with ordered items $S_1 \prec S_2 \prec \dots \prec S_Q$. In our case the ordinal space is spanned by all possible partial orders of the input data vector. The symbol $\prec$ (precedes) denotes the order between different ranks. A rank $\mathcal{R}_{\mathbb{S}}$ is an integer number in the range of $[1, Q]$ such that $\mathcal{R}_{\mathbb{S}}(S_j) = j$ for an ordinal item [38]. The rank depends on the set on which it is calculated. In order to clarify the used set, we denote it in the subscript as we calculate the rank for different sets. The rank defines a relationship between two items of the set $\mathbb{S}$, such that each item can be *ranked higher* $\prec$ or *ranked lower* $\succ$ than another item. In the mathematical order theory the precedence $\prec$ can be rewritten as $<$ because the items of the set, i.e. real numbers, are numerically comparable. The rank in lexicographical order[5] is simply the number of items that precede the item [79].

For leaf assignment in our Direct Ordinal Ferns we use a transformation of the input samples that integrates the partial order of the features within the input vector. Therefore

[5]Formally, lexicographical order on $\mathbb{N}^n$ is defined by $(x_1, \dots x_n) \leq (y_1, \dots y_n) \iff (x_1, \dots x_n) = (y_1, \dots y_n) \vee (\exists k \in \{1, \dots n\}\ with\ x_i = y_i : i < k \wedge x_k < y_k)$. [97]

we use a transformation similar to the CRT [18]. The original CRT transforms a $k \times k$ image neighborhood into a $K = k^2$ long vector $\boldsymbol{\sigma}$, where each element encodes the rank within the neighborhood with a value in the range of $[1, K]$. An example of this transformation is shown in Figure 10. Figure 10a shows an arbitrary neighborhood of a gray-scale image. The ranking is shown in Figure 10b where each element encodes the partial order within the neighborhood. In case of ties, both tied values are encoded with the same index in this example[6]. The ranking is transformed into a vector $\boldsymbol{\sigma}$ from top left to bottom right, thus forming the CRT vector (descriptor) for this neighborhood.

1	3	3
4	15	6
17	8	19

(a) Intensity values.

1	2	2
3	6	4
7	5	8

(b) Order.

$$\boldsymbol{\sigma} = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 3 \\ 6 \\ 4 \\ 7 \\ 5 \\ 8 \end{pmatrix}$$

(c) CRT.

Figure 10: Example of the original formulation of the CRT for a 3×3 neighborhood of a gray-scale image.

We generalize this transformation not only to pixel neighborhoods but to generic input data. Thus we do not define the neighborhood as direct pixel neighbors, but as an arbitrary combination of S random features selected by each fern. An example of this transformation is depicted in Figure 11 where $\mathbf{x}$ is the input vector and $\boldsymbol{\sigma}$ the generalized CRT vector.

$$\mathbf{x} = \begin{pmatrix} 10 \\ 3 \\ 3 \\ 6 \\ 17 \\ 8 \end{pmatrix}$$

(a) Input data.

$$\boldsymbol{\sigma} = \begin{pmatrix} 4 \\ 1 \\ 1 \\ 2 \\ 5 \\ 3 \end{pmatrix}$$

(b) CRT.

Figure 11: Example of our generalized CRT for an arbitrary input vector.

The integration of the generalized CRT into our RFe framework can be seen in Figure 12. Again, for each fern we randomly select the fern-specific subspace $\mathbb{R}^D \rightarrow \mathbb{R}^S$ and from this subspace we calculate the CRT as a feature transformation. Further we use

[6]The ties are encoded in a *dense* matter. Thus if a tie arises, the subsequently ranked value is encoded with the subsequent index number. Another encoding scheme would be the *competition* method, which skips the subsequent index number in case of ties. It is familiar from e.g. sports. Nevertheless this is simply a different representation that does not change the behavior.

the rank of the CRT vector to assign each of the CRT vectors to a leaf. In the following paragraphs we first discuss different order methods, next we show the probabilistic framework, and then we describe the leaf assignment of the CRT vectors.

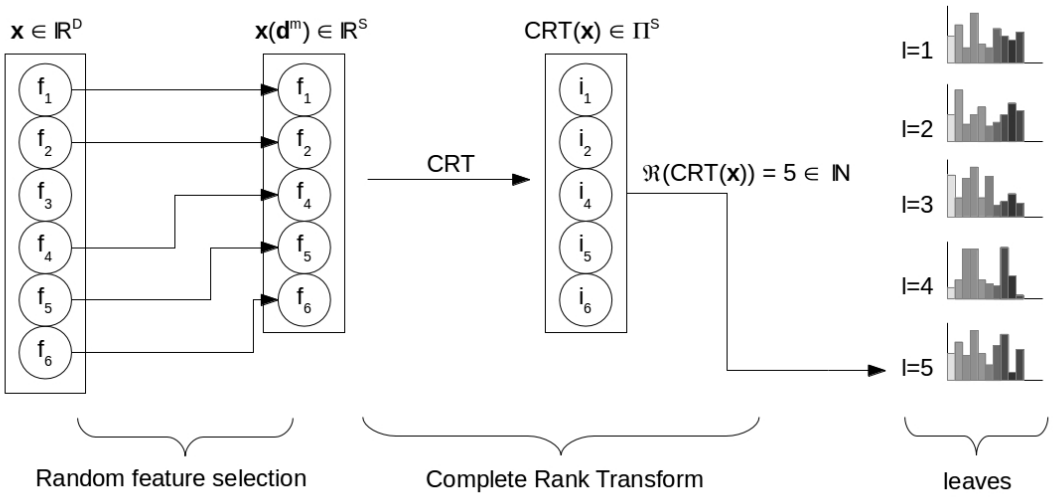


Figure 12: Illustration of a single fern of our Direct Ordinal Ferns which implements a generalized CRT as feature transformation and uses the rank of the CRT vector for leaf assignment.

In order to integrate this feature transformation into the RFe framework, we introduce a mapping that assigns each possible CRT vector to a leaf. Therefore we use the rank of the CRT vector, but before we focus on the leaf assignment, we discuss a problem that occurs if two input data values are the same. In that case a tie arises and the order is not explicitly defined anymore. In the previous example we solved this issue by assigning both features the same index, but there is a different solution. Thus we propose two different order methods that mitigate this problem, which are *weak* order and *total* order. The difference between the two order methods lies in the handling of ties. Total order uses the position of the tied values within the data vector to resolve the tie. Thus the first element which causes the tie is encoded with the lower position index and the second tied value with the higher index. Weak order encodes the tie in such a way, that both tied values are encoded with the same position index as already depicted in our example in Figure 11. Each of these encoded vectors makes up an individual rank. This effect is illustrated in Table 2. In the case of total order different input data vectors are assigned to the same rank, if ties occur. For the weak order, each possible tie is encoded in an individual rank.

As discussed, the original CRT is sensitive to ties within the order, i.e. Demetz et al. [18] use weak order which explicitly encodes ties. This consideration comes at the expense of an increased number of possible combinations. The number of possible combinations for total order is denoted by the factorial $S!$. For weak order the number of

Order method / Input data	Weak	Total
$\mathbf{x} = [10, 10, 10]$	$[1, 1, 1] \rightarrow \mathcal{R}_{\Pi_w^S} = 1$	$[1, 2, 3] \rightarrow \mathcal{R}_{\Pi_t^S} = 1$
$\mathbf{x} = [10, 10, 20]$	$[1, 1, 2] \rightarrow \mathcal{R}_{\Pi_w^S} = 2$	$[1, 2, 3] \rightarrow \mathcal{R}_{\Pi_t^S} = 1$
$\mathbf{x} = [10, 20, 10]$	$[1, 2, 1] \rightarrow \mathcal{R}_{\Pi_w^S} = 3$	$[1, 3, 2] \rightarrow \mathcal{R}_{\Pi_t^S} = 2$
$\mathbf{x} = [10, 20, 20]$	$[1, 2, 2] \rightarrow \mathcal{R}_{\Pi_w^S} = 4$	$[1, 2, 3] \rightarrow \mathcal{R}_{\Pi_t^S} = 1$
$\mathbf{x} = [10, 20, 30]$	$[1, 2, 3] \rightarrow \mathcal{R}_{\Pi_w^S} = 5$	$[1, 2, 3] \rightarrow \mathcal{R}_{\Pi_t^S} = 1$
$\mathbf{x} = [10, 30, 20]$	$[1, 3, 2] \rightarrow \mathcal{R}_{\Pi_w^S} = 6$	$[1, 3, 2] \rightarrow \mathcal{R}_{\Pi_t^S} = 2$
$\mathbf{x} = [20, 10, 10]$	$[2, 1, 1] \rightarrow \mathcal{R}_{\Pi_w^S} = 7$	$[3, 1, 2] \rightarrow \mathcal{R}_{\Pi_t^S} = 5$

Table 2: Comparison of different order methods for $S = 3$ features. The partial order is given in brackets and the ranks are stated for lexicographical order.

possible combinations is expressed by the n-th ordered Bell number [54]

$$\mathrm{OBN}(n) = \sum_{k=0}^{n} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n. \tag{40}$$

For example OBN(6) = 4683 and 6! = 720 for 6 features or OBN(7) = 47293 and 7! = 5040 for 7 compared to the original feature evaluation method of [84] with $2^6 = 64$ or $2^7 = 128$. This significantly increases the number of leaves for a given number of features.
Although this approach limits the number of compared features, for a similar number of leaves, e.g. 7! = 5040 and OBN(6) = 4683 compared to $2^{12} = 4096$, the complete order of the features within the data vector is mutually encoded. Thus each order enforces multiple inequalities, i.e. $x_{i_1} < x_{i_2} < \ldots x_{i_S}$, which makes the commonality of the data assigned to one leaf high.

As we have introduced the feature representation, we can model a probabilistic framework. By using the ranks as features we can formulate the Bayesian decision rule. From the subsampled input data $\mathbf{x}' = \mathbf{x}(\mathbf{d}^m)$ each feature is the rank of the value $\mathcal{R}_{\mathbf{x}'}(x'_i)$ within the input vector. Therefore we estimate the posterior probability as

$$\begin{aligned} &p(Y = y^i | \mathcal{R}_{\mathbf{x}'}(x'_1), \mathcal{R}_{\mathbf{x}'}(x'_2), \ldots, \mathcal{R}_{\mathbf{x}'}(x'_C)) \\ &\quad \propto p(\mathcal{R}_{\mathbf{x}'}(x'_1), \mathcal{R}_{\mathbf{x}'}(x'_2), \ldots, \mathcal{R}_{\mathbf{x}'}(x'_C) | Y = y^i) \end{aligned} \tag{41}$$

by using Bayes' Formula and neglecting the class-independent denominator and assuming an evenly distributed prior probability $p(Y)$. The ranks within the input vector are not independent from each other, as each rank mutually defines each other. We model this in our Semi-Naive-Bayesian framework, i.e.

$$p(\mathcal{R}_{\mathbf{x}'}(x'_1), \mathcal{R}_{\mathbf{x}'}(x'_2), \ldots, \mathcal{R}_{\mathbf{x}'}(x'_C) | Y = y^i) = \prod_{m=1}^{M} p(\pi^m | Y = y^i) \tag{42}$$

where $\pi^m = [\mathcal{R}_{\mathbf{x}'}(x'_1), \mathcal{R}_{\mathbf{x}'}(x'_2), \ldots, \mathcal{R}_{\mathbf{x}'}(x'_S)]$ is the permutation that is generated from the fern-specific feature subset, which models the feature combinations mutually. S dimensions are randomly selected, thus assuming independence between the input values

and each permutation π models the joint probabilities of the Semi-Naive Bayesian model, by defining their rank with respect to the other values. As the permutations are drawn from different subsets they can be regarded as independent.
The final classification rule, equivalent to Equation 7 can be stated as

$$\hat{y} = \arg\max_{y^i} \sum_{m=1}^{M} \log(p(\pi^m | Y = y^i)). \tag{43}$$

The permutations π can be efficiently assigned to the leaf by using their rank with respect to the permutation set $\mathcal{R}_{\Pi^S}(\pi)$ as we show next.
Formally, the total order transformation represents the order between values of an ordinal variable. The indices of a vector that resemble the order are a permutation of integers. Let Π_t^S denote the set of all permutations of the vector $[1, 2, \dots S]$. A specific order or permutation π is an element of this set. Further, a mapping $\sigma : \mathbf{x} \to \Pi_t^S$ assigns a S-dimensional vector $\mathbf{x}$ to a permutation $\pi \in \Pi_t^S$. In the case of the CRT the mapping σ is implemented as $\pi = [\mathcal{R}_\mathbf{x}(x_1), \mathcal{R}_\mathbf{x}(x_2), \dots, \mathcal{R}_\mathbf{x}(x_S)]$ where each element encodes the rank within the vector $\mathbf{x}$. A computationally less complex but equivalent method is provided by [118], which sorts all elements of $\mathbf{x}$ such that $x_{i_1} < x_{i_2} < \dots < x_{i_S}$ with the binary relation $x_j < x_k \iff (x_j < x_k) \vee (x_j = x_k \wedge j < k)$ specific for total order. The indices serve as permutation $\pi = [i_1, i_2, \dots, i_S]$. This can be simply implemented by sorting the input vector using stable sorting [55], thus maintaining the relative order of tied items, and using the rearrangement indices of the sorting as permutation π.
Further we define a mapping function which maps each of these $S!$ permutations directly to a leaf of a fern $\beta : \pi \to \mathcal{L}$ with $\beta \in \{1, 2, \dots S!\}$.
Similarly, the weak order transformation is a mapping of the S-dimensional input vector $\mathbf{x}$ into the set Π_w^S such that $\sigma : \mathbf{x} \to \Pi_w^S$ and $\Pi_w^S \subset \{1, \dots S\}^S$ is a set of all possible orders of S elements with repetitions, i.e. ties [18]. Again, all elements of $\mathbf{x}$ are sorted such that $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_S}$ but with the binary relation $x_i \leq x_k \iff (x_i < x_k) \vee (x_i = x_k)$ for weak order. The indices are used as permutation π, but in case of tied values the index number is shared $x_j = x_k \leftrightarrow i_j = i_k$. The target set for leaf assignment of $\beta \in \{1, 2, \dots \mathrm{OBN}(S)\}$ has an increased cardinality for weak order.

Practically, the mapping β, which assigns each possible permutation to a leaf $\mathcal{L}$, can be implemented by using a hashmap in $\mathcal{O}(1)$ [55], or by using ranking algorithms in lexicographical order in $\mathcal{O}(S^2)$ [68], or without special order in $\mathcal{O}(S)$ [79].
We use a hashmap for this mapping which is computationally less complex than the ranking based methods but requires a precedent initialization of the mapping. The ranking based methods do not require a previous initialization but they are slower for classification, thus we optimize the computational complexity for classification rather than for training. Therefore all possible permutations have to be enumerated in order to create the mapping. For weak order we use Cayley Trees [76] and for total order Heap's algorithm [100] to generate all possible permutations. The mapping assigns each permutation to a leaf index. Note, that the mapping must be the same for training and classification. For this purpose the mapping is established only once when creating the ferns. Thus the creation of the mapping does not influence the runtime of our RFe method for classification and only marginally for training. For training the complexity increases by an additional $\mathcal{O}(S!)$ or $\mathcal{O}(\mathrm{OBN}(S))$ as we need to enumerate all permutations, but as

the number of features S is small ($S \ll 10$) this does not increase the runtime. The complexity of the classification is slightly increased, because the order of the input data has to be established. The assignment of the order to the leaf is done in $\mathcal{O}(1)$ but the creation of the order requires a sorting of the subsampled input vector, thus $\mathcal{O}(S \log S)$.

Each rank $\mathcal{R}_{\Pi^S}(\pi)$ fulfills $\binom{S+1}{2}$ inequalities, thus specifying the positions of the values within the vector. Given the vector $\mathbf{x} = [10, 20, 40, 30]$ for example. Its ranking $\pi = [\mathcal{R}_\mathbf{x}(10), \mathcal{R}_\mathbf{x}(20), \mathcal{R}_\mathbf{x}(40), \mathcal{R}_\mathbf{x}(30)] = [1, 2, 4, 3]$. Each rank encodes $S - 1$ inequalities, as shown in Table 3 for our example but they are not independent from each other. The inequalities from the latter rows can be partially constructed from the previous ones. For $S = 4$ each permutation vector encodes 6 inequalities. Our Direct Ordinal Ferns need $L = 4! = 24$ leaves to encode these inequalities, the original RFe, however, requires $L = 2^6 = 64$ leaves for 6 inequalities.
If we choose $S = 2$ for our Direct Ordinal Ferns, they correspond to the original RFe formulation [84] with parameter $S = 1$ as both incorporate a single comparison.

$\mathcal{R}_\mathbf{x}(x_1)$	$x_1 < x_2$	$x_1 < x_3$	$x_1 < x_4$
$\mathcal{R}_\mathbf{x}(x_2)$	$x_1 < x_2$	$x_2 < x_3$	$x_2 < x_4$
$\mathcal{R}_\mathbf{x}(x_3)$	$x_1 < x_3$	$x_2 < x_3$	$x_4 < x_3$
$\mathcal{R}_\mathbf{x}(x_4)$	$x_1 < x_4$	$x_2 < x_4$	$x_4 < x_3$

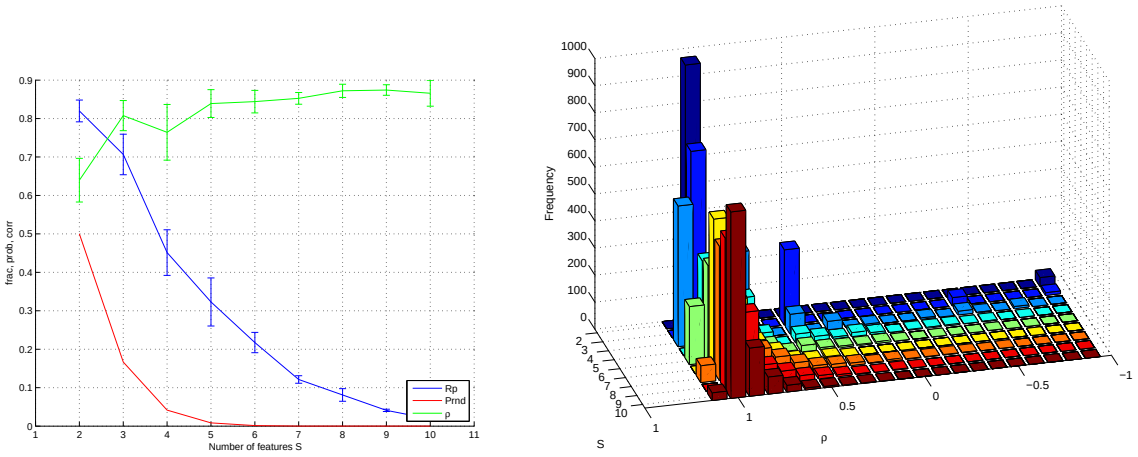
Table 3: Inequalities satisfied by each rank. Unique inequalities are colored.

3.3.4 Soft Assignment Ferns

The method of direct ordinal assignment is sensitive to noise, i.e. a perturbation of one feature value changes the complete partial order of the input features which leads to a different leaf assignment in the fern. Therefore we propose a soft assignment, which assigns different permutations π_i, π_j to the same leaf, if they do not excess a similarity measure threshold θ by $corr(\pi_i, \pi_j) < \theta$. As similarity measure we use a rank correlation measure $corr(\cdot, \cdot)$, which is e.g. Kendall τ or Spearman ρ [3]. This allows to tolerate perturbations of the input data.
Soft assignment is also used for example in the context of Naive Bayes classifiers, where a weighted version was proposed [30, 124] to increase the classification performance of Naive Bayes.

In order to show the influence of noise on the input data we perform a Monte Carlo simulation [39] with realistic input data and a Gaussian noise model. The inputs are $N = 1000$ randomly chosen patches from a gray-scale image from which S random feature dimensions are chosen. From this features we calculate the total order π_1, which specifies the leaf of the fern where the sample would be assigned. Next we generate randomly perturbed samples by adding Gaussian noise with zero mean and a standard deviation of 5% of the input feature magnitude to the original samples. From these perturbed input we calculate the total order π_2 and compare the total order of the perturbed input to the original total order using the Spearman ρ correlation measure. The result of this simulation is shown in Figure 13a. We show the fraction of rank preserving orders to all input orders $Rp = \frac{\#(\pi_1 = \pi_2)}{N}$, which decreases with the number of input features S. Further, $Prnd = \frac{1}{S!}$ is the probability that we randomly select the same order,

which is negligible compared to Rp in this case. At last we show the average correlation $\rho = \frac{1}{N}\sum_{i=1}^{N} corr(\pi_1^i, \pi_2^i)$, which does not fluctuate much. Although the rate of correctly matching orders decreases drastically, the overall correlation is still high. This can be observed in Figure 13b as well, which shows the distribution of the correlation measure for a different number S of input features. The noise perturbation itself does not change the order significantly, which can be seen by a high correlation, but if the order changes, the input data is still assigned to a different leaf of the fern.



(a) (b)

Figure 13: Evaluation of noise effects on the order. (a) depicts the Monte Carlo simulation of total order with Gaussian noise perturbation, showing the fraction of rank preserving orders Rp, the probability of random rank matching $Prnd$ and the average correlation measure ρ. (b) shows the histogram of the correlation measure ρ for a different number of features S.

In order to mitigate these noise problems with the input order we propose a soft assignment. For the integration of soft assignment into our RFe framework we use the already introduced Direct Ordinal Ferns with total order. It can be integrated for weak order as well, but instead of $S!$ we would have to consider $\mathrm{OBN}(S)$ different permutations and the correlation measure would have to be able to handle ties.
Recap Equation 42 introduced with the Direct Ordinal Ferns

$$p(\mathcal{R}_{\mathbf{x}'}(x'_1), \mathcal{R}_{\mathbf{x}'}(x'_2), \ldots, \mathcal{R}_{\mathbf{x}'}(x'_C)|Y = y^i) = \prod_{m=1}^{M} p(\pi^m|Y = y^i) \tag{44}$$

where we stated the Semi-Naive Bayesian estimate of the posterior probability with the feature representation π that represents the total order permutation. For this formulation we introduce the weight $w(\pi_i, \pi_j) \in \mathbb{R}^+$ to generate a soft assignment of the features depending on the rank correlation measure. The correlation measure acts as a distance between ranks, thus it is similar to different distance measures, as e.g. ℓ_2, but in ordinal space. A higher correlation denotes more similar permutations in ordinal space, which is in contrast to distance measures, where a smaller distance denotes more similar data in

Euclidean space. Thus we use the feature weight and rewrite the equation as

$$p(\mathcal{R}_{\mathbf{x}'}(x'_1), \mathcal{R}_{\mathbf{x}'}(x'_2), \ldots, \mathcal{R}_{\mathbf{x}'}(x'_C)|Y = y^i) = \prod_{m=1}^{M} \prod_{l=1}^{L} p(\pi_l^m|Y = y^i)^{w(\pi^m, \pi_l^m)} \tag{45}$$

where π^m is the permutation that is generated from the fern-specific feature subset. M is the number of ferns and L the number of leaves. This weighting scheme resembles the most general case of weighted Bayes, where each weight depends on each feature value [124]. The final classification rule, equivalent to Equation 7, can be stated as

$$\hat{y} = \arg\max_{y^i} \sum_{m=1}^{M} \sum_{l=1}^{L} w(\pi^m, \pi_l^m) \cdot \log\big(p(\pi_l^m|Y = y^i)\big). \tag{46}$$

We calculate the weight $w(\pi_i, \pi_j)$ between two permutations π_i and π_j by

$$w(\pi_i, \pi_j) = \frac{1}{Z} w'(\pi_i, \pi_j), \quad \text{and} \quad w'(\pi_i, \pi_j) = \begin{cases} 0, & \text{if } corr(\pi_i, \pi_j) < \theta \\ e^{\lambda \cdot corr(\pi_i, \pi_j)}, & \text{otherwise} \end{cases} \tag{47}$$

where θ and λ are two parameters which implement a threshold and an exponential backoff, and Z is a normalization factor. The weight is normalized to $[0, 1]$ such that

$$Z = \sum_{j=1}^{L} w'(\pi_i, \pi_j) \tag{48}$$

for each permutation π_i.
As correlation measure we use the Spearman ρ [3] rank correlation measure, which is calculated between two permutations π_1 and π_2 as

$$corr(\pi_1, \pi_2) = 1 - \frac{6}{S(S^2 - 1)} \sum_{i=1}^{S} (\pi_1^i - \pi_2^i)^2 \tag{49}$$

for S input features.
An example of the weights for different permutations is shown in Figure 14. We plot the weight $w(\pi, \pi_j)$ over all possible permutations π_j. More similar permutations get higher weights and less similar smaller weights. Due to the threshold θ most of the weights $w(\pi, \pi_j)$ become zero for permutations with low similarity.

Practically we use a weight vector $\mathbf{w} \in \mathbb{R}^L$ to calculate the coefficients for weighted class assignment. In order to speed classification up, we select $\mathbf{w}$ from a precalculated assignment matrix $\mathbf{W} \in \mathbb{R}^{L \times L}$, where $\forall \pi \in \Pi_t^S : \mathbf{W}(i, j) = w(\pi_i, \pi_j)$.

The complexity of this method has increased over the Direct Ordinal Ferns. For classification we establish the order of the input features thus requiring $\mathcal{O}(S \log(S))$ and further combine the probabilities which requires $\mathcal{O}(HL)$ with $L = S!$ for total order.

