Sketch Recognition with Few Examples

Kemal Tugrul Yesilbek, T. Metin Sezgin

Rumeli Feneri Mh., 34450 Istanbul, Turkey

Koc University

Abstract

Sketch recognition is the task of converting hand-drawn digital ink into symbolic computer representations. Since the early days of sketch recognition, the bulk of the work in the field focused on building accurate recognition algorithms for specific domains, and well defined data sets. Recognition methods explored so far have been developed and evaluated using standard machine learning pipelines and have consequently been built over many simplifying assumptions. For example, existing frameworks assume the presence of a fixed set of symbol classes, and the availability of plenty of annotated examples. However, in practice, these assumptions do not hold. In reality, the designer of a sketch recognition system starts with no labeled data at all, and faces the burden of data annotation. In this work, we propose to alleviate the burden of annotation by building systems that can learn from very few labeled examples, and large amounts of unlabeled data. Our systems perform self-learning by automatically extending a very small set of labeled examples with new examples extracted from unlabeled sketches. The end result is a sufficiently large set of labeled training data, which can subsequently be used to train classifiers. We present four self-learning methods with varying levels of implementation difficulty and runtime complexities. One of these methods leverages contextual co-occurrence patterns to build verifiably more diverse set of training instances. Rigorous experiments with large sets of data demonstrate that this novel approach based on exploiting contextual information leads to significant leaps in recognition performance. As a side contribution, we also demonstrate the utility of bagging for sketch recognition in imbalanced data sets with few positive examples and many outliers.

Keywords:
Sketch recognition, Learning from few examples, Self-learning

1. Introduction

Hand-drawn sketches are ubiquitous in design, arts, education and entertainment. More recently sketching has also been receiving attention as a natural human-computer interaction modality as seen from the continually increasing body of work on automated sketch recognition.

Sketch recognition is defined as the task of segmenting a full sketch into individual groups of ink representing domain symbols, and assigning labels denoting classes. State of