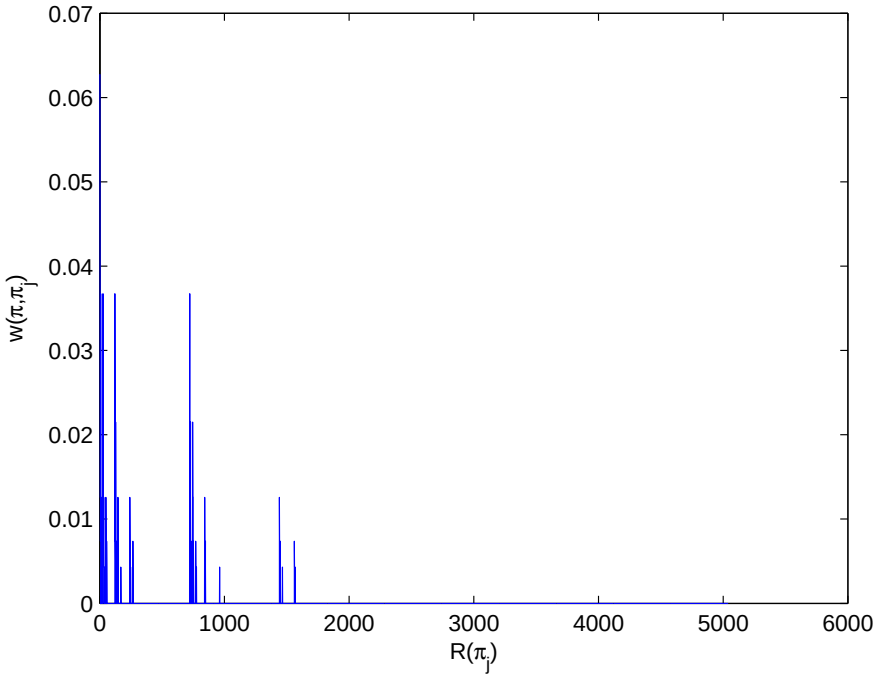


Figure 14: Weights for $S = 7$ features, $\theta = 0.7$ and $\lambda = 8$. $\pi = [1, 2, 3, 4, 5, 6, 7]$ and $\pi_j, j = 1 \dots S!$ are enumerated in lexicographic order with the rank as label on the x-axis.

3.3.5 Comparison of ordinal embeddings

We introduced our different ordinal embeddings above, where each of these embeddings incorporates the ordinal information in a different way. Therefore we provide a comparison showing the key properties of the proposed ordinal embeddings in Table 4.
In comparison to the subspace methods we do not have to consider any limitations of the subspace dimension, as they can be chosen through the parametrization for all of the ordinal embeddings. However, the ordinal methods have different memory and runtime requirements.

The overall memory complexity is unchanged with $\mathcal{O}(M \min(N, LH))$, but the number of leaves L depends on the embedding. For the WTA Ferns the number of leaves L depends on the window size K and the number of permutations S thus $L = K^S$. The Maximum Order Ferns can be parametrized similar to the original RFe, where the number of leaves directly depends on the number of feature evaluations $L = 2^S$ independent of any window size. For the Direct Ordinal Ferns we have to distinguish between total and weak order. For total order the number of leaves is factorial with $L = S!$ and for weak order $L = \mathrm{OBN}(S)$ for S features per fern. Thus the memory requirements significantly increase with the number of features per fern. As we use total order for our Soft Assignment Ferns there are $L = S!$ leaves.

The time complexity for a single sample and fern is also shown in Table 4. For the WTA and the Maximum Order Ferns the computational complexity depends on finding the maximum value within a window, thus the complexity is $\mathcal{O}(KS)$ for both. The Direct Ordinal Ferns require a sorting of the input subspace thus the runtime is $\mathcal{O}(S \log(S))$.

However, S has to be chosen small ($S \ll 10$) due to memory limitations. The complexity of Soft Assignment Ferns has drastically increased to $\mathcal{O}(HS!)$ over the Direct Ordinal Ferns due to the feature weighting scheme.

Two of the ordinal embeddings namely the WTA and the Maximum Order Ferns encode the maximum value, and the Direct Ordinal and Soft Assignment Ferns use the complete order of the input data. Although the maximum value does not contain as much information as the complete order of the data, it is more stable to perturbation of the data and faster to compute.

Ordinal embedding	Computational complexity	Space complexity
WTA	$\mathcal{O}(KS)$	$\mathcal{O}(M \min(N, HK^S))$
Maximum Order	$\mathcal{O}(KS)$	$\mathcal{O}(M \min(N, H2^S))$
Direct Ordinal total	$\mathcal{O}(S \log(S))$	$\mathcal{O}(M \min(N, HS!))$
Direct Ordinal weak		$\mathcal{O}(M \min(N, H \operatorname{OBN}(S)))$
Soft Assignment	$\mathcal{O}(HS!)$	$\mathcal{O}(M \min(N, HS!))$

Table 4: Comparison of different ordinal embeddings.

One can see from the comparison that the ordinal embeddings, except the Soft Assignment, are computationally less complex than the subspace embeddings, where the calculation of the projection is costly. Though the ordinal embeddings can require much memory. Especially the Direct Ordinal or Soft Assignment Ferns, where the memory is factorial in the number of features per fern, or the WTA embedding where a larger window size K also increases the memory requirement.

3.4 Subspace ordinal embeddings

At last we want to discuss a combination of our embeddings. One can think of a Ferns classifier for which we use ordinal embeddings in the subspace. Therefore the samples are projected into a subspace in which the ordinal analysis is performed. Instead of the threshold selection in subspace we use an ordinal embedding for leaf assignment and denote this *Subspace Ordinal Ferns*.

An illustration of a single Subspace Ordinal Fern is shown in Figure 15. The left part is similar to our Oblique Ferns. First the fern-specific input features are selected, thus $\mathbb{R}^D \to \mathbb{R}^T$. Then the selected features are projected onto the subspace $\mathbb{R}^T \to \mathbb{R}^S$ by using a linear projection $\mathbf{x}' = \mathbf{x}(\mathbf{d}^m)\mathbf{W}^m$, where $\mathbf{d}^m$ specifies the fern-specific input dimensions. In this subspace we use an ordinal embedding to assign the samples to the leaves $\mathbf{x}' \in \mathbb{R}^S \to \mathcal{L}$. The subspace projection as well as the ordinal embedding can be arbitrarily chosen as one of the methods we have already described in the previous sections. Thus the individual properties of this classifier depend on the chosen subspace method as well as on the ordinal embedding.

We have shown in Section 3.2.1 that the threshold selection is $\mathcal{O}(SN)$ for the median threshold and even larger for the optimal threshold. By using an ordinal embedding the computational complexity is significantly reduced, as training the leaf assignment does not depend on number of training samples N. This leads to a speedup for training but

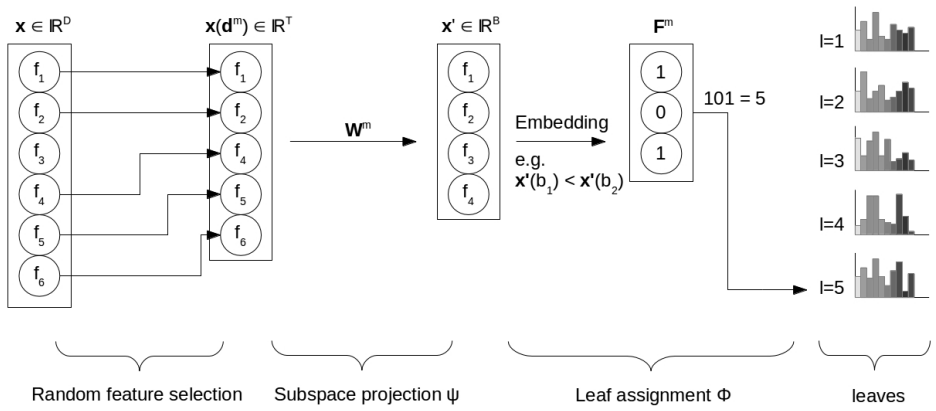


Figure 15: Illustration of our Subspace Ordinal Ferns depicting a single fern. The samples are first projected into the subspace and then an ordinal embedding is used for leaf assignment.

also to an increased classification time as the values for each assignment bit are not simply thresholded but ordinal analysis on the input is performed that may be more complex. However, the subspace projection of the training data is still required.

The training and testing is straightforward, by first projecting the samples on the subspace and then using the ordinal embedding to generate the leaf assignment.

There exists a variety of possible combinations, i.e. our 4 subspace methods with our 4 ordinal embeddings plus original feature evaluation summing up to 20 different possibilities. Each possible combination has an increased parameter space, which makes it hard to tune or cross-validate. Thus it is not feasible to evaluate all possible combinations. Therefore we select three of the possible combinations and evaluate them.

1. PCA subspace with original embedding: The PCA subspace can generate a high dimensional subspace. The original embedding is the simplest ordinal embedding, which only incorporates a single comparison per assignment bit. This combination exhibits similar computational complexity and memory requirements as the PCA Ferns.

2. PCA subspace with WTA embedding: Again, PCA is used to generate a high dimensional subspace. The WTA embedding resembles a more sophisticated ordinal embedding. The computational complexity and memory requirements are slightly increased due to the WTA embedding.

3. LDA subspace with WTA embedding: The LDA subspace provides a discriminative subspace for multi-class data in order to separate the classes, although with a reduced subspace dimensionality depending on the number of classes. We combine this with the WTA embedding, which again slightly increases the computational and memory complexity.

However, a fundamental problem exists for this type of Ferns. The subspace dimensionality B is limited by the subspace projection. But we need a number of independent

features in the subspace, in order to generate the leaf assignment. The generalized embedding provides a large number of independent features by choosing different thresholds. However, the ordinal embeddings as e.g. the original embedding can generate at most $\binom{B}{2}$ different features. With a subspace dimensionality of $B = 10$ we have 45 different features available at most. The WTA embedding can use $\binom{B}{K}$ different combinations. Thus for a window size of $K = B/2$ most different features can be created, though they are not completely independent. If $B = 1$ or $B = 2$, as it is given for CCA and LDA for two-class problems, no ordinal embeddings can be applied at all.

4 Evaluation

In this section we present and discuss the results of the evaluation. We evaluate each proposed Ferns classifier mentioned in Section 3. The evaluation is carried out on well-known computer vision and machine learning datasets. Further we evaluate the usability of our classifiers for 3D interest point recognition and we implement our classifiers for the task of pose tracking on planar surfaces, which was the original intention of the Random Ferns (RFe) classifier [84].

We evaluate on a computer with a Core 2 Duo with 2.6GHz and 4GB of RAM using Ubuntu 12.04 32bit. The implementation and evaluation of the different classifiers is carried out using Matlab 2012a in the style of Piotr Dollar's Image & Video Matlab Toolbox [20]. Selected classifiers are then integrated into an application for the task of pose tracking on planar surfaces, which is implemented in `C++` using the OpenCV 2.4.6 library[7]. For the calculation of the CCA we use the regularized least squares implementation of Sun et al. [107] in order to mitigate rank problem with non-full rank matrices.

4.1 Computer vision and machine learning datasets

First we evaluate the different classifiers on several common computer vision and machine learning datasets. The datasets used are shown in Table 5. Besides the dataset name, we denote the input data dimensionality D, the number of training samples N, the number of test samples M, the number of classes H, the distribution of the class labels, i.e. the fraction of the major and minor class samples, and the data type. For the data type we distinguish between numeric, categorical, and spectral (correlated numerical features) data. Machine learning datasets resemble the numeric and categorical datasets and the computer vision datasets can all be classified as spectral data.
Most datasets can be retrieved from the UCI machine learning repository[8] or from the respective authors website. All datasets are fully supervised, i.e. the full ground-truth is available. If the dataset has an explicit separation between training and test sets, we evaluate using these sets. Otherwise we evaluate using 10-fold cross-validation which is denoted by a "CV-10" in the test sample column.
We evaluate the classification accuracy on different datasets, different data representations (e.g. raw pixel values, descriptors, encodings), and different data categories in order to show the individual properties of our classifiers.

As evaluation criterion the average classification accuracy Acc on a K-fold cross-validation or on a test set is commonly used, e.g. in [78, 74], or the equivalent error $Err = 1 - Acc$, as e.g. in [9]. The objective is to maximize the accuracy of the classification, thus a higher classification accuracy denotes a better classifier.

[7] `opencv.org`
[8] `archive.ics.uci.edu/ml/`

Dataset name	D	N	M	H	Class distr. min/max	Data type
Australian [87]	14	690	CV-10	2	44.49/55.51%	num./cat.
Caltech 101 [25] †	12000	1515	1515	101	0.99/0.99%	spec.
Diabetes [104]	8	768	CV-10	2	34.9/65.1%	num.
Ionosphere [102]	34	351	CV-10	2	35.9/64.1%	num.
Iris [27]	4	150	CV-10	3	33.33/33.34%	num.
Isolet [24]	617	6238	1559	26	3.82/3.85%	num.
Fourclass [42]	2	862	CV-10	2	35.61/64.39%	num.
German [26]	24	1000	CV-10	2	30/70%	num./cat.
Leukemia [34]	7129	38	34	2	28.95/71.05%	num.
Liver disorders [28]	6	345	CV-10	2	42.03/57.97%	num.
MNIST [61] †	784	60000	10000	10	9.03/11.24%	spec.
Multi-PIE [37] †	254	54809	10000	150	0.21/0.76%	spec.
Mushroom [95]	112	8124	CV-10	2	48.2/51.8%	cat.
ORL [93] †	4096	400	CV-10	40	2.5/2.5%	spec.
Sonar [36]	60	208	CV-10	2	46.63/53.37%	num.
Spambase [44]	57	4601	CV-10	2	39.4/60.6%	num.
Splice [80]	60	1000	2175	2	48.3/51.7%	num.
SVMguide1 [48]	4	3089	4000	2	35.25/64.75%	num.
USPS [60] †	256	7291	2007	10	7.43/16.38%	spec.
Voting [95]	16	435	CV-10	2	38.62/61.38%	cat.
Vowels [19]	10	528	462	11	9.09/9.09%	num.
WDBC [106]	30	569	CV-10	2	37.25/62.75%	num.
WOBC [121]	9	683	CV-10	2	34.99/65.01%	cat.
Wine [1]	13	178	CV-10	3	26.96/39.88%	num.

Table 5: Overview with statistics of the datasets used for evaluation. D denotes the data dimensionality, N the number of training samples, M the number of test samples, and H the number of classes. † denotes computer vision datasets.

We conduct a K-fold cross-validation experiment as following.

1. First we randomly divide the dataset into K disjoint partitions, possibly equally sized, and with representative class distribution.
2. Then we use one partition to test and $K-1$ partitions to train the classifier.
3. At last we calculate the average classification accuracy as performance criterion.

The accuracy Acc_k of one fold can be calculated as

$$Acc_k = \frac{1}{N_k} \sum_{i=1}^{N_k} I(y^i = \hat{y}^i) \tag{50}$$

where y^i is the ground-truth class label, $\hat{y}^i$ the class label derived by our classifier, N_k the number of test samples of fold k, and

$$I(x) = \begin{cases} 1, & \text{if } x = \text{true} \\ 0, & \text{otherwise} \end{cases} \tag{51}$$

is the indicator function.
The average classification accuracy Acc over K folds is given by

$$Acc = \frac{1}{K} \sum_{i=1}^{K} Acc_i \tag{52}$$

with its standard deviation Acc_σ as

$$Acc_\sigma = \sqrt{\frac{1}{K-1} \sum_{i=1}^{K} (Acc_i - Acc)^2}. \tag{53}$$

In case there is an explicit separation between training and test set, we calculate the accuracy Acc over the M test samples by using

$$Acc = \frac{1}{M} \sum_{i=1}^{M} I(y^i = \hat{y}^i). \tag{54}$$

As we deal with randomly trained classifiers, we have to evaluate multiple trials to neglect random training effects. Therefore the training and evaluation step is executed 5 times[9] and the accuracy is averaged. Thus we state the average classification accuracy with its standard deviation.
For statistical significance among the results we use a two sided t-test with a confidence level of 95%, in order to test each method against the best or to compare our classifiers to the baseline [47]. We use the symbols $\bullet, \circ$ to show statistically significant improvement

[9]Due to the high computational effort of the evaluation we only use 5 repetitions. However, e.g. [74] or [9] use 100 repetitions. Nevertheless our evaluation shows that 5 repetitions already provide a good estimate as the variation of the classification accuracy is only minor for $M = 50$ base classifiers.

or degradation over the baseline, i.e. the original RFe [84], in the tables throughout the evaluation.

As baseline we use the original RFe classifier [84], and the related Random Forest (RFo) classifier [9], both implemented in [20] as well as a state-of-the-art linear Support Vector Machine (SVM) [22]. The linear SVM is a linear classifier that generates maximum margin hyperplanes for classification, i.e. it does not apply any kernel trick, which makes training a large amount of samples feasible. For the comparison of the runtime we do not state any timings explicitly, as the different methods are implemented in different environments (C, mex, Matlab) and thus a direct comparison would not be fair. Hence, we have already stated the computational complexity of the different classifiers in Section 3.

For all the stated evaluations we use a cross-validation over the classifier parameters in order to determine the best configuration among a set of parameters. For this cross-validation we use the full datasets. The baseline linear SVM is trained with L2-regularized L2-loss support vector classification and the penalty parameter C for the loss function is determined from $C \in \{2^{-15 \dots 15}\}$ as recommended in [22]. The input data for the SVM is normalized to zero mean and unit variance. The baseline RFo is fully trained without pruning for a depth of $S \in \{1 \dots 15\}$ in order to correspond with the RFe parameters, the number of features for each node split is $\sqrt{D}$, and the number of samples for training each tree is $N/2$. The parameters for the different Ferns classifiers are listed in Table 6. For the generalized RFe [20] the threshold value is uniformly sampled from $\tau_s \sim \mathcal{U}(-1, 1)$ and the input data is normalized to zero mean and scaled to an interval of $[-1, 1]$. For all RFe and the baseline RFo we keep the number of base classifier constant, i.e. $M = 50$. For the Oblique Ferns we use the median threshold for the evaluation because it is the faster method. However, we provide an evaluation of the different threshold methods separately. For the Subspace Ordinal Ferns we use a fixed $B = 0.7 \cdot T$ in order to reduce the parameter space for the sake of speed and a window size of $K = 4$ for the WTA embedding as for $K = 2$ it is equivalent to the original feature evaluation method. We do not evaluate our Subspace Ordinal Ferns on all datasets, as for most machine learning datasets the LDA subspace dimensionality is 1 for two-class problems and thus no ordinal leaf assignment is possible. Further they are a specialized type of Oblique Ferns and thus do not significantly differ from the Oblique Ferns, and due to the possibly many different combinations of subspace and embeddings it is computationally infeasible to evaluate them all. However, we show their performance on several selected datasets separately.
The parametrization is chosen such that the required memory is similar for all methods, because the memory is a limiting factor on our evaluation system. A different evaluation approach would consider the overall number of features that are used. Recall, that the overall features are grouped into the ferns structure depending on the degree of modeling the dependencies within the Semi-Naive Bayesian framework. Some methods cannot increase the number of features per fern above a certain limit, e.g. $S \leq 8$ for Direct Ordinal Ferns, as this would excess memory limitations. Thus they do not use as many overall features as others, which is derogatory for some datasets. However, we state the accuracy for the same number of overall features for selected examples.

We have separated the datasets of Table 5 into machine learning datasets and computer vision datasets. First we discuss the properties of our classifiers on the results of the machine learning datasets. Then we go into details for specific computer vision

Ferns classifier	Parameter space
Original [84]	$S \in \{1 \dots 15\}$
Generalized [20]	$S \in \{1 \dots 15\}$
Oblique (Random Oblique, PCA, CCA, LDA)	$S \in \{1 \dots 15\}$, $T \in \{\lceil D \cdot 2^{-1 \dots -7} \rceil\}$
WTA	$S \in \{1 \dots 15\}$, $K \in \{2 \dots 10\}$, $K^S \leq 32768$, $T \in \{\lceil D \cdot 2^{0 \dots -7} \rceil\}$
Maximum Order	$S \in \{1 \dots 15\}$, $K \in \{2 \dots 10\}$, $T \in \{\lceil D \cdot 2^{0 \dots -7} \rceil\}$
Direct Ordinal	total order $S \in \{1 \dots 8\}$ weak order $S \in \{1 \dots 7\}$
Soft Assignment	$S \in \{1 \dots 8\}$, $\theta = 0.7$, $\lambda = 8$
Subspace Ordinal	$S \in \{1 \dots 15\}$, $T \in \{\lceil D \cdot 2^{-1 \dots -7} \rceil\}$, $B = 0.7 \cdot T$, $K = 4$

Table 6: Parameter space used for determining the optimal parameter configuration. The number of ferns is $M = 50$ for all classifiers.

datasets, their problem formulations, and their specific characteristics. At last we analyze the Ferns classifiers with respect to their individual parameters and characteristics.

4.1.1 Machine learning

First we evaluate a number of common benchmark machine learning datasets in order to provide comparable results for other general machine learning methods.
The evaluation results are shown in Table 8. The RFo performs best on most datasets, which is undoubted as it applies a sophisticated feature and node test selection. However, this is computationally very intensive and it takes more time by a magnitude compared to the RFe. Nevertheless the RFe classifiers can outperform the RFo and the SVM on several datasets. For most of the datasets the accuracies are close together beyond any statistical significance. This fact has a lot to commend for the RFe classifier as it has a much lower computational complexity, i.e. it is faster for training and classification.
From the results it is hard to derive any conclusions on the performance on a specific data type of the dataset. The RFo performs best on several numeric datasets. The SVM performs best on numeric and on categorical data, although especially on two class problems, due to the dual class formulation of the support vectors. The original RFe is inferior to our proposed Ferns on all dataset, as either our Oblique or our Ordinal Ferns can outperform it.
The generalized RFe performs well on the Fourclass and WOBC dataset. It outperforms our other Ferns on most datasets with a small input data dimensionality. Due to the feature generation through random thresholds it can generate more features than the other Ferns, thus outperforming them for these datasets. However, it is inferior on all high dimensional datasets, because the random thresholds do not incorporate any data structure and thus the commonality of the data in the leaves is low.
For the WTA Ferns a window size of $K = 2$ works best for most datasets. They cannot leverage any additional informational from a larger window size and thus resemble an original RFe. The differences in the classification accuracies are due to random training effects. However, for the Spambase and Voting dataset a larger window size of $K = 8$

and $K = 6$ is used, which results in a significantly improved accuracy compared to the original RFe. The Voting dataset contains categorical features and uses 1-of-K coding. This creates a data property that can be well classified with a larger window size. The Spambase dataset contains histograms of word occurrences combined with others features. The histograms exhibit many zeros and by using a larger window size these zeros can be efficiently suppressed. Similar conclusion can be stated for the Maximum Order Ferns. For most of the datasets the used window size is small. However, for the Spambase and Voting datasets a window size of $K = 10$ is used, thus increasing the accuracy over the baseline original RFe. The inequality of the feature evaluation method enforces less comparisons of the data within the leaves, thus deteriorating the accuracy for most of the datasets compared to the original RFe or WTA Ferns.

The Direct Ordinal Ferns perform similar to the original RFe for all datasets. However, the number of features per fern is significantly lower than for the original RFe, because more inequalities are modeled with less features. The number of jointly modeled features depends on the dataset structure and stronger classifiers, i.e. modeling more joint features, are not beneficiary beyond a dataset specific point. For some of the machine learning datasets we obtain a small value for the number of features per fern S, or the accuracy does not change with S. This is due to independent features of several machine learning datasets, for which the Semi-Naive Bayesian approach is not beneficiary.

Our Soft Assignment Ferns perform worse than the Direct Ordinal Ferns for most datasets due to the lack of noise, which we explicitly assumed in the design. By using the soft assignment the conditional probabilities get "smoothed" over the leaves and thus loosing accuracy for non-noisy features. It might allow for fine tuning the two weighting parameters λ and θ, although this is very time consuming. The sole improvement over the Direct Ordinal Ferns is for the Isolet dataset, which is known to contain noisy feature values [24].

The Oblique Ferns perform very well on several datasets. The two unsupervised subspace projections random oblique and PCA perform similar on most datasets, though Random Oblique Ferns are trained faster because the projections are randomly chosen. The biggest differences occur on the Leukemia, Sonar, and WDBC datasets, where PCA Ferns work better on the prior two datasets and the Random Oblique Ferns on the last one. The Leukemia dataset is very sparse in a high dimensional space. In this case PCA acts more as dimensionality reduction than estimating the variances, yet PCA is a more reasonable dimensionality reduction method than random oblique directions, as it incorporates the data structure. The Sonar dataset is highly correlated with large variances, which especially benefits the PCA subspace, because it efficiently uses the directions of the empirical variances whereas random projections simply provide arbitrary projections with no data support and thus performing worse. The WDBC dataset has low dimensional and highly correlated data, but the different fern-specific PCA subspaces have too little variation between the projections, thus causing high correlation between the ferns. The random oblique projections provide different views for each fern, thus they generalize better resulting in a higher accuracy for this dataset.

The two supervised subspace methods CCA and LDA perform significantly different than their unsupervised counterparts on several datasets. Most of the machine learning datasets are two class problems, which derogates especially our CCA and LDA Ferns as their subspace dimensionality is limited by the number of classes. However, for some

datasets the underlying data structure benefits these methods such that we still obtain good results. Big differences are reported e.g. for the Isolet and Leukemia dataset, where in the first dataset LDA performs very well and in the second CCA. The Isolet dataset has a high number of classes thus providing a high subspace dimensionality for the LDA. Further, the data is well separable in the subspace that especially benefits the LDA method. For the Leukemia dataset CCA uses many input features to correlate them with the class labels, which works well for this sparse data. However, LDA performs worse for this dataset as it is very sparse and thus difficult to maximize inter-class distances due to lack of samples and the data is worse separable in the subspace. Advantage over the unsupervised projection methods is only given, if the data is well separable in subspace. Otherwise the supervised methods are inferior as e.g. shown on the Spambase dataset, which is a two-class dataset thus only providing a one-dimensional subspace with worse separability.

The last row of Table 8 shows the mean of the average classification accuracies over all datasets. The RFo clearly performs best due to its sophisticated, but time-consuming, feature selection. The generalized RFe perform best among the Ferns, as they do not imply any specific dataset properties. However, as described above, there are embeddings that work better for specific dataset properties.

Table 7 shows a ranking of the different Ferns classifiers. We rank the classifiers according to the number of datasets of Table 8 on which they perform best or on which the accuracy does not differ statistically significant from the best result. The generalized RFe rank overall first, as they perform best or similar to the best on 9 datasets. The WTA, Random Oblique, PCA, CCA, and LDA Ferns perform best or similar on 5 datasets thus ranking overall second, followed by the original RFe that perform best or similar on 3 datasets. Thus, when an out-of-the-box classifier for unknown dataset properties is required, the generalized RFe provide good results. However, if the dataset properties are known in advance, the accuracy can be significantly improved by choosing an adequate embedding.

Ranking	1	2	3
Overall	Gen. (9)	WTA, Random Obl., PCA, CCA, LDA (5)	Orig. (3)

Table 7: Ranking of the Ferns classifiers. The classifiers are ranked according to the number of datasets on which they performed best or similar to the best, which is stated in the brackets.

Dataset name	RFo [9]	Linear SVM [22]	Original RFe [84]	Generalized RFe [20]	WTA	Maximum Order
Australian	$86.67 \pm 0.48\%$	**87.10**%	$86.78 \pm 0.35\%$	$86.55 \pm 0.59\%$	$86.90 \pm 0.55\%$	$86.49 \pm 0.49\%$
Diabetes	$\mathbf{77.29 \pm 0.41}\%$	73.82%	$71.82 \pm 0.47\%$	$76.35 \pm 0.73\%$	$72.37 \pm 0.32\%$	$71.91 \pm 0.52\%$
Ionosphere	$\mathbf{93.34 \pm 0.32}\%$	89.17%	$90.20 \pm 0.55\%$	$91.81 \pm 0.74\%$	$92.77 \pm 0.32\%$	$88.38 \pm 1.22\%$
Iris	$\mathbf{96.13 \pm 0.55}\%$	86.00%	$82.53 \pm 0.30\%$	$95.73 \pm 1.12\%$	$83.73 \pm 0.53\%$	$82.67 \pm 0.00\%$
Isolet	$\mathbf{93.59 \pm 0.24}\%$	91.35%	$85.98 \pm 1.21\%$	$86.81 \pm 1.70\%$	$87.35 \pm 0.48\%$	$85.93 \pm 0.57\%$
Fourclass	$\mathbf{99.58 \pm 0.19}\%$	70.77%	$68.91 \pm 0.00\%$	$99.00 \pm 0.18\%$	$68.91 \pm 0.00\%$	$68.91 \pm 0.00\%$
German	$76.08 \pm 0.96\%$	70.20%	$74.46 \pm 0.42\%$	$75.32 \pm 0.55\%$	$75.18 \pm 0.34\%$	$75.36 \pm 0.58\%$
Leukemia	$80.00 \pm 4.36\%$	**91.18**%	$78.82 \pm 3.83\%$	$68.82 \pm 1.61\%$	$80.59 \pm 7.35\%$	$72.35 \pm 3.35\%$
Liver disorders	$\mathbf{73.00 \pm 0.69}\%$	65.50%	$69.35 \pm 0.62\%$	$64.16 \pm 1.54\%$	$69.28 \pm 1.28\%$	$69.41 \pm 0.53\%$
Mushroom	$\mathbf{100.00 \pm 0.00}\%$	**100.00**%	$99.99 \pm 0.01\%$	$98.74 \pm 0.30\%$	$99.98 \pm 0.01\%$	$99.97 \pm 0.02\%$
Sonar	$83.28 \pm 1.27\%$	77.36%	$83.47 \pm 1.05\%$	$80.30 \pm 2.06\%$	$82.80 \pm 1.33\%$	$81.06 \pm 1.18\%$
Spambase	$\mathbf{94.20 \pm 0.10}\%$	92.02%	$91.56 \pm 0.17\%$	$82.68 \pm 0.96\%$	$93.88 \pm 0.08\%$	$92.71 \pm 0.13\%$
Splice	$\mathbf{96.50 \pm 0.43}\%$	85.47%	$82.02 \pm 0.99\%$	$87.63 \pm 3.95\%$	$83.20 \pm 0.49\%$	$80.99 \pm 1.31\%$
SVMguide1	$\mathbf{97.17 \pm 0.15}\%$	84.88%	$83.65 \pm 1.00\%$	$94.22 \pm 0.78\%$	$84.48 \pm 0.05\%$	$84.50 \pm 0.00\%$
Voting	$\mathbf{96.14 \pm 0.25}\%$	95.17%	$93.23 \pm 0.45\%$	$93.79 \pm 0.96\%$	$95.03 \pm 0.57\%$	$94.99 \pm 0.44\%$
Vowels	$\mathbf{56.84 \pm 2.48}\%$	33.55%	$43.81 \pm 1.11\%$	$53.38 \pm 1.72\%$	$44.50 \pm 0.86\%$	$42.94 \pm 1.49\%$
WDBC	$96.17 \pm 0.19\%$	**98.25**%	$92.97 \pm 0.56\%$	$94.38 \pm 0.28\%$	$93.92 \pm 0.49\%$	$92.89 \pm 0.59\%$
WOBC	$97.28 \pm 0.27\%$	97.22%	$94.88 \pm 0.21\%$	$\mathbf{97.33 \pm 0.19}\%$	$95.23 \pm 0.20\%$	$94.50 \pm 0.26\%$
Wine	$97.86 \pm 0.26\%$	98.33%	$94.95 \pm 0.80\%$	$97.56 \pm 0.30\%$	$96.05 \pm 0.53\%$	$95.85 \pm 0.64\%$
Average	89.00%	83.54%	82.59%	85.50%	83.48%	82.20%

Table 8: Evaluation results for different classifiers and embeddings. Values show the average classification accuracy with its standard deviation $Acc \pm Acc_\sigma$. (continued on next page, part 1 of 2)

Dataset name	Direct Ordinal		Soft Assignment	Oblique			
	Total	Weak		Random	PCA	CCA	LDA
Australian	$86.84 \pm 0.28\%$	$86.61 \pm 0.39\%$	$85.57 \pm 0.57\%$	$85.10 \pm 0.49\%$	$84.09 \pm 0.28\%$	$86.70 \pm 0.75\%$	$86.55 \pm 0.43\%$
Diabetes	$69.19 \pm 0.63\%$	$70.52 \pm 0.37\%$	$68.54 \pm 0.43\%$	$72.03 \pm 0.61\%$	$72.97 \pm 0.44\%$	$72.55 \pm 0.70\%$	$72.18 \pm 0.38\%$
Ionosphere	$89.87 \pm 0.99\%$	$90.16 \pm 0.82\%$	$86.56 \pm 0.61\%$	$89.02 \pm 0.55\%$	$88.27 \pm 0.37\%$	$80.40 \pm 0.59\%$	$78.92 \pm 0.45\%$
Iris	$82.67 \pm 0.00\%$	$82.67 \pm 0.00\%$	$81.33 \pm 0.00\%$	$87.60 \pm 1.53\%$	$88.27 \pm 1.01\%$	$91.73 \pm 0.76\%$	$83.20 \pm 1.19\%$
Isolet	$84.17 \pm 0.61\%$	$84.93 \pm 1.26\%$	$86.38 \pm 0.96\%$	$86.48 \pm 1.15\%$	$86.35 \pm 1.07\%$	$85.59 \pm 1.15\%$	$91.49 \pm 0.47\%$
Fourclass	$68.91 \pm 0.00\%$	$68.91 \pm 0.00\%$	$68.91 \pm 0.00\%$	$71.46 \pm 0.00\%$	$71.50 \pm 0.10\%$	$71.99 \pm 0.42\%$	$71.46 \pm 0.00\%$
German	$75.00 \pm 0.80\%$	$74.26 \pm 0.98\%$	$39.42 \pm 1.10\%$	$74.82 \pm 1.15\%$	$74.70 \pm 0.61\%$	$\mathbf{76.12 \pm 0.40}\%$	$75.66 \pm 0.65\%$
Leukemia	$74.12 \pm 2.46\%$	$68.24 \pm 8.16\%$	$71.76 \pm 4.46\%$	$71.18 \pm 7.32\%$	$82.94 \pm 2.46\%$	$90.00 \pm 3.35\%$	$66.47 \pm 3.35\%$
Liver disorders	$68.51 \pm 0.53\%$	$69.33 \pm 1.05\%$	$68.69 \pm 0.82\%$	$69.13 \pm 0.95\%$	$69.05 \pm 0.55\%$	$64.14 \pm 1.30\%$	$65.89 \pm 0.87\%$
Mushroom	$98.13 \pm 0.26\%$	$97.81 \pm 0.20\%$	$97.42 \pm 0.27\%$	$\mathbf{100.00 \pm 0.00}\%$	$\mathbf{100.00 \pm 0.00}\%$	$98.86 \pm 0.11\%$	$99.86 \pm 0.03\%$
Sonar	$80.67 \pm 3.06\%$	$79.53 \pm 1.43\%$	$78.38 \pm 2.92\%$	$85.50 \pm 2.63\%$	$\mathbf{87.35 \pm 0.91}\%$	$79.89 \pm 0.40\%$	$79.20 \pm 0.86\%$
Spambase	$90.68 \pm 0.23\%$	$90.30 \pm 0.33\%$	$88.13 \pm 0.43\%$	$93.56 \pm 0.13\%$	$93.69 \pm 0.17\%$	$88.49 \pm 0.23\%$	$88.40 \pm 0.04\%$
Splice	$83.78 \pm 1.94\%$	$84.34 \pm 1.37\%$	$82.91 \pm 2.52\%$	$86.97 \pm 1.53\%$	$85.38 \pm 1.64\%$	$85.64 \pm 0.41\%$	$85.62 \pm 0.33\%$
SVMguide1	$84.50 \pm 0.00\%$	$84.50 \pm 0.00\%$	$84.50 \pm 0.00\%$	$91.16 \pm 1.32\%$	$92.19 \pm 0.31\%$	$92.03 \pm 0.67\%$	$92.19 \pm 0.31\%$
Voting	$93.15 \pm 0.75\%$	$93.88 \pm 0.31\%$	$89.93 \pm 0.55\%$	$92.22 \pm 0.41\%$	$93.84 \pm 0.19\%$	$93.50 \pm 0.19\%$	$93.27 \pm 0.34\%$
Vowels	$42.03 \pm 1.41\%$	$43.12 \pm 2.00\%$	$40.13 \pm 0.50\%$	$46.67 \pm 1.96\%$	$47.49 \pm 1.74\%$	$45.15 \pm 2.97\%$	$44.20 \pm 2.95\%$
WDBC	$93.04 \pm 0.46\%$	$92.97 \pm 0.48\%$	$91.28 \pm 0.27\%$	$94.84 \pm 0.36\%$	$92.87 \pm 0.44\%$	$94.20 \pm 0.48\%$	$91.35 \pm 0.23\%$
WOBC	$95.08 \pm 0.34\%$	$95.49 \pm 0.47\%$	$93.32 \pm 0.71\%$	$96.32 \pm 0.06\%$	$95.93 \pm 0.38\%$	$95.70 \pm 0.25\%$	$95.61 \pm 0.46\%$
Wine	$93.35 \pm 0.63\%$	$94.50 \pm 0.91\%$	$92.70 \pm 1.66\%$	$96.65 \pm 0.89\%$	$96.44 \pm 0.63\%$	$\mathbf{98.66 \pm 0.64}\%$	$97.67 \pm 0.82\%$
Average	81.77%	81.68%	78.72%	83.72%	84.38%	83.75%	82.06%

Table 8: Evaluation results for different classifiers and embeddings. Values show the average classification accuracy with its standard deviation $Acc \pm Acc_\sigma$. (continued, part 2 of 2)

Categorical data. Further we evaluate different encoding schemes for categorical data, although they are only relevant for machine learning datasets. There are different possibilities to encode categorical features. The two most common methods include 1-of-K binary coding [4], where each categorical feature is split into K binary features, or numeric coding, such that each category makes up a natural number in the range of $[1, K]$. Hence, we evaluate these two representations on the Australian and German dataset, which contain several categorical features. The 1-of-K coding increases the dimensionality from $D = 14$ for the Australian dataset to $D = 42$ and from $D = 24$ to $D = 49$ for the German dataset. The used parameters are acquired using a cross-validation over all parameters for the numeric encoding and kept for the 1-of-K encoding. The Direct Ordinal Ferns are used with total order. We evaluate 5 trails and report average classification accuracy with its standard deviation.

The results are shown in Table 9. The PCA Ferns and original RFe perform significantly better for 1-of-K encoding for both datasets. The PCA Ferns benefit from the lifting to a higher dimension, as they can get more variance from the different dimensions. For the numeric coding all variance is along one dimension, but for the 1-of-K coding the variance is spread over different dimensions and thus it is possible to derive a better subspace from a single feature. Further, a higher number of different subspaces is possible, which reduces the correlation between the different ferns.
The original RFe generates features from comparing two dimensions. If all information is encoded in one dimension by K values no comparison is possible, but when spread in different dimensions, comparisons from a single original feature are possible.
The WTA and Maximum Order Ferns, that both rely on the position of the maximum value, perform inferior for 1-of-K coding, because all values are 0 or 1, thus always selecting the first 1. The same applies for the Direct Ordinal Ferns, as the order is not distinctive with all values 0 or 1.

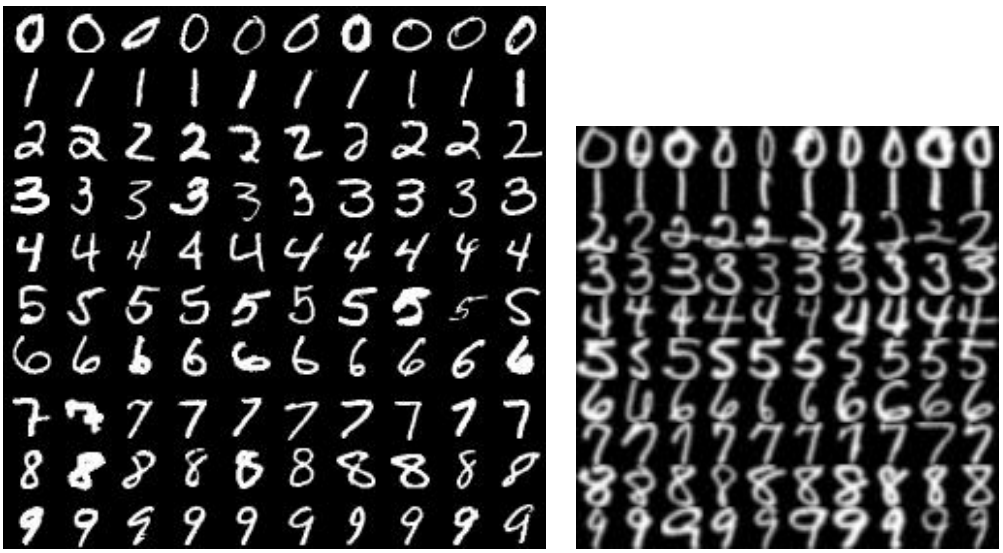
Classifier	Australian		German	
	numeric	1-of-K	numeric	1-of-K
RFo [9]	$86.67 \pm 0.48\%$	$87.07 \pm 0.48\%$	$76.08 \pm 0.96\%$	$75.72 \pm 0.91\%$
Linear SVM [22]	87.10%	86.38%	70.20%	69.90%
Original RFe [84]	$86.78 \pm 0.35\%$	$\mathbf{87.59 \pm 0.36}\%$	$74.46 \pm 0.42\%$	$\mathbf{75.12 \pm 0.43}\%$
Generalized RFe [20]	$86.55 \pm 0.59\%$	$86.49 \pm 0.85\%$	$\mathbf{75.32 \pm 0.55}\%$	$74.06 \pm 0.44\%$
Random Oblique	$85.10 \pm 0.49\%$	$\mathbf{86.78 \pm 0.64}\%$	$74.82 \pm 1.15\%$	$75.14 \pm 0.98\%$
PCA	$84.09 \pm 0.28\%$	$\mathbf{87.04 \pm 0.45}\%$	$74.70 \pm 0.61\%$	$\mathbf{75.90 \pm 0.74}\%$
CCA	$86.70 \pm 0.75\%$	$86.52 \pm 0.40\%$	$\mathbf{76.12 \pm 0.40}\%$	$75.08 \pm 0.66\%$
LDA	$86.55 \pm 0.43\%$	$86.58 \pm 0.17\%$	$\mathbf{75.66 \pm 0.65}\%$	$74.54 \pm 0.53\%$
WTA	$86.90 \pm 0.55\%$	$87.54 \pm 0.57\%$	$\mathbf{75.18 \pm 0.34}\%$	$74.46 \pm 0.23\%$
Maximum Order	$86.49 \pm 0.49\%$	$\mathbf{87.51 \pm 0.40}\%$	$\mathbf{75.36 \pm 0.58}\%$	$74.48 \pm 0.45\%$
Direct Ordinal	$86.84 \pm 0.28\%$	$86.61 \pm 0.63\%$	$75.00 \pm 0.80\%$	$74.30 \pm 0.61\%$

Table 9: Evaluation results for different categorical data encoding schemes showing the average classification accuracy with its standard deviation. Statistically significant improvement/degradation is in boldface.

4.1.2 Digit recognition

The first specific computer vision problem that we analyze is digit recognition. This task is a common benchmark for machine learning algorithms, not only in the field of computer vision. In pure machine learning it is often only used to distinguish between two digits (classes), as e.g. in [74], but we perform real classification among all classes. Therefore we use the two most common datasets MNIST and USPS for evaluation.

MNIST. The MNIST dataset [61] is a challenging dataset for handwritten digit recognition. It consists of a training set of 60 000 samples and a test set of 10 000 samples. Each sample is assigned a class $y \in \{0, 1, \dots 9\}$. Each digit is size-normalized and centered in a $28 \times 28px$ image. We use the sample images without preprocessing and transform each image into a 728-dimensional feature vector $\mathbf{x} \in \mathbb{R}^{728}$, which is then used for training and testing. The background of the images is black with the digits in white, but with gray levels at the borders due to aliasing effects. Random sample images are shown in Figure 16a for each class.



(a) MNIST samples each $28 \times 28px$. (b) USPS samples each $16 \times 16px$.

Figure 16: Sample images of digit recognition datasets.

For the evaluation we compare two different data representations.

1. We use the raw pixel values as input data for the classifier, which is the most simple representation.

2. As we have a PCA compressed version of the MNIST dataset readily available, we also analyze this data representation which is commonly used and thus worth discussing. We apply PCA dimensionality reduction on the full dataset and retain the 164 dimensions corresponding to the major principal components, which capture 95% of the variance of the original data.

The evaluation results for the MNIST dataset are shown in Table 10. We show the different classifiers as well as the different data representations. On the raw data our

proposed WTA Ferns as well as the Random Oblique and PCA Ferns clearly outperform the baseline RFe. However, the best performing classifier is the RFo, because it employs a more sophisticated feature selection at each node split.

The reason for these results can be found in the data representation. As one can see on the sample images, most of the image is black, i.e. the background pixel values are all zero. 80.88% of all pixels are zero and the dataset contains 67 zero-columns (out of 784), i.e. pixels that are black in all images resemble a zero-column in the vectorized dataset representation. This has an effect on all methods when the raw pixel values are used as input, that should not be underestimated. Especially some of the subspace methods suffer from this representation. The CCA is very sensitive to those zero-columns, because they reduce the rank of the data matrix and thus lead to numeric problems. Therefore a regularization of the CCA [63] is necessary. A look on the optimal parameters shows, that many input dimensions are used, e.g. the subsampled input dimensionality is $T = 32$ for the PCA Ferns for $S = 15$ features thus $T > 2 \cdot S$. This increases the chance of selecting non-zeros columns as features in the random input dimension selection process. The RFo on the other hand does not suffer from such problems, because it optimizes each node split and avoids using such zero-columns as they do not contain any discriminative information.

All zero values within the input sample resemble a tie for our Direct Ordinal Ferns. This is handled differently for weak and total order. Although the weak order explicitly encodes these ties, it performs not significantly different. We will discuss and analyze this in detail in Section 4.1.5. The WTA Ferns provide a high accuracy, because encoding the position of the maximum value is more tolerant to those zeros, as they can be suppressed by simply using any value greater than 0. Therefore the WTA uses a large window size of $K = 8$, which increases the probability of selecting non-zero values within the window. The Maximum Order Ferns behave similar, where the best results are obtained with $K = 6$. Though if the window size K decreases, the number of features per fern S increases within the parameter space in order to model the dependence between the features and to generate stronger classifiers.

As there are only 10 different digits, both datasets contain 10 classes. This limits the subspace dimensionality of both the CCA and LDA Ferns. A comparison of these methods that provides the same number of features would thus limit the number of used features per fern to $S = 9$. For $S = 9$ our baseline original RFe performs at $87.19 \pm 0.97\%$ on the raw pixels. Thus the CCA and LDA Ferns significantly outperform the baseline for the same number of features, which shows that the features are more discriminative in the subspace and provide a more efficient embedding. For comparison with the Direct Ordinal Ferns, the baseline RFe for the same number of features $S = 7$ performs at $85.27 \pm 0.66\%$. Thus the Direct Ordinal Ferns slightly outperform the baseline for the same number of features, due to the stronger base classifiers that encode more inequalities per feature.

Classifier	Raw pixels		PCA	
	Acc	Parameters	*Acc*	Parameters
RFo [9]	$\mathbf{95.04 \pm 0.11}\%$	$S = 15$	$\mathbf{94.92 \pm 0.13}\%$	$S = 15$
Linear SVM [22]	91.73%	$C = 2^0$	18.19%	$C = 2^{-15}$
Original RFe [84]	$90.85 \pm 0.46\%$	$S = 15$	$86.95 \pm 0.76\%$	$S = 10$
Generalized RFe [20]	$87.19 \pm 0.46\%$	$S = 15$	$77.87 \pm 3.23\%$	$S = 15$
Random Oblique	$91.65 \pm 0.30\%\bullet$	$S = 15$, $T = 49$	$87.16 \pm 0.75\%$	$S = 10$, $T = 11$
PCA	$92.02 \pm 0.23\%\bullet$	$S = 15$, $T = 49$	$86.06 \pm 1.58\%$	$S = 10$, $T = 11$
CCA	$90.78 \pm 0.29\%$	$S = 10$, $T = 25$	$93.34 \pm 0.15\%\bullet$	$S = 10$, $T = 82$
LDA	$89.07 \pm 0.53\%\circ$	$S = 9$, $T = 25$	$92.47 \pm 0.04\%\bullet$	$S = 9$, $T = 41$
WTA	$92.50 \pm 0.26\%\bullet$	$S = 5$, $K = 8$, $T = 784$	$87.64 \pm 0.54\%$	$S = 11$, $K = 2$, $T = 164$
Maximum Order	$90.75 \pm 0.49\%$	$S = 15$, $K = 6$, $T = 784$	$83.74 \pm 0.54\%\circ$	$S = 15$, $K = 2$, $T = 784$
Direct Ordinal	$86.43 \pm 1.46\%\circ$	$S = 7$, total order	$80.28 \pm 2.68\%\circ$	$S = 5$, total order
	$85.84 \pm 0.85\%\circ$	$S = 7$, weak order	$81.91 \pm 2.23\%\circ$	$S = 5$, weak order

Table 10: Evaluation results for the MNIST dataset comparing different embeddings by showing the average classification accuracy with its standard deviation $Acc \pm Acc_\sigma$.

Further we stated the results for a different representation, a PCA compression of the dataset, which shows other interesting properties of the classifiers. On the PCA compressed data all but the LDA and CCA Ferns perform worse than on the raw pixel data.
Both the LDA and CCA Ferns provide good classification performance. These subspace methods require independent variables as input and the class label as dependent variable for which the transformation is calculated. The prior PCA transformation provides a set of orthogonal, thus independent [92], variables that are used as input for the classifiers. Therefore they perform well for the PCA input. When used on the original input data, this independence assumption is not fulfilled for all input features, thus only providing suboptimal results.
The Random Oblique and PCA Ferns perform bad in this case. As already described in Section 3.2.3, the PCA projection creates an orthogonal subspace from the correlated input values, that resembles its principal components, i.e. captures most of the variance of the data. When applying the PCA again on this data, no correlations can be calculated, as all feature dimensions are orthogonal, i.e. uncorrelated and the new principal components coincide with the previous ones if $S \approx T$. If $S \not\approx T$ the repeated PCA transformation drops random feature dimensions from the subspace, thus the results are much worse ($< 80\%$ accuracy). Thus for $S \approx T$ the PCA subspace projection becomes close to identity and a classification similar to the generalized RFe is performed, which simply thresholds the data values for leaf assignment. The same holds for the Random Oblique Ferns, where we use different random projections, which mix the orthogonal dimensions similar to PCA. For this dataset the PCA dimensionality reduction creates a highly non-linear separable subspace that makes the classification extremely difficult for a linear classifier, which can be seen for the linear SVM. The principal components that provide high variance in the full dataset may not contain any discriminative information which leads to worse classification results.
On the PCA data representation all of the Ordinal Ferns perform worse. From the numeric values no meaningful classification hypothesis can be derived. A comparison of two dimensions is not necessarily meaningful in the subspace as in this subspace the dimensions might be arbitrarily ordered.

USPS. The USPS dataset [60] deals with the task of handwritten digit recognition, similar to the MNIST dataset. Its training set contains 7291 samples and its test set 2007. Both sets were acquired using different methods, thus it is regarded more complex than the MNIST dataset. Each scanned digit is deslanted and size-normalized, which results in a $16 \times 16px$ gray-scale image. An input sample $(\mathbf{x}^i, y^i)$ has a dimensionality of $\mathbf{x} \in \mathbb{R}^{256}$ and $y \in \{0, 1, \dots 9\}$. Sample images are shown in Figure 16b. Again, the digits are written in white on a black background, with shades of gray at the borders.
We evaluate two different data representations:

1. We use the raw pixel values as input, which are normalized to $[-1, 1]$.

2. Due to a smaller dataset size we evaluate the usage of descriptors. Therefore we extract the Spatial Pyramid Histogram of Oriented Gradients (SPHOG) descriptor [72] from each sample image, which was proposed by [73] for digit recognition as it archives good performance. We calculate the SPHOG for 3 pyramid levels

with a block size of $\{16, 8, 4\}$. The blocks are 50% overlapping. We use 12 gradient orientations calculated by using the first order derivative of a Gaussian with $\sigma = 1$. The descriptor dimensionality is $12 \cdot (1 + 8 + 40) = 588$.

The evaluation results are shown in Table 11. The reported human accuracy is 97.5% [103] for this dataset. For the raw pixel values the RFo performs best, due to the sophisticated node split optimization. However, our PCA Ferns significantly outperform the original RFe. The data is highly correlated which benefits the PCA embedding. The rest of our classifiers perform similar to the baseline RFe. However, a comparison of these methods using the same number of features shows, that for $S = 9$ our baseline original RFe performs at $88.58 \pm 0.83\%$ on the raw pixels. Again, the CCA and LDA Ferns significantly outperform the baseline for the same number of features. The baseline RFe with $S = 7$ performs at $87.95 \pm 0.25\%$. Thus the Direct Ordinal Ferns slightly outperform the baseline for the same number of features. Nevertheless the Maximum Order Ferns perform inferior in this case. The Maximum Order Ferns perform best with $K = 2$, which makes them similar to the original RFe. However, their feature evaluation uses a weaker criterion that derogates the performance.
Although 66.92% of all pixels of the USPS dataset are non-zero, it contains no zero-columns and is normalized to $[-1, 1]$ which relaxes the problem with the monochrome background. For the normalized data the WTA Ferns with a smaller window size of $K = 4$ performs best. The window size is reduced due to the normalization that mitigates the zero value issue.

We also report the performance on the SPHOG representation for the USPS dataset. The SPHOG representation accumulates gradients orientations over subwindows. The gradients are mostly located on the edges between digit and background and the uniform background contains no gradients. Thus the descriptor is sparsely filled with 77.61% non-zero. However, the gradients can be accumulated for greater areas of the image due to the pyramidal structure. The usage of the SPHOG descriptor benefits all but the generalized RFe. The RFo and the LDA Ferns perform best. Due to the pyramidal structure the features are less correlated, thus less dependent features are modeled, i.e. S is smaller. Further they are better separable in the, though augmented, subspace, which benefits the LDA Ferns, as well as the SVM. For the WTA and Maximum Order Ferns a larger window size is used, due to the increased dimensionality and the sparsity.

In contrast to the machine learning datasets where the parameters strongly depend on the dataset, the parameters for these datasets perform best, that provide the highest number of features as well as most modeled joint probabilities. So for example with $M = 50$ and $S = 15$ we use 750 individual features for classification. This shows that the independence assumption between the individual features does not hold for these datasets. Especially neighboring pixels are dependent, i.e. correlated.

Classifier	Raw pixels, normalized		SPHOG	
	Acc	Parameters	*Acc*	Parameters
RFo [9]	$\mathbf{93.11 \pm 0.09}\%$	$S = 15$	$\mathbf{95.53 \pm 0.14}\%$	$S = 15$
Linear SVM [22]	85%	$C = 2^{-9}$	93.97%	$C = 2^{-13}$
Original RFe [84]	$90.26 \pm 0.17\%$	$S = 15$	$93.02 \pm 0.70\%$	$S = 10$
Generalized RFe [20]	$87.53 \pm 0.46\%$	$S = 15$	$87.22 \pm 0.98\%$	$S = 9$
Random Oblique	$90.08 \pm 0.30\%$	$S = 15$, $T = 32$	$92.60 \pm 0.34\%$	$S = 15$, $T = 19$
PCA	$91.48 \pm 0.35\%\bullet$	$S = 15$, $T = 32$	$93.52 \pm 0.50\%$	$S = 15$, $T = 19$
CCA	$90.05 \pm 0.35\%$	$S = 10$, $T = 16$	$93.32 \pm 0.31\%$	$S = 10$, $T = 19$
LDA	$90.12 \pm 0.51\%$	$S = 9$, $T = 16$	$94.17 \pm 0.28\%\bullet$	$S = 9$, $T = 37$
WTA	$90.61 \pm 0.30\%$	$S = 7$, $K = 4$, $T = 256$	$93.17 \pm 0.24\%$	$S = 4$, $K = 7$, $T = 588$
Maximum Order	$88.99 \pm 0.28\%\circ$	$S = 15$, $K = 2$, $T = 256$	$92.33 \pm 0.31\%$	$S = 15$, $K = 4$, $T = 588$
Direct Ordinal	$88.90 \pm 0.59\%\circ$	$S = 7$, total order	$92.04 \pm 0.35\%\circ$	$S = 6$, total order
	$87.80 \pm 0.73\%\circ$	$S = 7$, weak order	$92.17 \pm 0.75\%$	$S = 5$, weak order

Table 11: Evaluation results for the USPS dataset comparing different embeddings by showing the average classification accuracy with its standard deviation $Acc \pm Acc_\sigma$.

4.1.3 Image classification

Another common task in computer vision is image classification for which a diversity of datasets exist. In order to evaluate our Ferns classifiers we use the probably most common dataset which is the Caltech 101 dataset [25]. It contains over 9000 images from 101 object categories thus $y \in \{1, \dots 101\}$. Sample images are shown in Figure 17. We skip the additional background class and follow the standard protocol for evaluation. Thus we use 15 images per class for training and randomly selected 15 from the remaining images for testing. Each image is represented by Pyramid Histogram Of Visual Words (PHOW) features [8]. This representation uses the descriptors of interest points and assigns them to the nearest visual words. The occurrences of visual words are then spatially counted in histograms, which is done for different spatial extends in a pyramidal manner. The histograms are then combined over the different spatial extends to form a single descriptor. The codebook for visual words assignment is computed using k-means [53] and we use a codebook size of 600. This results in a feature vector $\mathbf{x} \in \mathbb{R}^{12000}$ for each image. However, this is a very simple representation by using only a single image cue, i.e. dense SIFT. For calculating the image representations we use the implementation of [110]. Note, that we do not use any further kernel mappings [59], which are commonly used with this representation, despite the fact that the reported accuracies are higher with the kernel representation. The kernel maps the data into a higher dimensionality, which is hard to handle because of the memory restrictions on our evaluation system.
Due to the high feature dimensionality we further evaluate a PCA compressed dataset. Therefore we project the feature vectors on the 1160 major principal components, which capture 95% of the variance of the original data.

Figure 17: Sample images of the Caltech 101 dataset for randomly chosen categories.

The results for the different representations are shown in Table 12. The reported state-of-the-art accuracy for this descriptor is 64% using a kernel SVM [110]. The histogram representation has a very high dimensionality that is hard to handle. We use only 50 ferns with e.g. 15 features each, thus only a fraction of the 12 000 features are actually used for classification. The best results are obtained by the SVM which fits the maximum margin hyperplanes that can separate the data best. The SVM performs also well on the PCA compressed data, which is in contrast to the MNIST dataset, where the SVM performed worse. The MNIST dataset has more samples that are scattered in the PCA space, which makes it harder to fit the hyperplanes. The well separable data also benefits the LDA Ferns that perform very well on the histogram data.
Also the PCA Ferns outperform the original RFe, because they use a high number of input features for estimating the variances due to the sparse histograms. However, the CCA Ferns perform worse because the histograms are sparse which derogates the CCA method, similar to the zero-column issue described for the digit recognition.
The WTA Ferns perform best among the Ferns for this evaluation, which can be accounted to the histogram representation. The histograms contain counts of visual words and the

number of occurrences of a visual word should be the same for similar images, i.e. the histograms show the same peak. The WTA Ferns use the position of the maximum histogram bins within the subspace as feature and as a high value bin represents a high occurrence of a visual word this provides a distinctive feature. This can be seen on the parameters of the WTA Ferns, where the window size is $K = 10$ and only a single code per fern is used. However, the Maximum Order Ferns cannot benefit from this property, as they do not encode the position of the maximum directly, but only through an inequality constraint. This weakens the leaf assignment criterion such that $K = 2$ provides the best results, which resembles a leaf assignment similar to the original RFe.
The sparse histogram representation does not fit for the Direct Ordinal Ferns either, as they use only a small number of features per fern, which are presumably tied with empty bins. Further the Direct Ordinal Ferns require many training samples, as we show later in an explicit evaluation, which are not present for this evaluation.

For the PCA compressed data LDA and CCA Ferns outperform the baseline by magnitude. The reason is the orthogonal PCA space, as already described for digit recognition. However, in this experiment the PCA compressed dataset is linearly well separable, thus the linear SVM achieves a high classification accuracy as well.

Although the RFo performs a sophisticated feature selection, it performs comparably bad for both data representation. For each feature selection we consider $\sqrt{12000} \approx 110$ features, which seems too little for the sparse histograms. However, it performs similar to the original RFe for the histogram representation, and it clearly takes advantage of the feature selection for the PCA compressed dataset as shown in the results.

Classifier	Histogram		PCA	
	Acc	Parameters	*Acc*	Parameters
RFo [9]	$26.38 \pm 0.72\%$	$S = 15$	$24.26 \pm 1.61\%$	$S = 15$
Linear SVM [22]	$\mathbf{56.17}\%$	$C = 2^{-15}$	$\mathbf{57.49}\%$	$C = 2^{-15}$
Original RFe [84]	$27.34 \pm 1.36\%$	$S = 5$	$10.89 \pm 0.97\%$	$S = 4$
Generalized RFe [20]	$16.34 \pm 1.06\%$	$S = 15$	$5.77 \pm 0.50\%$	$S = 15$
Random Oblique	$21.62 \pm 0.70\%\circ$	$S = 5$, $T = 94$	$11.95 \pm 1.75\%$	$S = 3$, $T = 10$
PCA	$30.05 \pm 1.26\%\bullet$	$S = 11$, $T = 375$	$10.76 \pm 0.79\%$	$S = 5$, $T = 10$
CCA	$19.39 \pm 0.58\%\circ$	$S = 5$, $T = 94$	$32.90 \pm 0.61\%\bullet$	$S = 5$, $T = 145$
LDA	$35.17 \pm 0.82\%\bullet$	$S = 7$, $T = 375$	$50.47 \pm 0.55\%\bullet$	$S = 7$, $T = 580$
WTA	$36.45 \pm 1.24\%\bullet$	$S = 1$, $K = 10$, $T = 12000$	$13.15 \pm 1.64\%\bullet$	$S = 1$, $K = 10$, $T = 1160$
Maximum Order	$27.59 \pm 1.23\%$	$S = 7$, $K = 2$, $T = 12000$	$9.40 \pm 0.80\%\circ$	$S = 7$, $K = 2$, $T = 1160$
Direct Ordinal	$24.84 \pm 0.60\%\circ$	$S = 4$, total order	$7.67 \pm 0.72\%\circ$	$S = 4$, total order
	$25.70 \pm 1.35\%\circ$	$S = 2$, weak order	$7.88 \pm 0.65\%\circ$	$S = 4$, weak order

Table 12: Evaluation results for the Caltech 101 dataset comparing different embeddings by showing the average classification accuracy with its standard deviation $Acc \pm Acc_{\sigma}$.

4.1.4 Face recognition

The last specific computer vision problem that we discuss is face recognition, which is yet another common application of computer vision. The characteristics of face recognition datasets are the small amount of training images and the possibly minor changes between images. We evaluate two different datasets, which are the ORL [93] and the Multi-PIE [37] dataset, both common benchmarks for face recognition methods. Again, we use two different data representations. For the ORL dataset we use the normalized, raw images and for the Multi-PIE we evaluate the PCA compressed image data.

ORL. The ORL face recognition dataset [93] consists of 40 persons each represented by 10 different images showing them in an upright, frontal position. The images show variations in lightning, different facial expressions, and were taken at different times with some persons changing glasses. The background is homogeneous and dark. The initial images are cropped to $64 \times 64px$ such that the face is centered within the image. Each image is then normalized to unit norm. We follow the standard evaluation protocol for this dataset and train the classifier on 5 images of each person and test on the remaining 5 [11]. Therefore we use 10 random splits of the data into training and test set. Sample images are shown in Figure 18a.

(a) ORL samples each $64 \times 64px$ [11].

(b) Multi-PIE samples each $80 \times 80px$ [88].

Figure 18: Sample images of face recognition datasets.

The evaluation results are shown in Table 13. Our LDA Ferns outperform all methods. Also the linear SVM performs well, by using the maximum margin constraint it fits the separating hyperplanes very well due to the little training samples and linear separability. Our LDA Ferns perform even close to state-of-the-art methods on the dataset, which report e.g. 95.50% [109] or 96.30% [125]. Our simple Ferns structure also outperforms the RFo by magnitudes. LDA is well known to provide very good recognition rates for face detection on image data, because the subspace resembles the most discriminative features from the image data. Our LDA Ferns work comparable to random sampling LDA of Wang et al. [117], although this method is a specifically engineered state-of-the-art method for face recognition.

All of the Ordinal Ferns perform similar, yet worse. They suffer from difficulties to extract discriminative features from the image data, as different pixel combinations are prone to noise, different face alignment, facial expressions, and non-Lambertian surface effects, e.g. shadows, and do not provide good features for this task.

Classifier	*Acc*	Parameters
RFo [9]	$80.60 \pm 3.36\%$	$S = 15$
Linear SVM [22]	92.50%	$C = 2^{-7}$
Original RFe [84]	$78.60 \pm 2.16\%$	$S = 6$
Generalized RFe [20]	$75.20 \pm 3.73\%$	$S = 10$
Random Oblique	$77.00 \pm 3.26\%$	$S = 5$, $T = 32$
PCA	$79.40 \pm 1.95\%$	$S = 11$, $T = 128$
CCA	$64.00 \pm 2.37\%\circ$	$S = 7$, $T = 32$
LDA	$\mathbf{93.70 \pm 1.25\%}\bullet$	$S = 5$, $T = 64$
WTA	$75.50 \pm 2.15\%$	$S = 9$, $K = 2$, $T = 4096$
Maximum Order	$73.90 \pm 1.67\%\circ$	$S = 11$, $K = 2$, $T = 4096$
Direct Ordinal	$71.70 \pm 2.20\%\circ$	$S = 5$, total order
	$72.80 \pm 2.41\%\circ$	$S = 5$, weak order

Table 13: Evaluation results for the ORL dataset comparing different embeddings by showing the average classification accuracy with its standard deviation $Acc \pm Acc_\sigma$.

Multi-PIE. Further we evaluate the Multi-PIE dataset [37], which resembles a very large dataset with more than 750 000 images of 337 persons. The images were recorded in different sessions and provide different view points, illumination conditions, as well as facial expressions. From the frontal-most images we detect the faces using a Viola-Jones face detector [115] and normalize the detections to $80 \times 80px$ gray-scale patches with zero mean and unit variance, which are 64 809 faces from 150 persons. As this dataset size is not tractable memory-wise, we use PCA to reduce the dimensionality by projecting the patches onto the 254 major principal components that resemble 95% of the variance [88]. We randomly select 10 000 samples as our test set and use the remaining samples for training. Sample images are shown in Figure 18b. The dataset is more complex, because the face detector has detection jitter and the face orientations also vary more than in the ORL dataset.

The results for the different classifiers are shown in Table 14. The reported performance is low for all classifiers as this dataset is quite difficult. However, our CCA and LDA Ferns outperform all other methods by magnitude. This dataset is PCA compressed and thus the CCA and LDA Ferns perform very well as already discussed for the digit recognition.

Classifier	*Acc*	Parameters
RFo [9]	$31.81 \pm 0.51\%$	$S = 15$
Linear SVM [22]	14.11%	$C = 2^{-15}$
Original RFe [84]	$31.29 \pm 0.99\%$	$S = 8$
Generalized RFe [20]	$14.99 \pm 1.51\%$	$S = 15$
Random Oblique	$31.02 \pm 1.37\%$	$S = 8$, $T = 8$
PCA	$31.18 \pm 0.89\%$	$S = 8$, $T = 8$
CCA	$35.74 \pm 0.64\%\bullet$	$S = 7$, $T = 16$
LDA	$\mathbf{50.42 \pm 0.31\%}\bullet$	$S = 9$, $T = 64$
WTA	$30.88 \pm 0.84\%$	$S = 7$, $K = 2$, $T = 254$
Maximum Order	$26.14 \pm 0.46\%\circ$	$S = 15$, $K = 2$, $T = 254$
Direct Ordinal	$22.21 \pm 2.54\%\circ$	$S = 4$, total order
	$22.00 \pm 2.00\%\circ$	$S = 5$, weak order

Table 14: Evaluation results for the Multi-PIE dataset comparing different embeddings showing the average classification accuracy with its standard deviation $Acc \pm Acc_\sigma$.

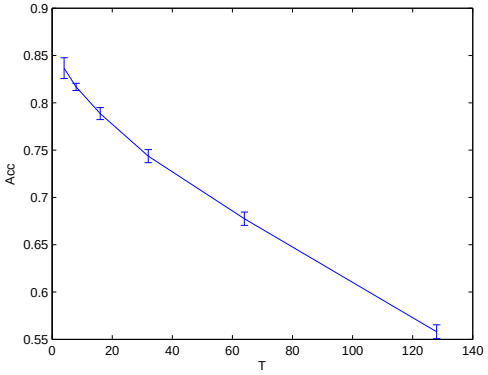
4.1.5 Ferns analysis

We have presented the results for different specific datasets and in the following paragraphs we analyze different aspects of our proposed Ferns in detail. First we show the influence of the different parameters and then we compare the different thresholds for the Oblique Ferns and present evaluation results for the Subspace Ordinal Ferns. Further we compare the total and weak order for Direct Ordinal Ferns and show how our classifiers react to noise perturbed input. Then we discuss the influence of the number of training samples on the classification accuracy and show effects of the probabilistic formulation.

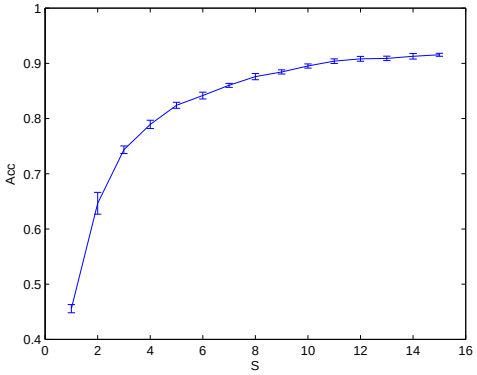
Oblique Ferns parameter analysis. In comparison to the original RFe, our Oblique Ferns require one more parameter, namely the input dimensionality T. We analyze the parameters in the following paragraphs for the Oblique Ferns.
Our Oblique Ferns require parameters for the number of random input features T and the number of features per fern S. We perform two experiments, one by varying T and fixing S and the other one vice versa. We report the average classification accuracy with its standard deviation for each parameter combination acquired from 5 random trails. The parameters are varied within the limits that have already been shown in Table 6.

We first show the effects of the parameter choice for the PCA Ferns, with the evaluations depicted in Figure 19 for the USPS dataset. Some parts of the plot are spared out, because $S > T$ are non-valid parameter combinations.
For a fixed S the accuracy decreases when T gets larger, because more dimensions are used for calculating the subspace projection. Thus feature-specific data properties of the dataset get neglected towards more global data properties. This has similar effects as applying PCA dimensionality reduction on the full dataset. Hence, using the local data properties is advantageous. Further, by using more random input dimensions T the projections change towards a global representation and there is also information in the other neglected components that is missed, because we are still using only the S most major principal components. Thus using most of the T dimensions is better, although

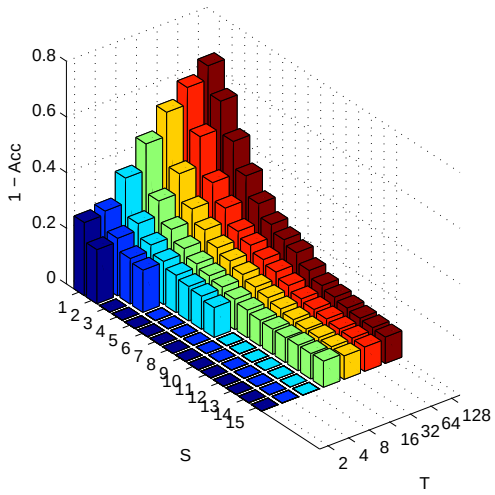
this depends on the data as seen for the evaluations before. This can also be observed on the second plot, where T is fixed and the performance increases with using more features of the subspace. However, the increase gets smaller when S gets closer to T and increases only marginally between $S = 10$ and $S = 15$. The projections are sorted for our PCA Ferns with the major principal components in the front and the lower in the back, thus the prior dimensions contain the higher variances. Only minor variances are used for the latter dimensions, which are less significant and thus only contribute to the result marginally.



(a) $Acc \pm Acc_\sigma$ over T with $S = 3$.



(b) $Acc \pm Acc_\sigma$ over S with $T = 32$.



(c) $Acc \pm Acc_\sigma$ over S and T.

Figure 19: Classification accuracy with its standard deviation $Acc \pm Acc_\sigma$ for PCA Ferns using different parameters. Shown for the USPS dataset.

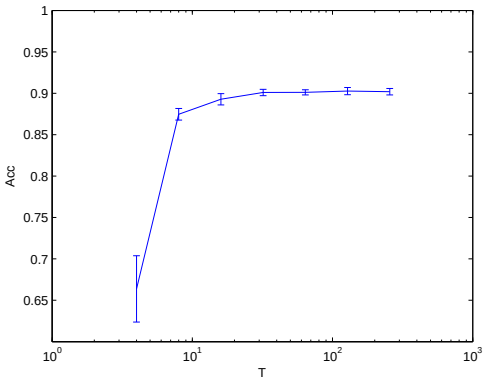
The same evaluations can be conducted for the Random Oblique, LDA, and CCA Ferns. However, the results are similar and for the sake of compactness we spare them out. Nevertheless the input dimensionality T depends on the data structure, where some datasets might need more and others less input dimensions for an optimal result.
The subspace is limited for the LDA and CCA Ferns by the number of classes, i.e. $S \leq H - 1 = 9$ and $S \leq H = 10$, for the USPS dataset. But similar to the PCA

subspace, the performance increases by concerning more subspace dimensions per fern. The Random Oblique Ferns provide the best results if $T \approx S$ by simply using different projections per fern. Using the random oblique subspace for dimensionality reduction does not make sense, unless there are dimensions that contain no information and thus can be suppressed, as e.g. the monotone background in the MNIST dataset.

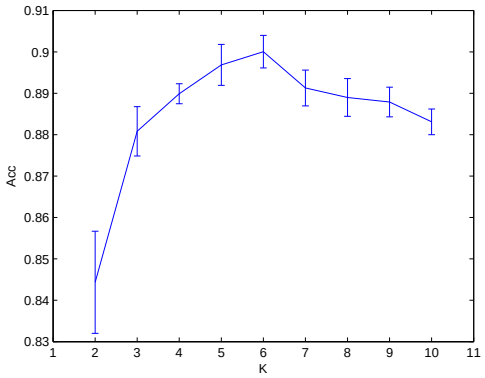
Ordinal Ferns parameter analysis. Further we analyze the Ordinal Ferns in the same manner.
The Direct Ordinal Ferns have a straight forward parametrization similar to original RFe, as they need only one parameter, which is the number of features per fern S and the type of order, i.e. total or weak. These properties are shown in different experiments in detail in the next section.
However, the WTA and Maximum Order Ferns require further parameters for the window size K, the input dimensionality T, and the number of features or codes per fern S, which we evaluate next.
The influence of the parameters for the WTA Ferns are shown in Figure 20 for the USPS dataset and in Figure 21 for the Ionosphere dataset. We use two different datasets in order to show the different behavior of the window size depending on the complexity of the dataset. Again, impossible configurations are spared out, as e.g. the required memory for $K^S > 32768$ is too large.
For both datasets a larger subspace T of the feature selection benefits the classification accuracy. A larger subspace for feature selection leads to more possible window combinations $\binom{T}{K}$ and a smaller correlation between the individual hash codes. The increase is significant for the first few dimensions, but flattens when a certain number of dimensions is reached.
A larger window size K increases the number of inequalities per feature and thus increases the strength of each fern. However, this is not advantageous for all datasets. While for the USPS the best result is achieved with a window size of $K = 6$, the Ionosphere dataset requires a smaller window size of $K = 2$ for good results. With a larger K the ferns tend to overfit and more samples would be necessary to provide reasonable leaf probabilities.
Similar properties can be observed for the number of codes per fern. While more codes per fern provide better results for the USPS dataset with $S = 7$, the optimal parameter for the Ionosphere dataset is $S = 5$. The USPS dataset resembles highly correlated data thus modeling more dependencies is beneficiary, which is in contrast to the Ionosphere dataset, because it contains a less correlated data structure.

Similar evaluations can be conducted for the Maximum Order Ferns. Again, more input dimensions T provide less correlated features per fern, which increases the classification performance. This is also unison with the original RFe, which potentially use the full input space for feature selection. Again, more features per fern S provide better results. However, the performance increases with more features per fern further, while for the WTA Ferns the curve already flattens for $S = 7$. The Maximum Order Ferns are more prone to overfitting due to the randomization in the feature generation process, which we show in the following section for the evaluation of the probabilistic formulation in detail.

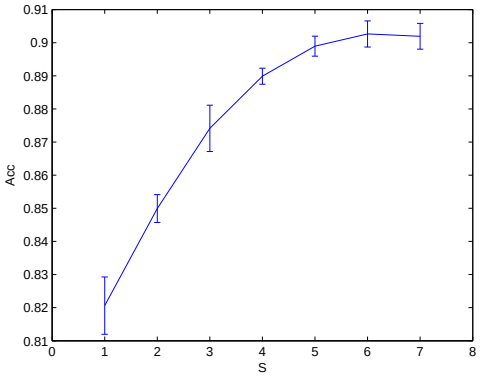
At last we discuss our Soft Assignment Ferns, which require two parameters λ and θ for calculating the feature weights. Therefore we evaluate the Soft Assignment Ferns



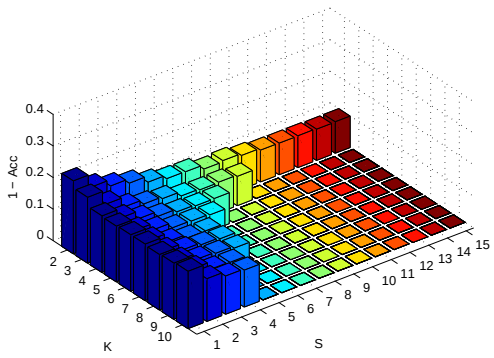
(a) $Acc \pm Acc_\sigma$ over T with $K = 4$ and $S = 7$.



(b) $Acc \pm Acc_\sigma$ over K with $T = 256$ and $S = 4$.

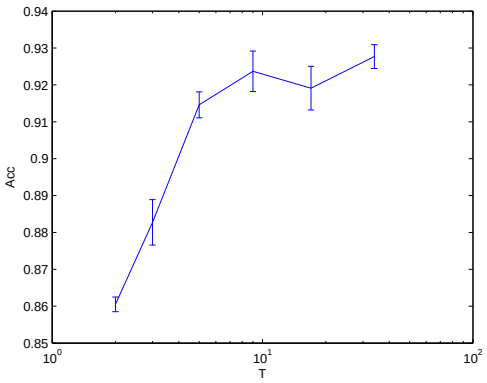


(c) $Acc \pm Acc_\sigma$ over S with $T = 256$ and $K = 4$.

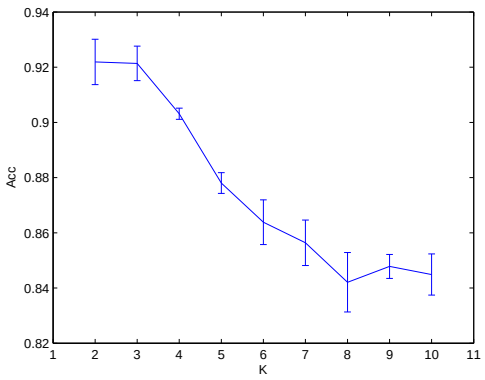


(d) $Acc \pm Acc_\sigma$ over S and K with $T = 256$.

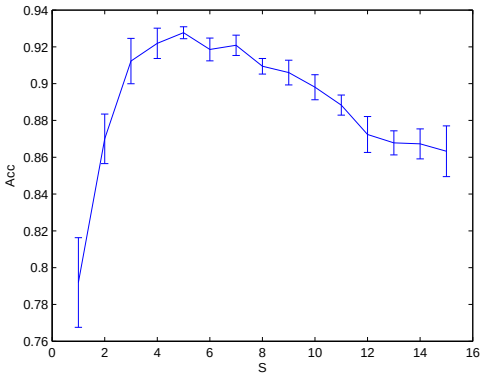
Figure 20: Classification accuracy with its standard deviation $Acc \pm Acc_\sigma$ for WTA Ferns using different parameters. Shown for the USPS dataset.



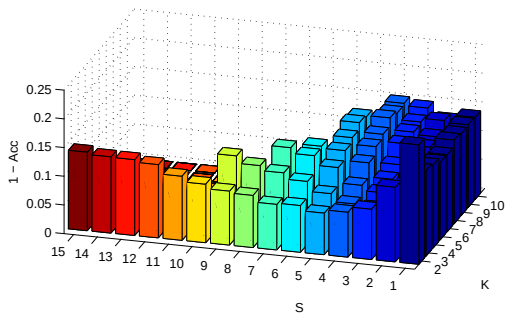
(a) $Acc \pm Acc_\sigma$ over T with $K = 2$ and $S = 5$.



(b) $Acc \pm Acc_\sigma$ over K with $T = 17$ and $S = 4$.



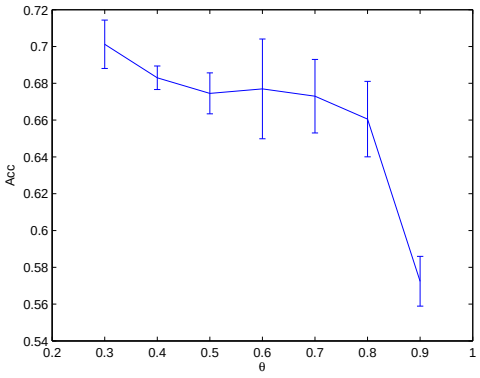
(c) $Acc \pm Acc_\sigma$ over S with $T = 17$ and $K = 2$.



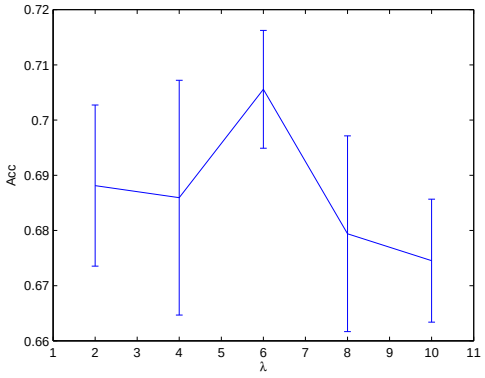
(d) $Acc \pm Acc_\sigma$ over S and K with $T = 17$.

Figure 21: Classification accuracy with its standard deviation $Acc \pm Acc_\sigma$ for WTA Ferns using different parameters. Shown for the Ionosphere dataset.

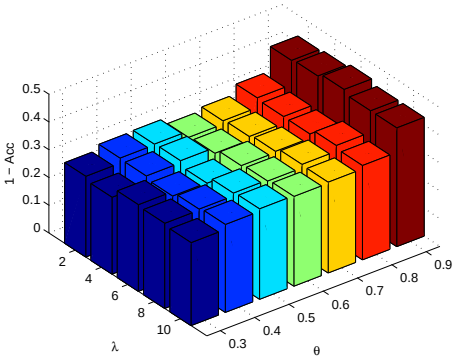
for the Isolet dataset, which shows significantly different results compared to the Direct Ordinal Ferns.
If no additional noise is added, the influence of the parameters on the classification result is negligible, as shown in Figure 22d. Thus we add Gaussian noise to the feature values with zero mean and a standard deviation of 40% of the feature value. The threshold θ models the number of features, i.e. leaves, that are concerned for soft assignment. A lower threshold leads to more leaves that are considered and vice versa. If noise is added, weighting more leaves provides better accuracy. For $\theta = 0.9$, which almost resembles the case of no soft assignment, the accuracy is worst due to the fact that the highly perturbed input features lead to noisy leaf assignments. However, the exponential factor λ does not effect the result much and might be neglected.



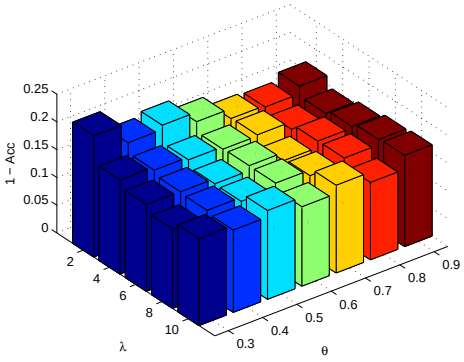
(a) $Acc \pm Acc_\sigma$ over θ with $S = 5$, $\lambda = 10$ and 40% noise.



(b) $Acc \pm Acc_\sigma$ over λ with $S = 5$, $\theta = 0.5$ and 40% noise.



(c) $Acc \pm Acc_\sigma$ over λ and θ with $S = 5$ and 40% noise.



(d) $Acc \pm Acc_\sigma$ over λ and θ with $S = 5$ without noise.

Figure 22: Classification accuracy with its standard deviation $Acc \pm Acc_\sigma$ for Soft Assignment Ferns using different parameters. Shown for the Isolet dataset.

Subspace Ordinal Ferns. Next we evaluate the Subspace Ordinal Ferns. As their results strongly depend on the used subspace method we do not evaluate them on all datasets, but on some selected.

The evaluation results are shown in Table 15. We use the Caltech dataset that provides a high number of classes which does not penalize the LDA subspace, further the ORL dataset that represents raw image data with a reasonable number of classes, the Isolet dataset with a reasonable number of classes but for non-pixel data, and the USPS dataset with pixel representation and a small number of classes thus causing a small LDA subspace.
We evaluate 5 random trails and report average accuracy with its standard deviation. We report the accuracy for the optimal parameters which are chosen from a cross-validation among the parameter space. Both PCA versions are trained with the same randomly chosen input dimensions.
The magnitude of each subspace dimension can change due to the subspace projection, which influences the ordinal analysis. As we already discussed for the ordinal embeddings, the magnitude of the feature dimensions must be similar for ordinal analysis, i.e. derived from one feature scale. This requirement might be violated by the subspace projection, and therefore in this case a feature normalization is required. However, an evaluation has shown that there is no significant difference in the accuracy and by neglecting the normalization we can reduce the runtime.
For the Caltech dataset the accuracy of the PCA Ferns decrease from 30.05% to 25.93% and 27.60% compared to the threshold leaf assignment. The LDA Ferns decrease from 35.17% to 30.65%. However, for the ORL dataset the accuracy of the PCA Ferns increase from 79.40% to 82.00% and 83.60%, though the LDA Ferns' performance decreases significantly from 93.70% to 86.30%. For the Isolet dataset all evaluated Subspace Ordinal Ferns increase their performance. For the PCA Ferns the performance rises from 86.35% to 88.66% and 88.79%, and for the LDA Ferns from 91.49% to 93.75%. On the USPS dataset the Ferns do not show any statistically significant difference in the accuracy compared to the threshold leaf assignment.

Dataset	PCA + original	PCA + WTA	LDA + WTA
Caltech 101 histograms	$25.93 \pm 1.92\%\circ$ $S = 5, T = 94$	$27.60 \pm 1.16\%\circ$ $S = 3, T = 94$	$30.65 \pm 1.14\%\circ$ $S = 3, T = 375$
ORL raw pixels	$82.00 \pm 1.50\%\bullet$ $S = 5, T = 64$	$83.60 \pm 3.38\%\bullet$ $S = 3, T = 64$	$86.30 \pm 0.76\%\circ$ $S = 3, T = 32$
Isolet	$87.57 \pm 0.28\%\bullet$ $S = 7, T = 39$	$88.17 \pm 0.82\%\bullet$ $S = 3, T = 39$	$93.75 \pm 0.20\%\bullet$ $S = 5, T = 309$
USPS raw pixels	$91.12 \pm 0.43\%$ $S = 11, T = 64$	$90.72 \pm 0.37\%$ $S = 7, T = 64$	$89.55 \pm 0.27\%$ $S = 7, T = 32$

Table 15: Comparison of different Subspace Ordinal Ferns showing the average classification accuracy with its standard deviation $Acc \pm Acc_\sigma$. $\bullet$ denotes superior accuracy compared to threshold leaf assignment and $\circ$ inferior accuracy.

In order to understand these partially contradictory results we have to take a closer look on the leaf assignment for the case of ordinal and threshold leaf assignment. Therefore we visualize the decision boundary of a single assignment bit, which is shown in Figure 23. We compare the threshold split with the simple ordinal split of the original feature evaluation method due to simplicity. The WTA decision boundaries are not easily

visualizable, but for a window size of $K = 2$ they are the same as for the original feature evaluation method. While the threshold provides an axially parallel split in the subspace, the ordinal split is the bisecting line between the x_1- and x_2-axis for $x_1, x_2 \in \mathbb{R}^+$.

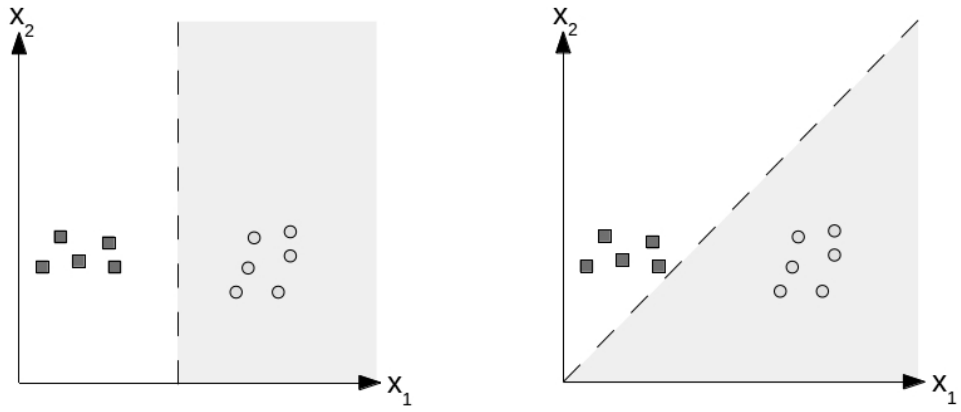
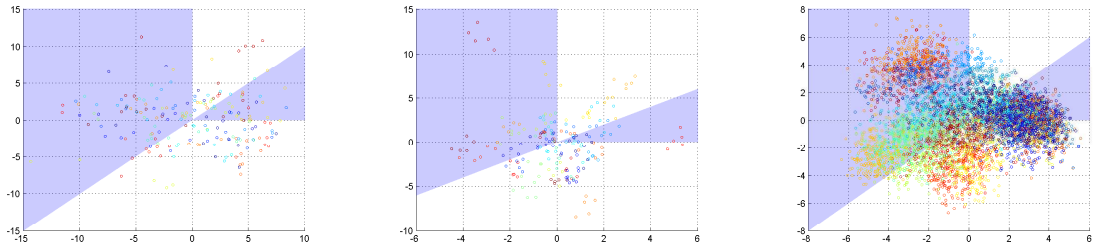


Figure 23: Topology of leaf assignment for threshold (left) and ordinal (right), i.e. original feature evaluation method, in a two dimensional subspace.

Obviously the splits are different in the subspace. While the threshold uses axially parallel decision boundaries, they are oblique at 45° angles in the ordinal case. However, the separability of the data through leaf assignment strongly depends on the structure of the data in the subspace. Therefore we plot the subspace for the two different subspace methods for the ORL dataset, which shows an improvement for the PCA and degradation for the LDA subspace, and further for the Isolet dataset, which shows more samples and also an improvement for the PCA subspace. The scatter plots of the data are shown in Figure 24 with the corresponding decision boundaries for $x_1 < x_2$ as split criterion. We project the data on the two major subspace components. However, the data is zero centered, which creates rather strange decision boundary for the negative feature values. The LDA subspace has an elongated shape for the ORL dataset, while the PCA subspace is more spherical as well as for the Isolet dataset. Considering the leaf assignment for the LDA subspace, most of the data is located in one leaf, which reduces the commonality within the leaves and leads to worse classification accuracy. However, the PCA subspace for the ORL dataset separates the data into two balanced sets, though with reduced commonality because the data is obviously linearly non-separable. For the Isolet dataset the PCA subspace with ordinal assignment works well, because the data is spherical and thus the data separation is balanced. Further the form of the data benefits the oblique splits, as they increase the commonality of the data in the leafs compared to axially parallel splits.
The subspace methods create a new orthogonal subspace where e.g. the variance or the inter-class distance is maximized along each axis. This benefits the threshold assignment, which creates the splits orthogonal to these axis along which the methods property is maximized. By using the oblique decision boundaries in the subspace, i.e. ordinal leaf assignment, these maximization properties are ignored which leads to suboptimal leaf assignment. However, one cannot control the subspace properties, as they depend

on the original data structure and the projection method. Thus for some datasets the classification accuracy is improved with the ordinal leaf assignment.



(a) ORL with PCA subspace $T = 128$.

(b) ORL with LDA subspace $T = 64$.

(c) Isolet with PCA subspace $T = 39$.

Figure 24: Scatter plots of the data in the subspace with decision boundaries.

Oblique Ferns threshold selection. In Section 3.2.1 we proposed two different threshold selection methods for the Oblique Ferns. The first and computationally fast method uses the median to bisect the data for each assignment threshold. Our second and more sophisticated method optimizes the threshold on the true positive and true negative rates on the training data. Thus we want to evaluate the effects of the threshold on the classification accuracy. We compare the median threshold to the optimal threshold on several datasets. We use the same random input dimensions for both methods in order to mitigate random training effects. The optimal parameters, i.e. the input dimensionality T and the number of features per fern S, are determined from a cross-validation over the parameter space using the median threshold. The optimal parameters are then used for both methods. For the optimal threshold we sample $t = 100$ thresholds equidistantly between the maximum and minimum value of the individual dimension, from which our proposed algorithm selects the best. We perform 5 trails and report the average classification accuracy with its standard deviation.

The results for some selected datasets are shown in Table 16. For the Australian and USPS dataset there is no statistically significant difference between the two methods. Nevertheless there are significant improvements with the optimal threshold method for the Ionosphere and Isolet dataset.
Our Subspace Ordinal Ferns also use a subspace projection but with a different leaf assignment in the subspace that uses non-axially parallel decision boundaries as we described before. However, the results show a similar behaviour for the datasets. For the Isolet dataset the Subspace Ordinal Ferns with LDA improved the accuracy to 93.75% and with PCA to 88.17%, which is even slightly better than our optimal threshold. Though the performance of the Subspace Ordinal Ferns did not increase for the USPS dataset where PCA scored 91.12% and LDA 89.55%, which is slightly worse than the optimal threshold. However, the optimal threshold is computationally more expensive and does not necessarily provide better results. The performance depends on the separability of the data in the subspace as we show next.

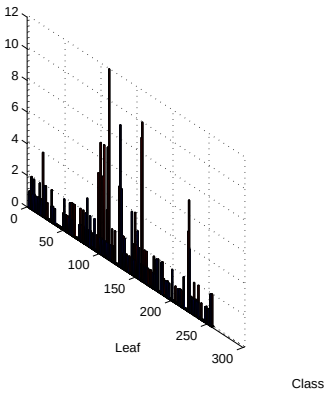
The separability of the data in the subspace can be observed from the scatter plots and the leaf distribution. Figure 25 shows the leaf distribution for the median and for

Dataset	PCA Ferns		LDA Ferns	
	Median	Optimal	Median	Optimal
Australian	$85.04 \pm 0.55\%$	$85.10 \pm 0.47\%$	$86.93 \pm 0.45\%$	$86.72 \pm 0.47\%$
Ionosphere	$86.96 \pm 0.58\%$	$\mathbf{91.69 \pm 0.37}\%$	$79.31 \pm 0.94\%$	$\mathbf{87.93 \pm 0.51}\%$
Isolet	$86.22 \pm 1.01\%$	$\mathbf{87.26 \pm 0.88}\%$	$90.70 \pm 0.60\%$	$\mathbf{92.25 \pm 0.87}\%$
USPS	$91.35 \pm 0.24\%$	$91.40 \pm 0.44\%$	$90.28 \pm 0.33\%$	$90.15 \pm 0.52\%$

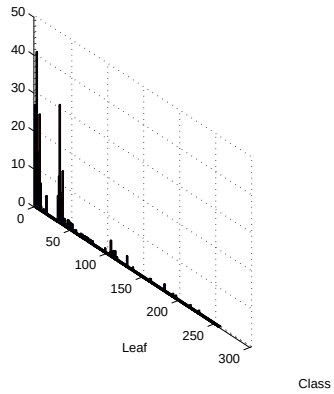
Table 16: Comparison of different threshold methods for Oblique Ferns showing the average classification accuracy with standard deviation $Acc \pm Acc_\sigma$. Statistically significant improvements are marked in boldface.

the optimal threshold as well as the scattered data in the subspace for the Ionosphere dataset. Therefore we use the PCA subspace with $T = 9$ for this illustration and show the first two dimensions of the subspace, i.e. the major principal components of the subspace. However, the same principles apply to the other subspace methods, which are not shown here explicitly. The Ionosphere dataset contains two classes that can be well separated in the subspace by the optimal threshold, as shown in the scatter plot where the two different thresholds are compared. This increases the commonality of the data within the leaves compared to the median threshold, which is indicated by the higher peaks in the histogram. A higher leaf commonality is beneficiary for an increased accuracy. Further the optimal threshold algorithm calculates the threshold for each assignment bit by using the one-vs-all approach as described before. However, the Ionosphere dataset contains only two classes which resembles an one-vs-one approach. This benefits the optimal threshold algorithm, because the separation problem is easier for two classes, thus achieving better results.

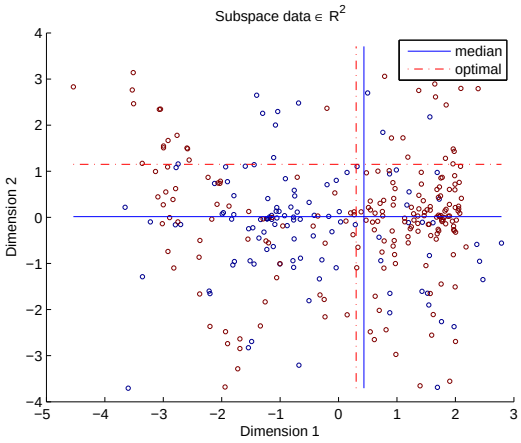
The same plots are shown in Figure 26 for the USPS dataset. We use a LDA subspace with $T = 16$ and show the first two dimensions of the subspace. The multi-class data cannot be well separated as shown in the scatter plots. However, the optimal method finds the optimal global solution, which in this case separates the best separable class from the rest. The best separable class is scattered in the top blue cluster, which is class label 2 that corresponds to the digit 1. This increases the commonality of the data in the leaves for one class, but the rest is slightly worse separable and worse balanced, as e.g. the leaf that corresponds to the top right decision area contains very little samples. The median threshold simply chooses the median that bisects the data, which provides a similar accuracy in this case by enforcing more balanced leaf distributions. Although the optimal threshold increases the classification accuracy for class 2 by 2%, the accuracies of the other classes decrease. Further, the distribution of the leaves shows, that there is hardly any change between the two methods, thus the similar accuracies.



(a) Leaves with median threshold.

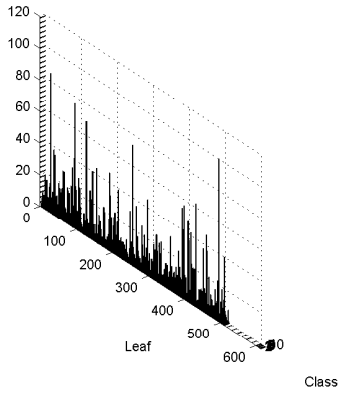


(b) Leaves with optimal threshold.

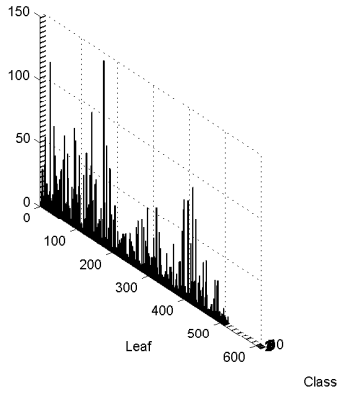


(c) Data in subspace.

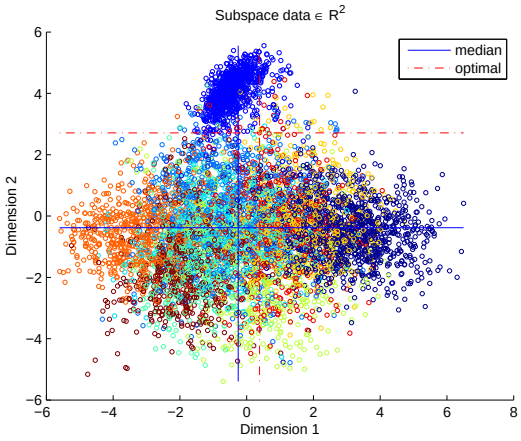
Figure 25: Leaf distributions and scatter plots with thresholds. Shown for the Ionosphere dataset.



(a) Leaves with median threshold.



(b) Leaves with optimal threshold.



(c) Data in subspace.

Figure 26: Leaf distributions and scatter plots with thresholds. Shown for the USPS dataset.

Direct Ordinal Ferns order. Our Direct Ordinal Ferns support two different order methods that handle ties within the input data differently. Therefore we evaluate the two different orders, i.e. weak and total order, and show the effects for different datasets. In the previous experiments we used randomly chosen input features for all methods. In this evaluation we choose random features, but we use the same features for both methods in order to eliminate any influence from the random input feature selection. We perform 5 trails and report average classification accuracy with its standard deviation.

The results are shown in Table 17. The difference between the classification accuracy of the different orders is statistically not significant for all datasets. The datasets Ionosphere and Isolet cause hardly any ties in the input data, but the MNIST dataset has a lot of background pixels, i.e. 0, that cause ties. However, the Australian dataset, which contains several categorical input variables that cause ties, does not benefit from the weak order either.
The minor differences in the accuracies are caused by the increased number of possible permutations for weak order which imposes more leaves. If there are no ties, the leaves that correspond to the tied permutations are empty. Thus the normalization of the frequency matrix with the Dirichlet prior increases its influence. The samples are more spread and the density in the frequency matrix is lower. This causes a difference in the classification accuracy, but with the Dirichlet prior $u \to 0$ the difference decreases towards zero.
The evaluation of the MNIST dataset, which contains ties, does not show this behavior. With a Dirichlet prior $u = 0.01$ the weak order performs at $87.63 \pm 0.50\%$ and the total order at $87.23 \pm 0.47\%$. However, the statistical significance is minor. If there are lots of equal feature values, weak order performs better because it explicitly encodes this equality. If there are not many equal features there is no benefit of a different order. Thus the total order achieves the same accuracy but with lower memory requirements.

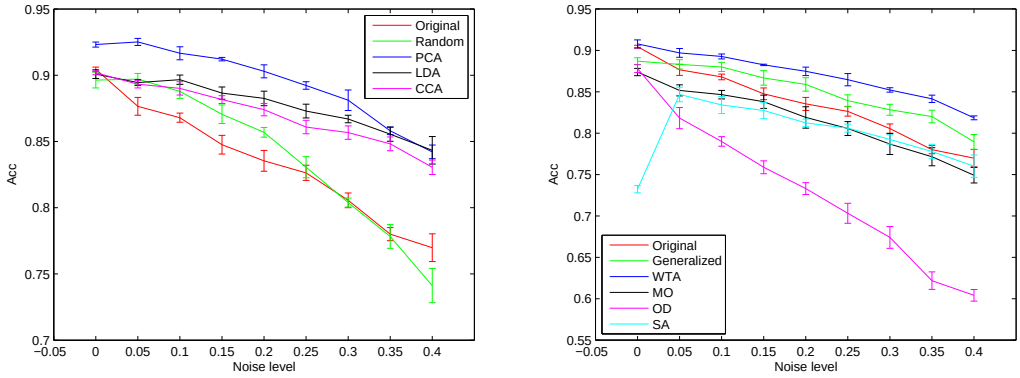
Dataset	Weak order	Total order	Parameters
Australian	$86.29 \pm 0.42\%$	$86.00 \pm 0.51\%$	$S = 5$
Ionosphere	$90.20 \pm 1.08\%$	$89.81 \pm 0.56\%$	$S = 5$
Isolet	$84.61 \pm 1.32\%$	$84.54 \pm 1.19\%$	$S = 5$
MNIST	$85.04 \pm 1.15\%$	$84.81 \pm 1.30\%$	$S = 7$

Table 17: Comparison of different order methods for Direct Ordinal Ferns showing the average classification accuracy with its standard deviation $Acc \pm Acc_\sigma$.

Noise. Further we show the effects of noise on the classification accuracy. Noise is an important factor in classification, as it perturbs the samples and thus causes potentially wrong class assignments or wrong learned models. In order to analyze the effects of noise on our Ferns classifiers, we use simple Gaussian feature noise. Therefore we add Gaussian noise with zero mean and variable standard deviation to the data. We perturb the training and test samples independently, where the standard deviation of the noise depends on the data range of the individual feature dimension. The standard deviation is varied in the range of $[0 \ldots 40\%]$ of the maximum value of the dimension. We perform 5 trails and report average classification accuracy with its standard deviation.

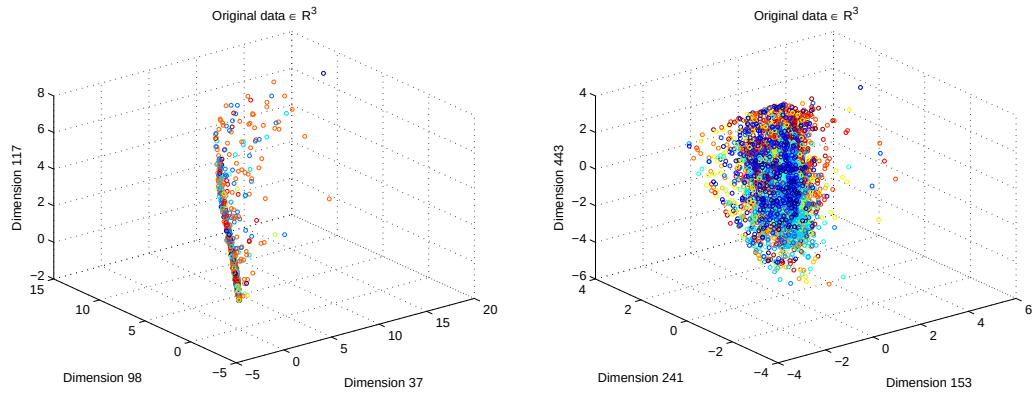
The influence of the noise perturbation is shown in Figure 27 for the USPS dataset. The evaluation shows, that each Ferns classifier reacts differently on the noise perturbation of the data. The USPS dataset contains image data which is known to be exposed to noise.
From the Oblique Ferns the Random Oblique Ferns show the worst performance, because the noise has a direct impact on the subspace projection. For PCA Ferns a small noise perturbation does not change the subspace at all, as the principal components that resemble the variance of the input stay the same. This leads to a resistance for a small noise perturbation, but if the noise level increases, the noise itself makes up a major part of the variances which deteriorates the result. The CCA and LDA Ferns show a similar behavior. They start at lower accuracy, but loose less accuracy than the PCA Ferns. CCA maximizes the correlations between the noisy input data and the class labels thus the Gaussian noise evens out. A similar behavior shows the LDA by increasing the inter-class distance and minimizing intra-class distances it can cope with noise well, by increasing inter-class distance. These observations are in accordance to [5].
However, all Oblique Ferns except the Random Oblique Ferns are more resistant to noise compared to the original RFe and Ordinal Ferns. The original RFe is shown in both plots for comparison.
The Direct Ordinal Ferns have inferior tolerance to noise, as already shown in the simulation results in Section 3.3.4. A single change in the input order causes a different leaf assignment and thus different class probabilities. Our Soft Assignment Ferns mitigate these noise problems, as already theoretically described. For high noise levels they perform similar to all other Ordinal Ferns. However, in the absence of noise they score worse results, because they smooth the conditional probability among the leaves. Further they have a high computational complexity, which makes them inferior to the other Ordinal Ferns.
All other Ordinal Ferns perform similar with a decreasing accuracy for higher noise levels. However, the WTA Ferns perform better than the original RFe and they can keep the advance in accuracy over the other classifiers, because the maximum value is more tolerant to noise. Also the generalized feature evaluation outperforms the original for high noise levels. Due to the threshold the assignment bit stays the same if the value including noise is smaller or larger than the threshold, which is especially given if the value differs significantly from the threshold.

The evaluation on a second dataset, the Isolet dataset, is shown in Figure 29. The dataset resembles numeric features that are non-pixel values and it is considered a noisy dataset by default [24]. All of the Oblique Ferns show similar characteristics as for the USPS dataset. However, the PCA Ferns perform significantly worse for this dataset. A reason for this result can be found in the data structure. The scatter plot of the two datasets in Figure 28 shows, that the Isolet dataset has a more spherical shape and the USPS an elongate shape with highly correlated dimensions. Thus the major principal component of the PCA is located along this major variance which is, due to the scale, rather robust to noise. The Isolet dataset, however, is more prone to noise as projections of the data on the principal components of the PCA exceed the leaf assignment threshold easier due to the greater distance of the features to the principal components, and thus deteriorate the result.



(a) Oblique Ferns. (b) Ordinal Ferns.

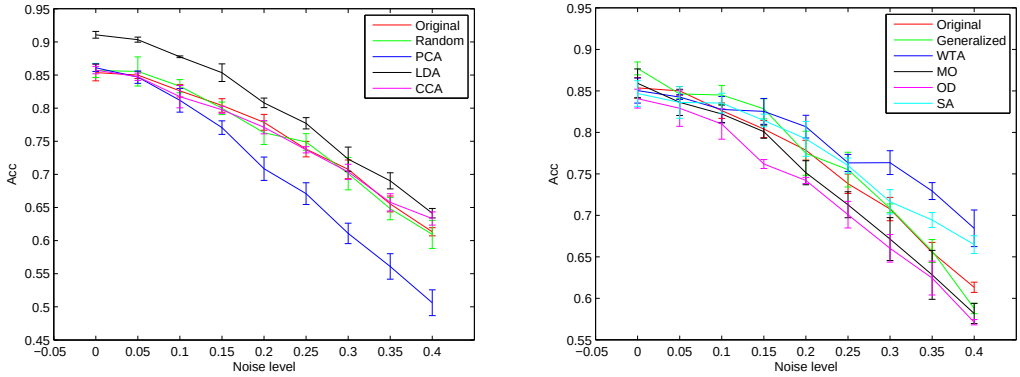
Figure 27: Classification accuracy with its standard deviation $Acc \pm Acc_{\sigma}$ over the percentage of noise added. Shown for the USPS dataset.



(a) USPS. (b) Isolet.

Figure 28: Scatter plots for USPS and Isolet datasets.

The Direct Ordinal Ferns perform worst of all Ordinal Ferns. However, as noise is present in all samples the Ordinal Direct Ferns are not inferior if adding even more noise, because the less correlated data structure tolerates this. The Soft Assignment Ferns perform similar to the other Ordinal Ferns, even outperforming most of the other Ordinal Ferns for high noise levels. Thus the soft assignment is still beneficiary for high noise levels.



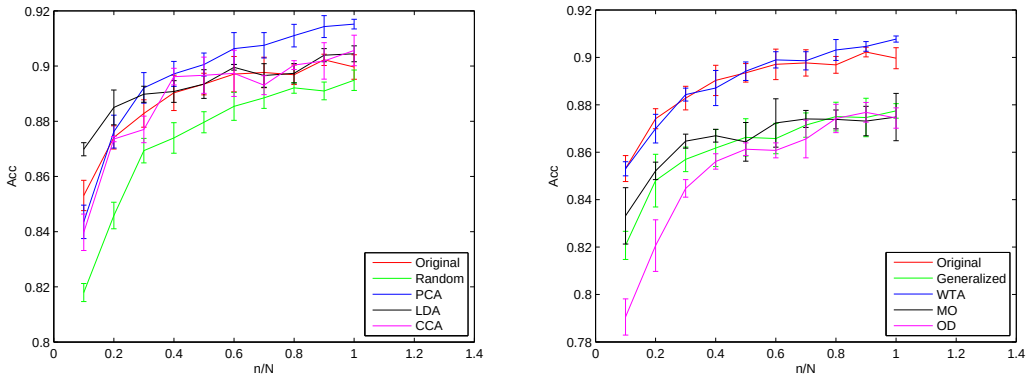
(a) Oblique Ferns.

(b) Ordinal Ferns.

Figure 29: Classification accuracy with its standard deviation $Acc \pm Acc_\sigma$ over the percentage of noise added. Shown for the Isolet dataset.

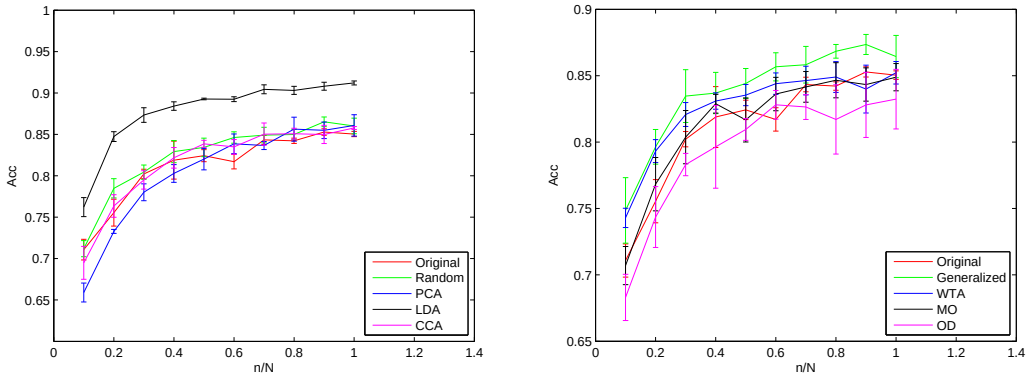
Training samples. Further we analyze how the number of training samples influences the classification accuracy, as the different Ferns classifiers directly depend on the estimation of the probabilities from the frequency matrix. Therefore we use n out of the total N training samples with $n \in \{0.1 \cdot N, \dots N\}$ for training and discard the rest. However, we do not change the overall class distribution while subsampling the training samples. The test samples are not altered and the full test sets are used for testing. For training we use the optimal parameters that are derived from the cross-validation within the parameter space using all samples.
The influence on the number of training samples is shown in Figure 30 and 31 for the USPS and the Isolet dataset, respectively. The tendency is the same for all types of RFe, as they increase their accuracy with the number of training samples.

The more training samples are provided, the better the approximation of the probabilities through the frequency matrix. Thus by using more samples, a better classification accuracy can be achieved.
For both datasets our LDA Ferns show the best results for a small number of training samples. Maximizing the inter-class distances, as LDA does, increases the margin for new samples. The PCA Ferns show the biggest difference between few and many training samples. For a few training samples only small or noisy variances exist in the data, thus the principal components do not resemble the true variance of the full dataset which derogates the accuracy.



(a) Oblique Ferns. (b) Ordinal Ferns.

Figure 30: Classification accuracy with its standard deviation $Acc \pm Acc_\sigma$ over the fraction of training samples used $\frac{n}{N}$. Shown for the USPS dataset.



(a) Oblique Ferns. (b) Ordinal Ferns.

Figure 31: Classification accuracy with its standard deviation $Acc \pm Acc_\sigma$ over the fraction of training samples used $\frac{n}{N}$. Shown for the Isolet dataset.

The Ordinal Ferns perform all very similar except the Direct Ordinal Ferns, which perform worst for a small number of training samples. Due to their strong base classifiers, they tend to overfit the small amount of training samples.
Especially if we use more features per fern, thus increase the number of leaves, the Ferns start to overfit the training samples. For example S features per fern and N samples make $\frac{N}{2^S}$ samples in average per leaf. If there is a small number of training samples, many leaves might be empty. This increases the influence of the Dirichlet prior, which is supposed to mitigate these empty bins in the frequency matrix. The final classification result of the full ensemble might be performed by only a small number of ferns, that actually learned a relevant and discriminative feature combination. This circumstance makes the Ferns classifiers prone to wrong classifications. However, this issue can be mitigated by providing more training samples, by using more ferns thus evaluating more feature combinations, or by using less features per fern. Though using less features per fern can deteriorate the results, because this is a data depending parameter. Increasing the number of ferns in the ensemble is also limited, as it exhibits saturation effects which we show in the next section.
The fundamental problem of providing enough training data to efficiently estimate the conditional probabilities is solved in the original RFe application by generating a large amount of artificial training data [84]. However, this is done for image data and it is significantly more difficult to generate representative training examples for machine learning datasets.

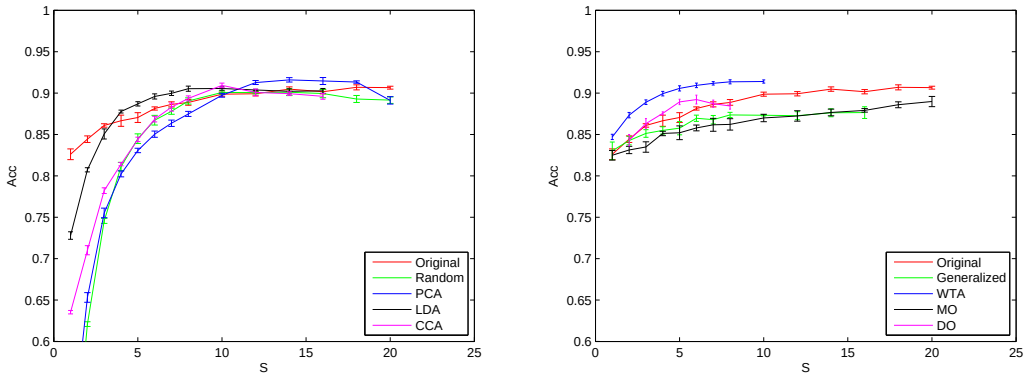
Effects of probabilistic formulation. The core element of the Ferns classifiers is their Semi-Naive Bayesian feature combination. The joint probabilities are modeled by combining the features within the ferns, depending on the degree of dependence between the features. This motivates some evaluations that deal with this probabilistic formulation. Therefore we first evaluate the Semi-Naive Bayesian feature combination and then how the ferns perform with different ensemble sizes, i.e. more features in total.

Complete independence between the features is a very extreme formulation that we relaxed by modeling the joint probabilities in groups of $S = \frac{C}{M}$ features. Thus using the joint probabilities from Equation 5 of a fern $\mathbf{F}^m$ the conditional probabilities are

$$p(f_1, f_2, \ldots, f_C|Y = y^i) = \prod_{c=1}^{C} p(f_c|Y = y^i) = \prod_{m=1}^{M} p(\mathbf{F}^m|Y = y^i). \tag{55}$$

The degree of modeling the joint probabilities can be varied by using a different number of features per fern S. This property depends on the complexity of the data as we show next. Therefore we keep the overall number of features $C = 1000$ constant and model the trade-off between the number of ferns and the features per fern thus $M = \lfloor \frac{C}{S} \rfloor$ for $S \in \{1, \ldots 20\}$. For each S we run 5 trails and report the average classification accuracy. The best working fern-specific parameters, as e.g. the subspace dimensionality T, are selected from a cross-validation among all possible parameters of Table 6.
We show the results for this evaluation in Figure 32 for the USPS dataset. Due to restrictions for some Ferns, e.g. memory or $T \geq S$, not all curves reach $S = 20$ features. Oblique Ferns only perform well for a higher number of features per fern S, thus simply using the major component of each projection is not sufficient for good results. However, the choice

of S depends on the subspace dimensionality T. The major projection vectors, e.g. the principal components, provide the highest increase in performance. In the USPS dataset the pixel values are highly correlated, thus modeling more dependent input dimensions leads to better results.



(a) Oblique Ferns.

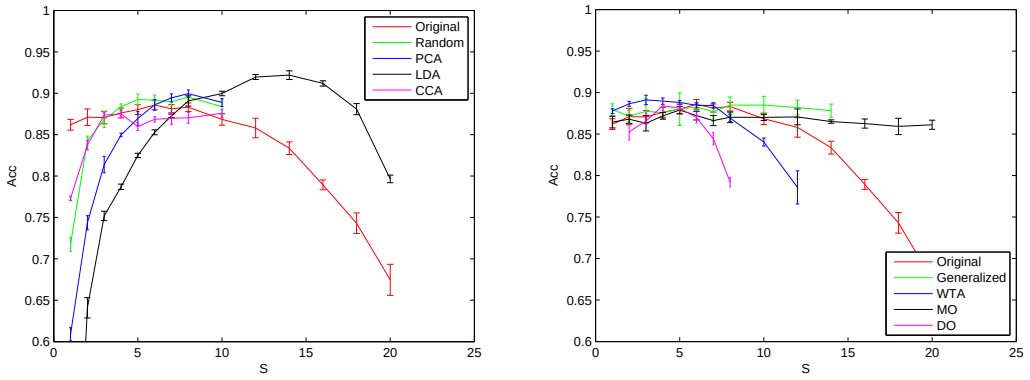
(b) Ordinal Ferns.

Figure 32: Classification accuracy with its standard deviation $Acc \pm Acc_\sigma$ for the same number of features overall but different number of features per fern. Shown for the USPS dataset.

The same evaluation is provided for the Isolet dataset in Figure 33, which has similar properties as the USPS dataset with respect to the dimensionality, dataset size, and number of classes. Some embeddings provide worse performance with higher number of features per fern. The Isolet dataset contains less correlation between the features than the USPS dataset, which leads to overfitting for a higher number of features per fern S. However, the generalized feature evaluation and the Maximum Order Ferns show a fairly constant accuracy even for higher values of S, which indicates their tolerance to overfitting. By using randomized features, they do not overfit the data.

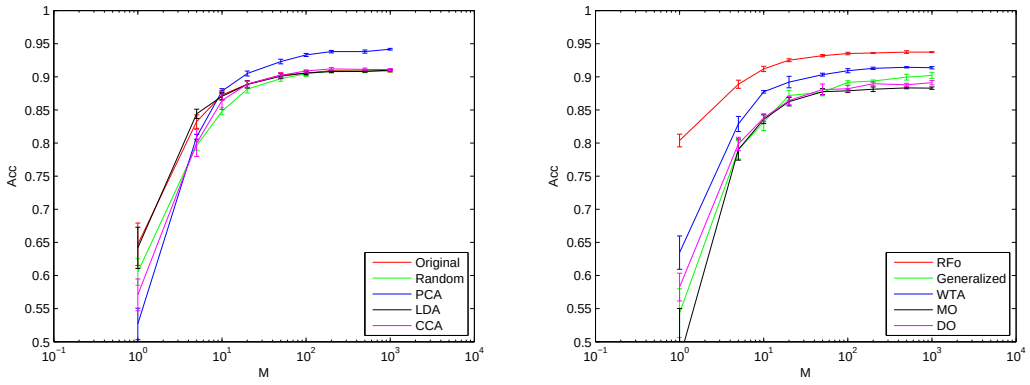
Further we evaluate the influence of the number of features on the classification accuracy. Therefore we keep the number of features per fern constant and increase the overall number of features by using more ferns M. The number of features per fern is set to the optimal parameters for the dataset found by using a cross-validation over the parameter space. The number of ferns is varied with $M \in \{1, \dots 1000\}$, thus using $S \cdot M$ features total. For each number of ferns we randomly train 5 times and report the average classification accuracy.
The results are shown in Figure 34 for the USPS dataset. The classification accuracy increases asymptotically with the number of base classifiers, i.e. ferns, for all of our Ferns classifiers. This property is already familiar from the RFo [9]. This characteristic of the classifier is independent of the dataset and its properties, and can be shown for all datasets. Also the standard deviation of the classification accuracy declines with more base classifiers. However, increasing the number of features takes more time and only increases the accuracy asymptotically. For selecting the optimal number of base classifiers we can refer to e.g. [99].

For $M = 1$ the plot also shows the "strength" of a single fern. Of course, the RFo creates the strongest base classifier within the ensemble, because it chooses the best



(a) Oblique Ferns. (b) Ordinal Ferns.

Figure 33: Classification accuracy with its standard deviation $Acc \pm Acc_\sigma$ for the same number of features overall but different number of features per fern. Shown for the Isolet dataset.



(a) Oblique Ferns. (b) Ordinal Ferns.

Figure 34: Classification accuracy with its standard deviation $Acc \pm Acc_\sigma$ over the number of base classifiers M for a fixed number of features per fern. Shown for the USPS dataset.

features for each tree node but the improvement from one to multiple base classifiers is small. However, PCA has the lowest classification accuracy of a single Oblique Fern, but can increase the accuracy most. LDA Ferns, WTA Ferns and original RFe have the strongest base classifier of which WTA has the strongest ensemble.
It is often assumed, that each base classifier, i.e. fern in our case, provides an independent prediction by using different random features per fern. However, this assumption does not hold because the random feature selection does not necessarily provide independent ferns [9].
The LDA Ferns provide strong base classifiers, but their accuracy saturates faster with the number of ferns M, because the ferns are less independent. LDA derives a similar subspace for each fern from the different random input dimensions, i.e. the subspace that separates the classes best. However, the PCA subspace provides more independent ferns, because the principal components depend stronger on the chosen input features, and thus deriving different projections for each set of input features. This benefits the ensemble structure, because the ferns are more independent and thus the saturation effect starts with a higher number of ferns M. The WTA Ferns have strong base classifiers and can increase the accuracy with M further, as they have more possible window configurations than binary comparisons of the original RFe, i.e. the number of possible features $\binom{D}{2} < \binom{D}{K}$ if $K > 2$. This increased number of possible features leads to less correlated ferns. Although the base classifiers of the generalized RFe and the Maximum Order Ferns are the weakest, they can increase the accuracy for a higher number ferns in the ensemble even further, because the randomized features provide less correlated ferns.

In the derivation of the Semi-Naive Bayesian framework in Section 3.1 we assumed an even class distribution and thus have not accounted the prior probabilities. For most of our problems this assumption holds, but some machine learning datasets have a non-even class distribution. However, evaluations have shown that this assumption still works well, because adding priors shifts the classification results towards the major classes, causing worse results at the minor class. The Dirichlet prior of the frequency matrix already accounts for these non-balanced cases.

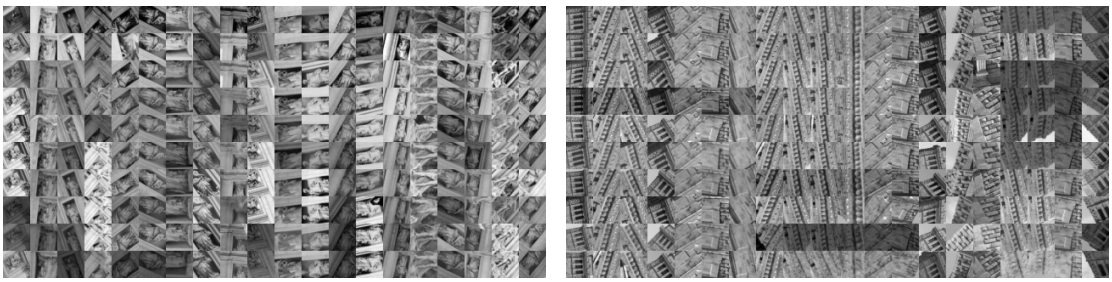
4.2 Interest point recognition

Özuysal et al. [84] initially proposed the RFe classifier for planar object tracking, which is interest point recognition in the broader context. Thus we also evaluate our embeddings for the purpose of interest point recognition. Therefore we first evaluate 3D interest point recognition and further planar object tracking using our proposed Ferns classifiers.

4.2.1 3D interest point recognition

For the evaluation of 3D interest point recognition we use the datasets named *Trevi* and *Liberty* provided by Winder and Brown [120]. Each dataset contains over $100k$ image patches, which are sampled from a 3D scene reconstruction. Therefore the 3D points are back-projected into the original images and a $64 \times 64px$ patch $\mathbf{P} \in \mathbb{R}^{4096}$ is extracted around the back-projected point. Thus each 3D point $\mathcal{P}$ is associated with 2 to over 50 patches which show the same 3D point, but from different viewpoints, scene brightness or partial occlusions. Figure 35 shows the most common image patches for different

datasets. All patches share the same canonical scale and orientation. This procedure is idealized because given an actual interest point detector, the localization, orientation and scale accuracy is much lower. Nevertheless the datasets provide a good starting point and basis for an evaluation. The Liberty dataset contains over $450k$ patches and the Trevi dataset over $100k$. The Liberty dataset is considered more complex, because it contains much repetitive structure which induces similarities between different 3D points and it contains less patches per 3D point than the Trevi dataset.



(a) Trevi. (b) Liberty.

Figure 35: Sample input images each $64 \times 64px$ for the datasets Trevi and Liberty. Each column depicts patches associated with one 3D point.

Each 3D interest point can be represented in different ways, as it consists of many patches. Therefore we train our classifiers using the most common representations which include mean [94] or median [67] over all patches of the same 3D point, or all patches [69]. We also extract the SIFT descriptor [71] from the patches and evaluate the classification performance in the 128-dimensional feature space. We use the implementation of [110] to calculate the SIFT descriptors. Therefore we locate the interest point in the center of the patch and calculate the Difference of Gaussians (DoG) interest point orientation, which corresponds to the dominant gradient orientation. The scale is chosen such that the patch is fully covered by the descriptors' spatial support. The descriptor is then extracted given the position, orientation and scale.

We evaluate our classifiers on the dataset using the different interest point representations. Therefore given a patch $\mathbf{P}$, we want to assign the corresponding 3D point $\mathcal{P}$ such that $g : \mathbf{P} \rightarrow \mathcal{P}$ with $\mathcal{P} \in \{1, \dots H\}$. Thus each 3D point is represented by an individual class and all patches associated with the 3D point share the same class label. From each dataset we use the H 3D points which have most associated image patches. This benefits the training because more samples are available. All experiments are performed as 10-fold cross-validation. Again, we report the average classification accuracy as performance criterion.

As baseline we use a k-Nearest Neighbors (kNN) classifier, which is commonly used in 3D interest point recognition [69, 67]. A true kNN is computationally not feasibly for such high dimensional data in huge dataset, thus we use an approximative kNN algorithm named Fast Approximate Nearest Neighbor (FLANN) [77], which is much faster and requires less memory at comparative classification performance.

The gray-value patches are normalized to zero mean and unit variance for the kNN and the Oblique Ferns. The Ordinal Ferns are invariant to this normalization by design and we use the raw pixel values. The used optimal parameters are acquired using a

cross-validation over a set of parameters, which is sparsely evaluated due to runtime and memory restrictions as these datasets are excessive in both.

First we evaluate our different RFe embeddings and then the different interest point representations. The results of the evaluation comparing different embeddings is shown in Table 18. Due to memory restrictions on the evaluation platform we are limited to $< 10k$ patches. This leads to a different number H of 3D points for Trevi $H = 400$ and Liberty $H = 1000$, which is caused by the different number of patches per 3D point. Liberty has a much lower number of patches per 3D point, thus more 3D points are considered.
The classification accuracy for the Trevi dataset is high for all classifiers, because the patches of one interest point are very similar. However, some of our RFe classifiers clearly outperform the kNN baseline. The LDA Ferns significantly outperform the original RFe as well as the baseline kNN. Due to the small amount of training samples per 3D point and the high intra-class similarity, LDA can create a discriminative subspace which benefits the classification, as already discussed for the face detection. Due to the low amount of training data, the empirical variances of the training data used by PCA are mostly caused by noise and do not resemble the real variances which worsens the result.
The other Oblique Ferns do not significantly differ from the baseline RFe. The WTA Ferns degenerate to an original RFe with $K = 2$, which provides the best results for the WTA Ferns. Using a higher window size K leads to overfitting due to the low number of training samples combined with the increasing number of leaves. The same applies for the Maximum Order Ferns, but as they apply a weaker leaf assignment function, the results are slightly worse. The Direct Ordinal Ferns perform similar to the baseline RFe. However, weak order does not benefit the classification, as there are hardly any ties in these natural images. The results are yet worse as discussed before, due to the influence of the Dirichlet prior as we have a small amount of training data and the Ordinal Direct Ferns require more training samples.
For Oblique Ferns and kNN we use normalized gray-scale patches. This provides a basic invariance to light changes, and thus improves the kNN accuracy significantly by over 10%.
The biggest problem for these datasets is the small amount of training samples per 3D point, i.e. ≈ 9 for the Liberty and ≈ 20 for the Trevi dataset. This makes it hard for all RFe to estimate the conditional probabilities. The chosen parameters of the Ordinal Ferns show, that they only model a small number of joint probabilities, i.e. the number of features per fern S is low. A higher number leads to overfitting, and further the samples per leaf decrease for a higher number of features per fern. Further we use a high input dimensionality of 4096, where the dependence between the input features is lower, thus requiring a lower number S of features per fern. An analysis of the correlation between the input pixels shows, that pixels are mostly correlated with neighboring pixels. Thus using a higher dimensional input decreases the chance of choosing a correlated pixel.

Next we compare different interest point representations. The results are shown in Table 19. For the Trevi dataset we use $H = 400$ 3D points and for Liberty $H = 1k$ 3D points, again limited due to memory restrictions. Further we evaluate Trevi with $H = 10k$ SIFT descriptors due to smaller memory requirements of the SIFT descriptors. We only use $H = 1k$ 3D points for Liberty because they are already hard to classify and for $H = 10k$ it would be still harder causing worse accuracy.

Classifier	Trevi $H = 400$ Acc	Liberty $H = 1k$ Acc	Parameters
kNN [77]	$95.38 \pm 0.78\%$	$14.60 \pm 0.95\%$	default
Original RFe [84]	$96.87 \pm 0.62\%$	$15.62 \pm 1.11\%$	$S = 5$
Generalized RFe [20]	$96.38 \pm 0.69\%$	$15.45 \pm 1.23\%$	$S = 9$
Random Oblique	$88.68 \pm 1.09\%\circ$	$12.36 \pm 1.17\%\circ$	$S = 12$, $T = 16$
PCA	$94.61 \pm 0.73\%\circ$	$15.52 \pm 1.58\%$	$S = 13$, $T = 32$
CCA	$94.48 \pm 0.14\%\circ$	$15.68 \pm 0.78\%$	$S = 8$, $T = 16$
LDA	$\mathbf{97.57 \pm 0.32\%}\bullet$	$\mathbf{19.20 \pm 0.71\%}\bullet$	$S = 12$, $T = 64$
WTA	$96.60 \pm 0.55\%$	$15.34 \pm 1.15\%$	$S = 5$, $K = 2$, $T = 4096$
Maximum Order	$95.08 \pm 0.46\%\circ$	$12.03 \pm 0.96\%\circ$	$S = 4$, $K = 3$, $T = 4096$
Direct Ordinal	$96.93 \pm 0.67\%$	$14.80 \pm 1.06\%$	$S = 5$, total order
	$96.09 \pm 0.74\%$	$14.70 \pm 0.96\%$	$S = 4$, weak order

Table 18: Evaluation results for Trevi and Liberty. Both evaluations use approximately $9k$ patches. We compare different embeddings by showing the average classification accuracy with its standard deviation $Acc \pm Acc_\sigma$.

We only evaluate a subset of the Ferns which have given good results in the previous evaluation. The used Ferns classifiers for this evaluation are the original RFe as baseline, the LDA Ferns, and the Direct Ordinal Ferns. We evaluate the LDA Ferns because they performed well in the previous experiment and the Direct Ordinal Ferns as they provide a different feature evaluation scheme in comparison to the WTA and Maximum Order Ferns. The WTA and Maximum Order Ferns were shown to degenerate to $K = 2$ for this experiment and thus perform equivalent to the original RFe, which is not interesting as we have the original RFe as baseline.
We use the parameters which were acquired above, i.e. $S = 5$ for the original RFe and Direct Ordinal Ferns with total order, and for the LDA Ferns $S = 5$ and $T = 8$ for the case of mean and median representation in order to account for the reduced training set. The results show that our LDA Ferns perform better than the other Ferns for most of the experiments. However, they are not well suited for real-time applications as they need normalized patches and require a computationally more complex training.

Using SIFT descriptors increases the accuracy of the classifiers as shown for the Liberty with $H = 1k$. The increase is significant for all but the LDA Ferns. The descriptor provides a more discriminative representation at lower dimensionality compared to the image patch. While the accuracy for the mean and median representations decreases for the image patches, the accuracy increases for this descriptor.

For the mean representation we first calculate the mean and then round to the next integer. The mean and median are calculated over the training set and the test set is not included for the calculation. In case of LDA Ferns and kNN we do not round the mean or median, as they need normalized data with higher precision anyway. For the Trevi dataset the mean and median representations get worse results in all cases. However, for the Liberty dataset they achieve better results. The mean and median representations are not well suited for the RFe classifier, as there is only one sample per 3D point (class) for training. Thus the kNN performs best for this representation for most of the datasets.

Direct Ordinal Ferns with $S = 5$ have more leaves as the original RFe, thus they score worse results for mean and median representation due to increased overfitting.

We also evaluate the benefits of generating artificial training images. Therefore we generate 50 views per patch as described in Section 4.2.2, but we only use the central $15 \times 15px$ area of the patch due to border artifacts caused by the affine transformation and smaller memory requirements. A mean and median representations for the synthetically generated views is not meaningful, as the reason for generating them is to use a diversity of training samples. The usage of the generated training samples usually benefit the classification. However, generating 50 views does not seem to be enough to calculate representative leave distributions, as the original RFe performs at $23.66 \pm 1.23\%$ for the Liberty dataset with $H = 500$. Due to the similar patches for different 3D points, the random affine transformations induce even more similarities that worsen the results. For this evaluation the Ordinal Direct Ferns perform best, as they resemble the strongest classifier among the evaluated for $S = 5$. Also the kNN cannot cope well with this amount of data, because the additional samples induce possibly contradictory nearest neighbors.

Dataset H 3D points	Classifier	Acc		
		all	mean	median
Trevi $H = 400$ 9170 patches	kNN [77]	$95.38 \pm 0.78\%$	$\mathbf{92.71 \pm 0.64}\%$	$\mathbf{90.52 \pm 1.32}\%$
	Original RFe [84]	$96.87 \pm 0.62\%$	$85.88 \pm 1.29\%$	$84.85 \pm 1.20\%$
	LDA Ferns	$\mathbf{97.57 \pm 0.32}\%$	$87.47 \pm 0.75\%$	$87.08 \pm 1.29\%$
	Direct Ordinal Ferns	$96.93 \pm 0.67\%$	$76.22 \pm 1.37\%$	$75.15 \pm 1.57\%$
Trevi SIFT $H = 10k$ 58638 descriptors	kNN	$\mathbf{69.11 \pm 0.46}\%$	$54.37 \pm 0.64\%$	$54.54 \pm 0.63\%$
	Original RFe	$54.26 \pm 0.53\%$	$53.34 \pm 0.94\%$	$57.14 \pm 0.69\%$
	LDA Ferns	$57.26 \pm 0.98\%$	$\mathbf{56.13 \pm 0.45}\%$	$\mathbf{57.53 \pm 0.69}\%$
	Direct Ordinal Ferns	$55.69 \pm 0.77\%$	$49.50 \pm 0.74\%$	$54.05 \pm 0.62\%$
Liberty $H = 1k$ 8623 patches	kNN	$14.60 \pm 0.95\%$	$\mathbf{17.78 \pm 1.08}\%$	$\mathbf{16.83 \pm 1.20}\%$
	Original RFe	$15.62 \pm 1.11\%$	$15.09 \pm 1.28\%$	$14.78 \pm 1.19\%$
	LDA Ferns	$\mathbf{19.20 \pm 0.71}\%$	$16.00 \pm 1.21\%$	$16.45 \pm 1.42\%$
	Direct Ordinal Ferns	$14.80 \pm 1.06\%$	$12.40 \pm 0.80\%$	$12.68 \pm 1.16\%$
Liberty SIFT $H = 1k$ 8623 descriptors	kNN	$19.47 \pm 1.33\%$	$24.48 \pm 1.16\%$	$\mathbf{23.67 \pm 0.97}\%$
	Original RFe	$20.74 \pm 1.14\%$	$23.13 \pm 1.41\%$	$23.29 \pm 1.00\%$
	LDA Ferns	$\mathbf{20.97 \pm 0.87}\%$	$\mathbf{25.06 \pm 1.58}\%$	$23.29 \pm 1.21\%$
	Direct Ordinal Ferns	$20.65 \pm 0.94\%$	$22.05 \pm 1.28\%$	$22.63 \pm 0.88\%$
Liberty synth. $H = 500$ 50 views per patch 231150 patches	kNN	$17.95 \pm 1.09\%$	–	–
	Original RFe	$15.54 \pm 1.05\%$	–	–
	LDA Ferns	$17.97 \pm 1.26\%$	–	–
	Direct Ordinal Ferns	$\mathbf{21.05 \pm 1.29}\%$	–	–

Table 19: Evaluation results for 3D interest point recognition comparing different embeddings and interest point representations by showing the average classification accuracy with its standard deviation $Acc \pm Acc_\sigma$.

4.2.2 Planar object tracking

Next we evaluate the RFe classifiers for planar object tracking. Therefore we follow the protocol of [65, 83]. We use a user-defined planar template that should be detected consecutively in each frame of a video.
We compare the original RFe to our WTA Ferns and Direct Ordinal Ferns. Although we have shown in the previous experiment that our LDA Ferns perform well too, they are not well suited for real-time tracking as they require memory intensive normalization (floating points) and considerable time for training the subspace.

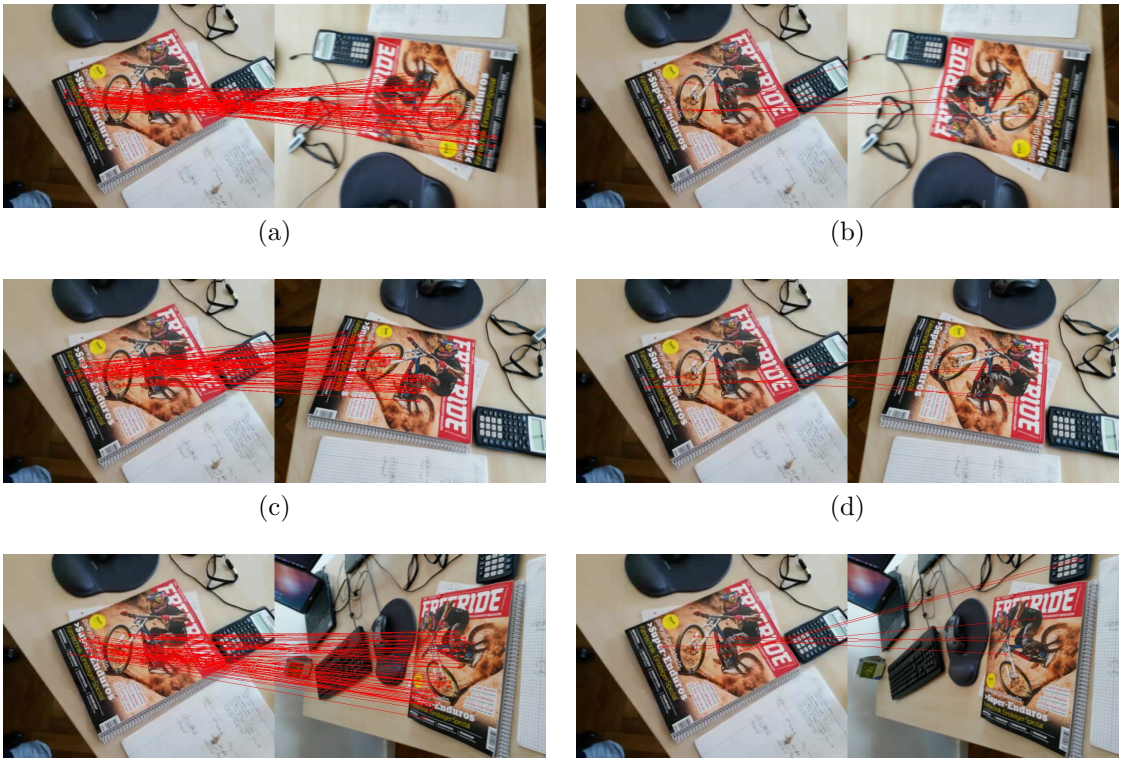
We use the planar object tracking workflow of [65, 83]. First we extract the most stable interest points from the template. Therefore we create 5000 random affine transformations of the original template and run a interest point detector on each transformed image. The random affine transformations are generated by warping the image with an affine transformation matrix $\mathbf{A} = \mathbf{R}_\theta \mathbf{R}_\phi^{-1} \mathbf{S} \mathbf{R}_\phi$. $\mathbf{R}_\theta$ and $\mathbf{R}_\phi$ are two rotation matrices which are parametrized by the angles θ and ϕ, and $\mathbf{S} = diag(\lambda_1, \lambda_2)$ is a scaling matrix. The parameters are randomly chosen from $\theta \sim \mathcal{U}(0, 2\pi)$, $\phi \sim \mathcal{U}(0, \pi)$, and $\lambda \sim \mathcal{U}(0.5, 1.5)$. For the interest point detection we use an approximated Laplacian of Gaussian detector [64] and run the detector on each level of a Gaussian pyramid [10] with 4 levels and a 7×7 Gaussian kernel with $\sigma = 2^{2/3}$. The most stable interest points are derived by back-projecting the detected interest points of the warped images into the original image and checking if the distance between detected and back-projected interest points is smaller than $2px$. The 400 interest points that satisfy this condition at most are then used to represent the model.

Next we train our Ferns classifiers. Therefore we use these most stable interest points and generate 10 000 random affine transformations of the template. We add white Gaussian noise within the range of $[-20, 20]$ to the image and apply a Gaussian smoothing with a randomly chosen kernel size of $\sigma \sim \mathcal{U}(3, 7)$. The patches of size $32 \times 32px$ around the interest point locations, which are the stable interest point locations projected into the generated images, are extracted and further used to train the classifier, as already described in Section 3.1.

Finally we use the trained classifier for planar object tracking. Therefore we detect a maximum of 1000 interest points in each video frame at 4 scales and classify each interest point. Given the initial correspondences between video frame interest points and model interest points, the homography is estimated using the Progressive Sample Consensus (PROSAC) method [15] with 1500 iterations and an inlier threshold of $10px$. The template is considered detected if at least 10 inliers are found.

Figure 36 shows some of the inlier correspondences for individual image frames. Each pair shows the correspondences between the model image on the left side and the video frame on the right side. For the left image pairs we used our Direct Ordinal Ferns and for the right image pairs the original RFe, both using $S = 7$ features per fern and $M = 20$ ferns. Our Direct Ordinal Ferns encodes more inequalities in each fern, i.e. they provide stronger base classifiers, as shown in Section 3.3.3, thus providing more correspondences for the selected video frames.

Next we evaluate the number of tracked interest points, i.e. the inliers of the homography estimation, which is the performance criterion for this evaluation. The more interest



(a) (b) (c) (d) (e) (f)

Figure 36: Detected interest points for individual frames and the correspondences to the model frame shown as red lines. The model image is on the left side. (a), (c), and (e) show the matches using the Direct Ordinal Ferns, and (b), (d), and (f) show the matches for the original RFe. Both methods use $S = 7$ features per fern and $M = 20$ ferns.

points are tracked as inliers, the better the estimation of the homography. Therfore we perform two evaluations.

1. First we evaluate the number of tracked interest points for the same number of features per fern. Therefore we use $S = 7$ features per fern for all Ferns classifiers and a window size of $K = 4$ for the WTA Ferns.

2. Second we evaluate the number of tracked interest points for the same amount of used memory. Therefore we use $S = 7$ for the Direct Ordinal Ferns, $S = 12$ for the original RFe, and for the WTA Ferns we use $K = 4$ and $S = 6$. The number of leaves is $7! = 5040$ for the Direct Ordinal Ferns, $2^{12} = 4096$ for the original RFe and $4^6 = 4096$ for the WTA Ferns, which is the closest comparable set of parameters.

The results for the first evaluation are shown in Figure 37. Our two embeddings perform significantly better when using the same number of features than the original RFe, because the Direct Ordinal Ferns as well as the WTA Ferns encode more inequalities per feature, thus leading to stronger ferns. Further, by using 10 000 training samples per interest point the conditional probabilities can be sufficiently estimated. The runtime is similar for all of the three compared Ferns classifiers, each performing at about $12 fps$ on the evaluation system with a video resolution of $640 \times 480 px$.

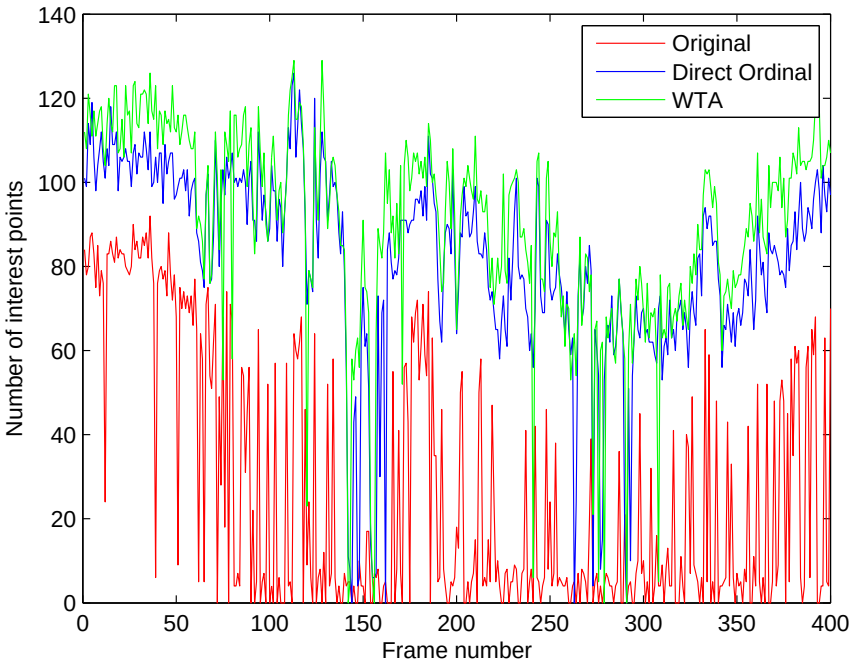


Figure 37: Tracked interest points (inliers) throughout the video comparing the original RFe with our Ferns classifiers. All use the same number of features $S = 7$ and $M = 20$ ferns.

The results for the second evaluation are shown in Figure 38. By using the same amount of memory all Ferns classifiers perform similar, with a few fluctuations. The stronger classifiers of the previous experiment are created at the cost of memory, which

is depicted in Figure 39. In this case the required memory is proportional to the number of leaves per fern. The WTA and Direct Ordinal Ferns encode more inequalities per leaf, i.e. provide stronger classifiers, than the original RFe for a small number of features per fern. However, for a higher number of features per fern, the memory requirements of the WTA and Direct Ordinal Ferns increase much faster than for the original RFe, which decreases the number of inequalities per leaf significantly.

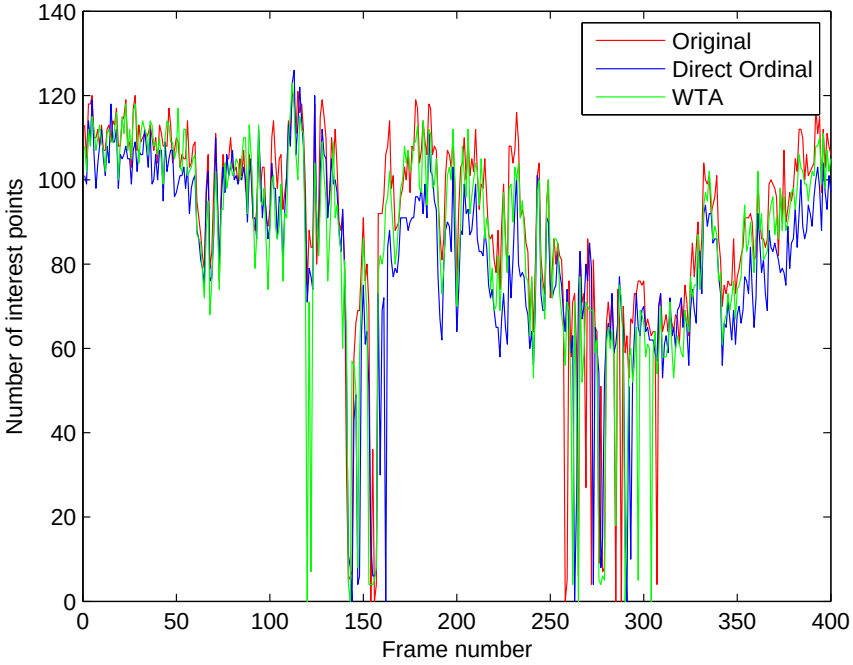


Figure 38: Tracked interest points (inliers) throughout the video comparing the original RFe with our Ferns classifiers by using the same amount of memory to store the leaf probabilities.

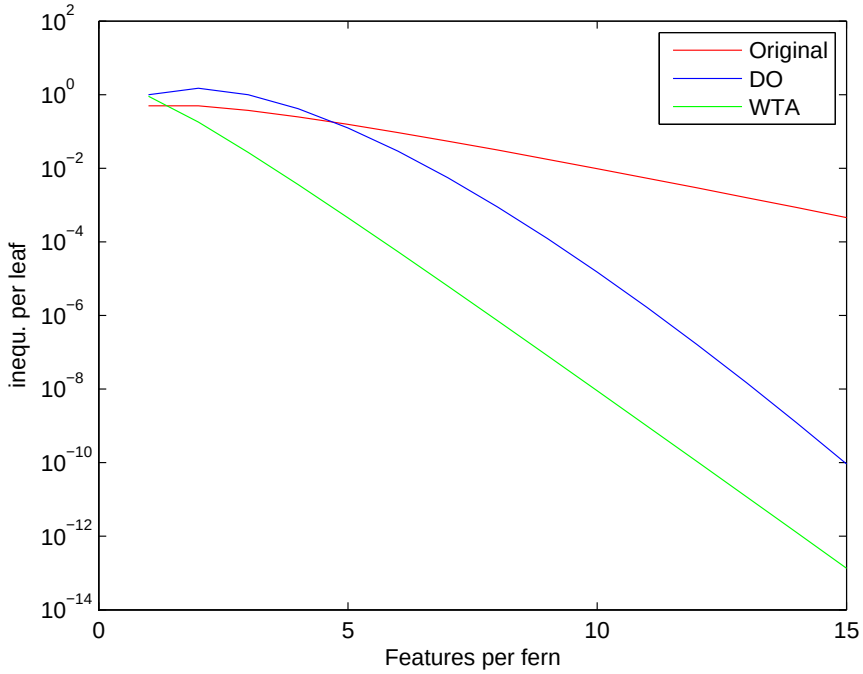


Figure 39: Number of inequalities encoded per leaf.

4.2.3 Oxford dataset

In order to evaluate the interest point recognition in a more controlled environment we use the Oxford dataset [75] of which we use the *Graffiti* sequence. As the number of tracked interest points, i.e. the inliers, in the previous evaluation can be influenced by several parameters of the PROSAC algorithm, e.g. the number of iterations or the inlier threshold, we directly evaluate the accuracy of the classification by calculating the fraction of the number of correctly classified interest points to all detected interest points in the reference image. The Oxford dataset is designed for interest point recognition and consists of image sequences of mostly planar objects taken from different viewpoints. This perspective change between the images is resembled by a homography. The homography is known for all images with respect to the reference image.

For the evaluation we use a Harris corner detector [40] and detect the 200 most prominent corners in the reference image, where the points close to the borders are discarded. We extract $15 \times 15px$ patches around the interest points and train our classifiers by using the same procedure as described in Section 4.2.2 by artificially creating 100 warped samples for each patch. Note, that we do not use an affine covariant detector and thus we do not normalize the patches around the keypoints. Further we transform the interest point locations of the reference image $\mathbf{p}$ into the other images $\mathbf{p}'$ of the sequence by using the ground-truth homography $\mathbf{H} \in \mathbb{R}^{3\times3}$ with

$$\mathbf{p}' = \mathbf{H}\mathbf{p}, \quad \mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{p}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{56}$$

in homogeneous coordinates. Then we query our trained classifiers with the patches around these calculated interest point locations from the other images of the sequence. Figure 40 shows some of the artificially created image patches used for training.

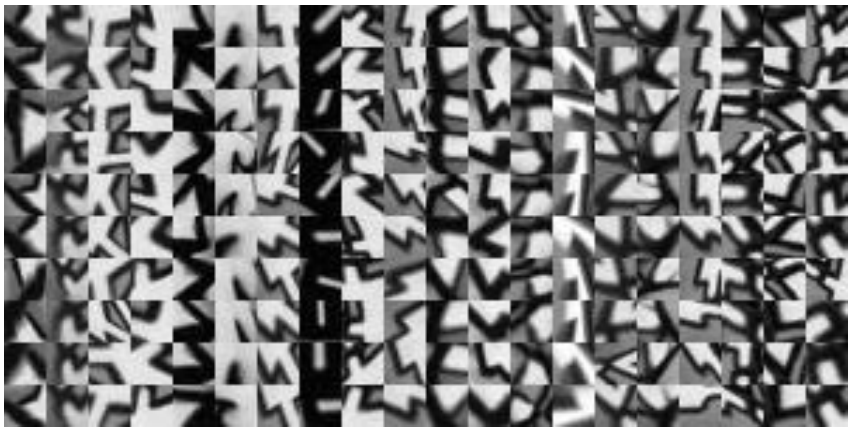


Figure 40: Artificial input patches each $15 \times 15px$ used for training. Each column resembles the generated samples for one interest point.

Again, we compare the original RFe with our WTA Ferns and the Direct Ordinal Ferns by using the same number of features per fern $S = 7$ and for the WTA Ferns we use a window size of $K = 4$.
The results are shown in Figure 41, which are similar to tracking experiment before, but in this experiment we obtained the matching percentage and not the number of tracked interest points. The WTA Ferns outperform both, the original RFe and the Direct Ordinal Ferns. However, the accuracy decreases for all Ferns similarly with greater viewpoint changes, i.e. higher image number. The results are consistent with the tracking results before. The original RFe perform worst. Though there is a larger difference between WTA and Direct Ordinal Ferns, that can be accounted to the number of training samples. In this example we use 100 artificial training samples per interest point and for the tracking application 10000 due to the better performance. As shown in Section 4.1.5 the Direct Ordinal Ferns perform inferior for a small number of training samples.

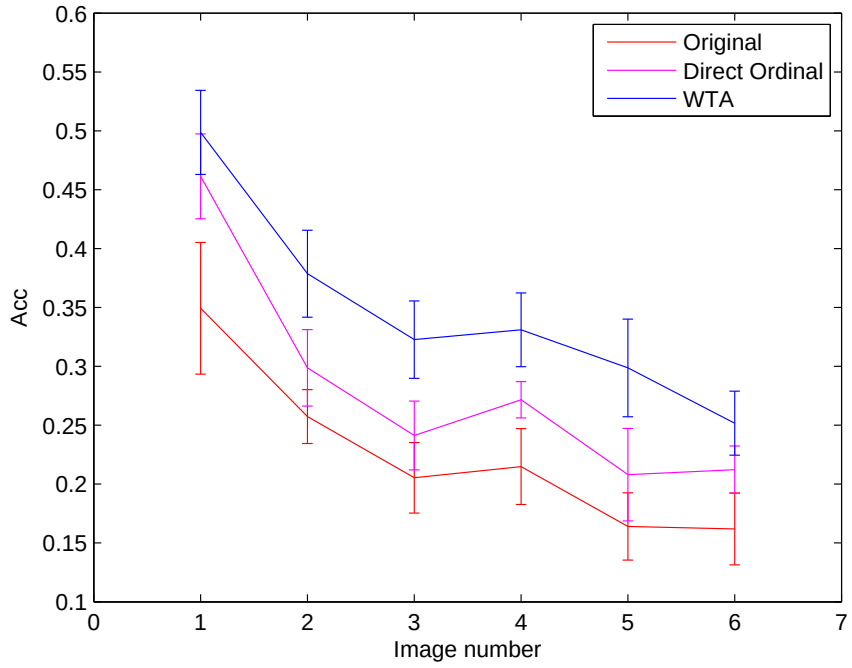


Figure 41: Average classification accuracy over the different views for the Graffiti sequence.

4.3 Guidelines for using embeddings

In the previous sections we have shown different applications of our Ferns classifiers and their performance on the different tasks. Finally we summarize these results in order to define possible criteria for an application. Notable criteria for an application can be

1. the number of classes,
2. the data dimensionality,
3. dataset properties,
4. required data normalization,
5. the number of training samples,
6. the number of adjustable parameters,
7. computational complexity, or
8. memory requirements.

A comparison of the different Ferns classifiers based on these criteria is shown in Table 20. Note, that we compare the criteria with respect to the other mentioned Ferns classifiers and not to classifiers in general.
The original RFe provides a very simple ordinal embedding, that consists of a single comparison, and thus it is invariant to linear transformations of the input data. It has only

two parameters to set, which are the number of ferns and the number of feature per fern, i.e. the degree of modeling joint probabilities. It does not require any normalization of the input data, though all input features must be derived from the same feature scale. Due to the simple feature comparison it has a low computational complexity as well as a low memory footprint.
The generalized RFe uses random thresholds for leaf assignment. Thus it is efficient for low dimensional data, as it can generate many random features using the random thresholds. However, the input data must be normalized within a certain range, which is an additional parameter besides the number of ferns and the number of features per fern. Again, the computational complexity and memory requirements are low.
The Oblique Ferns work well on high dimensional data. They all need normalized data, e.g. to zero mean and unit variance. The subspace dimensionality is an additional parameter that has to be specified. Further, LDA and CCA Ferns prefer a high number of classes as the subspace dimensionality depends on the number of classes. LDA and CCA Ferns perform especially well on statistically independent or orthogonal data, e.g. PCA compressed data, and PCA Ferns work well on highly correlated data. The Random Oblique Ferns work well on correlated, low dimensional datasets, because they create different views on the data which causes less correlation between the ferns. The computational complexity of the Ordinal Ferns is slightly increased, as they require a subspace projection, i.e. vector multiplication, but the memory requirements are still low.
The Ordinal Ferns use more sophisticated ordinal embeddings. High dimensional data is preferred in order to generate a large number of features. Although the data properties are irrelevant, the evaluations have shown that they perform well on image or histogram data and not very well on orthogonal, e.g. PCA compressed, data for example. They do not need normalized data, though all features must be derived from the same feature scale. The Soft Assignment Ferns work comparable to other Ordinal Ferns on noisy data, but inferior in the absence of noise. The required parameters depend on the Ferns classifier, thus WTA and Maximum Order Ferns need an additional window size and the subspace dimensionality, which can be set to the full input dimensionality without performance concerns. Direct Ordinal Ferns require the same parameters as original RFe plus the order, and the Soft Assignment as an extension of the Ordinal Direct Ferns requires two feature weighting parameters. The computational complexity is low for all Ordinal Ferns except the computationally complex soft assignment. The memory requirements are low for Maximum Order Ferns, which provide a similar parametrization as the original RFe. However, the memory requirements are slightly increased for WTA Ferns as the window size influences the used memory, and high for the Direct Ordinal and Soft Assignment Ferns as the memory consumption quickly increases with the number of features per fern.
Most of the Subspace Ordinal Ferns properties depend on the applied subspace method as well as the used ordinal embedding. In all cases normalized data is required for the subspace projection. If a CCA or LDA subspace is used, the number of classes should be high in order to generate a high subspace dimensionality, which is desired for the ordinal embeddings.
All Ferns need a large amount of training samples, though the Direct Ordinal and Soft Assignment Ferns require more training samples in comparison to the other Ferns.

Classifier \ Criterion	1	2	3	4	5	6	7	8
Original RFe [84]	$\sim$	high	$\sim$	feat. scale	$\sim$	2 (M,S)	low	low
Generalized RFe [20]	$\sim$	low	$\sim$	range	$\sim$	3 (M,S,range)	low	low
Random Oblique	$\sim$	high	$\sim$	norm.	$\sim$	3 (M,S,T)	medium	low
PCA	$\sim$	high	corr.	norm.	$\sim$	3 (M,S,T)	medium	low
CCA	high	high	stat. indep.	norm.	$\sim$	3 (M,S,T)	medium	low
LDA	high	high	stat. indep.	norm.	low	3 (M,S,T)	medium	low
WTA	$\sim$	high	$\sim$	feat. scale	$\sim$	4 (M,S,T,K)	low	medium
Maximum Order	$\sim$	high	$\sim$	feat. scale	$\sim$	4 (M,S,T,K)	low	low
Direct Ordinal	$\sim$	high	$\sim$	feat. scale	high	2 (M,S)	low	high
Soft Assignment	$\sim$	high	noisy	feat. scale	high	2 (M,S,θ,λ)	high	high
Subspace Ordinal	$\sim$	high	$\sim$	norm.	$\sim$	3+ (M,S,T,...)	medium+	low+

Table 20: Comparison of characteristic properties of different Ferns classifiers. The criteria are denoted by 1 the number of classes, 2 the data dimensionality, 3 dataset properties, 4 required data normalization, 5 the number of training samples, 6 the number of adjustable parameters, 7 computational complexity, and 8 memory requirements.

5 Conclusion

In this work we have proposed different embeddings for the well-known Random Ferns (RFe) classifiers. All of our classifiers share similar advantages. They are inherently multi-class, provide a high computational performance, can handle multi-modal data, are tolerant to noise, and naturally handle high dimensional data by applying a randomized subspace principle.
In order to introduce the embeddings, we first generalized the RFe in an appropriate framework in which we integrated all of our proposed embeddings. Then we described our Oblique Ferns, that use different linear subspace projections as embeddings. The projections can be derived from different methods, as e.g. PCA, LDA or CCA. The subspace method allows our classifier to adapt to individual dataset properties, as e.g. correlated or well separable data. Further, the Oblique Ferns provide more emphasis on local structures of the data compared to a global subspace representation, by applying the subspace methods on randomly selected input dimensions.
Next we proposed different ordinal embeddings, i.e. Ordinal Ferns, that apply ordinal analysis on the input data to generate features for the classification. They do not rely on any numeric values but only on order relations, which make them invariant to different kinds of input perturbations. We presented different ordinal embeddings, e.g. one integrates an ordinal hash function, or another one uses the complete order of the input data directly.
Further we discussed a combination of these two embeddings by using ordinal analysis within the subspace.
At last we have evaluated the performance of our Ferns classifiers and compared them to baseline RFe and Random Forest (RFo) classifiers. We provided an extensive evaluation on computer vision and machine learning datasets, and analyzed the influence of the parameters on the classification accuracy as well as the influence of different dataset properties. We have shown, that the Oblique Ferns work well for machine learning and computer vision datasets. Especially the PCA Ferns work well on datasets with highly correlated features and the LDA Ferns work well for low intra-class variations and on orthogonal data, e.g. PCA compressed data. Further we have evaluated our Ferns for interest point recognition. We have shown for various applications that the Ordinal Ferns, especially our proposed WTA and Direct Ordinal Ferns, perform well in real-time applications.

Our generalized RFe framework provides manifold possibilities for future work, by integrating different embeddings. In the context of our Oblique Ferns formulation, one could evaluate alternative thresholding strategies, e.g. similar to [23], with respect to runtime and accuracy, or even go a step further and evaluate iterative strategies for obtaining the linear projections, e.g. like [74, 78]. In this work, however, we did not consider these iterative methods due to an increased computational complexity, but approximations might be possible that work reasonably well.
Moreover, there are extensions for our different Ordinal Ferns possible. For example Yagnik et al. [123] proposed polynomial kernels for the WTA hash, which could be evaluated for our WTA Ferns. As the Direct Ordinal Ferns suffer from high memory requirements, one could find possibilities to reduce the required memory, or to show, that there is an information theoretic boundary on the number of inequalities encoded per leaf.

References

[1] Stefan Aeberhard, Danny Coomans, and Olivier de Vel. Comparison of classifiers in high dimensional settings. Technical report, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland, 1992.

[2] Selim Aksoy and Robert M. Haralick. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognition Letters*, 22(5):563–582, 2001.

[3] Dinkar N. Bhat and Shree K. Nayar. Ordinal measures for visual correspondence. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 1996.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[5] Matthew B. Blaschko and Christoph H. Lampert. Correlational spectral clustering. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[6] Avrim Blum and Ronald L. Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117–127, 1992.

[7] Liefeng Bo, Ling Wang, and Licheng Jiao. Feature scaling for kernel fisher discriminant analysis using leave-one-out cross validation. *Neural Computation*, 18(4):961–978, 2006.

[8] Anna Bosch, Andrew Zisserman, and Xavier Muñoz. Image classification using random forests and ferns. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2007.

[9] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[10] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *Transactions on Communications*, 31(4):532–540, 1983.

[11] Deng Cai, Xiaofei He, Yuxiao Hu, Jiawei Han, and Thomas Huang. Learning a spatially smooth subspace for face recognition. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[12] Jaime S. Cardoso and Joaquim F. Pinto da Costa. Learning to classify ordinal data: The data replication method. *Journal of Machine Learning Research*, 8:1393–1429, 2007.

[13] Tat-Jun Chin and David Suter. Improving the speed of kernel pca on large scale datasets. In *Proceedings of International Conference on Advanced Video and Signal-Based Surveillance*, 2006.

[14] Yuan Shih Chow and Henry Teicher. *Probability Theory: Independence, Interchangeability, Martingales*. Springer, 3rd edition, 1997.

[15] Ondrej Chum and Jiri Matas. Matching with prosac - progressive sample consensus. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[16] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[17] Thomas L. Dean, Mark A. Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[18] Oliver Demetz, David Hafner, and Joachim Weickert. The complete rank transform: A tool for accurate and morphologically invariant matching of structures. In *Proceedings of British Machine Vision Conference (BMCV)*, 2013.

[19] David Deterding. *Speaker Normalisation for Automatic Speech Recognition*. PhD thesis, University of Cambridge, 1989.

[20] Piotr Dollár. Piotrs image and video matlab toolbox (PMT). `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`, 2013.

[21] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[22] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[23] Rong-En Fan and Chih-Jen Lin. A study on threshold selection for multi-label classification. Technical report, Department of Computer Science, National Taiwan University, 2007.

[24] Mark A. Fanty and Ronald A. Cole. Spoken letter recognition. In *Proceedings of Neural Information Processing Systems Conference (NIPS)*, 1990.

[25] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[26] Chen-Chieh Feng, Andrew Sutherland, Ross D. King, Stephen H. Muggleton, and Bob Henery. Comparison of machine learning classifiers to statistics and neural networks. In *Proceedings of International Workshop on Artificial Intelligence and Statistics*, 1993.

[27] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[28] Richard S. Forsyth. *PC/BEAGLE User's Guide*. BUPA Medical Research Ltd., 1990.

[29] Victor Fragoso, Matthew Turk, and Joao Hespanha. Locating binary features for keypoint recognition using noncooperative games. In *Proceedings of International Conference on Image Processing (ICIP)*, 2012.

[30] Eibe Frank, Mark Hall, and Bernhard Pfahringer. Locally weighted naive bayes. In *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

[31] Ce Gao, Yixu Song, and Peifa Jia. Multilayer ferns: A learning-based approach of patch recognition and homography extraction. In *Proceedings of International Conference on Machine Learning and Applications (ICMLA)*, 2010.

[32] James E. Gentle. *Numerical Linear Algebra for Applications in Statistics*. Springer, 1998.

[33] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[34] Todd R. Golub. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

[35] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[36] Paul R. Gorman and Terrence J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural networks*, 1(1):75–89, 1988.

[37] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-pie. In *Proceedings of International Conference on Automatic Face and Gesture Recognition*, 2008.

[38] Pedro Antonio Gutiérrez, Maria Pérez-Ortiz, Francisco Fernández-Navarro, Javier Sánchez-Monedero, and César Hervás-Martínez. An experimental study of different ordinal regression methods and measures. In *Proceedings of International Conference on Hybrid Artificial Intelligence Systems*, 2012.

[39] John Michael Hammersley and David Christopher Handscomb. *Monte Carlo Methods*. Methuen, 1975.

[40] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of Alvey Vision Conference*, 1988.

[41] Tin Kam Ho. The random subspace method for constructing decision forests. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(8):832–844, 1998.

[42] Tin Kam Ho and Eugene M. Kleinberg. Building projectable classifiers of arbitrary complexity. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, 1996.

[43] Joseph P. Hoffbeck and David A. Landgrebe. Covariance matrix estimation and classification with limited training data. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(7):763–767, 1996.

[44] Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. Spambase data set, 1999.

[45] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[46] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3-4):321–377, 1936.

[47] Torsten Hothorn, Friedrich Leisch, Achim Zeileis, and Kurt Hornik. The design and analysis of benchmark experiments. *Journal of Computational and Graphical Statistics*, 14(3):675–699, 2005.

[48] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

[49] Hui Hu, Jie Ma, and Jinwen Tian. A new electronic image stabilization technology based on random ferns for fixed scene. In Jingsheng Lei, FuLee Wang, Mo Li, and Yuan Luo, editors, *Network Computing and Information Security*, volume 345 of *Communications in Computer and Information Science*, pages 501–508. Springer, 2012.

[50] Jens C. Huhn and Eyke Hüllermeier. Is an ordinal class structure useful in classifier learning? *International Journal of Data Mining, Modelling and Management*, 1(1):45–67, 2008.

[51] Ian Jolliffe. *Principal Component Analysis*. Springer, 2002.

[52] Piotr Juszczak, David Tax, and Robert Dui. Feature scaling in support vector data descriptions. In *Proceedings of Annual Conference of the Advanced School for Computing and Imaging*, 2002.

[53] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(7):881–892, 2002.

[54] Arnold Knopfmacher and Michael E. Mays. A survey of factorization counting functions. *International Journal of Number Theory*, 1(4):563–581, 2005.

[55] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 1973.

[56] Donald E. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley Professional, 3rd edition, 1997.

[57] Igor Kononenko. Semi-naive bayesian classifier. In *Proceedings of European Working Session on Learning (EWSL)*, 1991.

[58] Miron B. Kursa. Random ferns method implementation for the general-purpose machine learning. *Computing Research Repository*, abs/1202.1121, 2012.

[59] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[60] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In *Proceedings of Neural Information Processing Systems Conference (NIPS)*, 1989.

[61] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[62] Suwon Lee, Sang-Wook Lee, Yeong Nam Chae, and Hyun Seung Yang. Lightweight random ferns using binary representation. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2012.

[63] Zhen Lei, ShengCai Liao, and Stan Z. Li. Efficient feature selection for linear discriminant analysis and its application to face recognition. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2012.

[64] Vincent Lepetit and Pascal Fua. Towards recognizing feature points using classification trees. Technical report, École Polytechnique Fédérale de Lausanne, 2004.

[65] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(9):1465–1479, 2006.

[66] Tao Li, Shenghuo Zhu, and Mitsunori Ogihara. Using discriminant analysis for multi-class classification. In *Proceedings of International Conference on Data Mining (ICDM)*, 2003.

[67] Yunpeng Li, Noah Snavely, and Daniel P. Huttenlocher. Location recognition using prioritized feature matching. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2010.

[68] Jens Liebehenschel. Ranking and unranking of lexicographically ordered words: An average-case analysis. *Journal of Automata, Languages and Combinatorics*, 2(4):227–268, 1997.

[69] Hyon Lim, Sudipta N. Sinha, Michael F. Cohen, and Matthew Uyttendaele. Real-time image-based 6-dof localization in large-scale environments. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[70] Shaoguo Liu, Haibo Wang, Jixia Zhang, Franck Davoine, and Chunhong Pan. Soft-ferns for homography estimation. In *Proceedings of International Conference on Image Processing (ICIP)*, 2011.

[71] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[72] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Efficient classification for additive kernel svms. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(1):66–77, 2013.

[73] Subhransu Maji and Jitendra Malik. Fast and accurate digit classification. Technical report, EECS Department, University of California, Berkeley, 2009.

[74] Bjoern H. Menze, B. Michael Kelm, Daniel Nicolas Splitthoff, Ullrich Köthe, and Fred A. Hamprecht. On oblique random forests. In *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2011.

[75] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(10):1615–1630, 2005.

[76] Moshe Mor and Aviezri S. Fraenkel. Cayley permutations. *Discrete Mathematics*, 48(1):101–112, 1984.

[77] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of International Conference on Computer Vision Theory and Applications (VISAPP)*, 2009.

[78] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

[79] Wendy J. Myrvold and Frank Ruskey. Ranking and unranking permutations in linear time. *Information Processing Letters*, 79(6):281–284, 2001.

[80] Michiel O. Noordewier, Geoffrey G. Towell, and Jude W. Shavlik. Training knowledge-based neural networks to recognize genes. In *Proceedings of Neural Information Processing Systems Conference (NIPS)*, 1990.

[81] Olusegun Oshin, Andrew Gilbert, John Illingworth, and Richard Bowden. Action recognition using randomised ferns. In *Proceedings of International Conference on Computer Vision Workshops (ICCV Workshops)*, 2009.

[82] Mustafa Özuysal. *Learning Pose Invariant and Covariant Classifiers from Image Sequences*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2010.

[83] Mustafa Özuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(3):448–461, 2010.

[84] Mustafa Özuysal, Pascal Fua, and Vincent Lepetit. Fast keypoint recognition in ten lines of code. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[85] Matthew Partridge and Rafael A. Calvo. Fast dimensionality reduction and simple pca. *Intelligent Data Analysis*, 2(1-4):203–214, 1998.

[86] Maja Pohar, Mateja Blas, and Sandra Turk. Comparison of logistic regression and linear discriminant analysis: a simulation study. *Metodolski Zvezki*, 1(1):143–161, 2004.

[87] Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.

[88] Deva Ramanan and Simon Baker. Local distance functions: A taxonomy, new algorithms, and an evaluation. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(4):794–806, 2011.

[89] Cong Rao, Cong Yao, Xiang Bai, Weichao Qiu, and Wenyu Liu. Online random ferns for robust visual tracking. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2012.

[90] Radhakrishna Rao. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society, Series B*, 10(2):159–203, 1948.

[91] Ryan M. Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[92] Joseph Lee Rodgers, Alan Nicewander, and Larry Toothaker. Linearly independent, orthogonal, and uncorrelated variables. *The American Statistician*, 38(2):133–134, 1984.

[93] Ferdinando S. Samaria and Andy C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of Workshop on Applications of Computer Vision*, 1994.

[94] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Improving image-based localization by active correspondence search. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2012.

[95] Jeffrey Curtis Schlimmer. *Concept acquisition through representational adjustment.* PhD thesis, Department of Information and Computer Science, University of California, 1987.

[96] Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Proceedings of International Conference on Artificial Neural Networks (ICANN)*, 1997.

[97] Bernd Schroeder. *Ordered Sets: An Introduction.* Birkhäuser, 2002.

[98] Samuel Schulter, Peter M. Roth, and Horst Bischof. Ordinal random forests for object detection. In *Proceedings of German Conference Pattern Recognition (GCPR)*, 2013.

[99] Alexander G. Schwing, Christopher Zach, Yefeng Zheng, and Marc Pollefeys. Adaptive random forest - how many "experts" to ask before making a decision? In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[100] Robert Sedgewick. Permutation generation methods. *ACM Computing Surveys*, 9(2):137–164, 1977.

[101] Toby Sharp. Implementing decision trees and forests on a gpu. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2008.

[102] Vincent G. Sigillito, Simon P. Wing, Larrie V. Hutton, and Kile B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10:262–266, 1989.

[103] Patrice Simard, Yann LeCun, and John S. Denker. Efficient pattern recognition using a new transformation distanc. In *Proceedings of Neural Information Processing Systems Conference (NIPS)*, 1992.

[104] Jack W. Smith, James E. Everhart, William C. Dickson, William C. Knowler, and Richard S. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of Symposium on Computer Application in Medical Care*, 1988.

[105] Christoph Strecha, Alexander M. Bronstein, Michael M. Bronstein, and Pascal Fua. Ldahash: Improved matching with smaller descriptors. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(1):66–78, 2012.

[106] Nick Street, William H. Wolberg, and Olvi L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Proceedings of Symposium on Electronic Imaging: Science and Technology*, 1993.

[107] Liang Sun, Shuiwang Ji, and Jieping Ye. Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(1):194–200, 2011.

[108] Feng Tang, Suk Hwan Lim, Nelson L. Chang, and Hai Tao. A novel feature descriptor invariant to complex brightness changes. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[109] Sweta Thakur, Jamuna Kanta Sing, Dipak Kumar Basu, Mita Nasipuri, and Mahantapas Kundu. Face recognition using principal component analysis and rbf neural networks. In *Proceedings of International Conference on Emerging Trends in Engineering and Technology (ICETET)*, 2008.

[110] Andrea Vedaldi and Brian Fulkerson. Vlfeat: an open and portable library of computer vision algorithms. In *Proceedings of International Conference on Multimedia*, 2010.

[111] Michael Villamizar, Juan Andrade-Cetto, Alberto Sanfeliu, and Francesc Moreno-Noguer. Bootstrapping boosted random ferns for discriminative and efficient object classification. *Pattern Recognition*, 45(9):3141–3153, 2012.

[112] Michael Villamizar, Anais Garrell, Alberto Sanfeliu, and Francesc Moreno-Noguer. Online human-assisted learning using random ferns. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2012.

[113] Michael Villamizar, Helmut Grabner, Francesc Moreno-Noguer, Juan Andrade-Cetto, Luc J. Van Gool, and Alberto Sanfeliu. Efficient 3d object detection using multiple pose-specific classifiers. In *Proceedings of British Machine Vision Conference (BMCV)*, 2011.

[114] Michael Villamizar, Francesc Moreno-Noguer, Juan Andrade-Cetto, and Alberto Sanfeliu. Shared random ferns for efficient detection of multiple categories. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, 2010.

[115] Paul A. Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[116] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *Transactions on Visual Computing and Graphics*, 16(3):355–368, 2010.

[117] Xiaogang Wang and Xiaoou Tang. Random sampling lda for face recognition. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[118] Zhenhua Wang, Bin Fan, and Fuchao Wu. Local intensity order pattern for feature description. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2011.

[119] Brian Patrick Williams, Georg Klein, and Ian D. Reid. Real-time slam relocalisation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2007.

[120] Simon A. J. Winder and Matthew Brown. Learning local image descriptors. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[121] William H. Wolberg and Olvi L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences*, 87(23):9193–9196, 1990.

[122] Tao Xiong, Jieping Ye, Qi Li, Ravi Janardan, and Vladimir Cherkassky. Efficient kernel discriminant analysis via qr decomposition. In *Proceedings of Neural Information Processing Systems Conference (NIPS)*, 2004.

[123] Jay Yagnik, Dennis Strelow, David A. Ross, and Ruei-sung Lin. The power of comparative reasoning. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2011.

[124] Nayyar A. Zaidi, Jesús Cerquides, Mark James Carman, and Geoffrey I. Webb. Alleviating naive bayes attribute independence assumption by attribute weighting. *Journal of Machine Learning Research*, 14(1):1947–1988, 2013.

[125] Cheng-Yuan Zhang and Qiu-Qi Ruan. Face recognition using l-fisherfaces. *Journal of Computing and Information Science in Engineering*, 26(4):1525–1537, 2010.

[126] Fei Zheng and Geoffrey I. Webb. A comparative study of semi-naive bayes methods in classification learning. In *Proceedings of Australasian Data Mining Conference (AusDM)*, 2005.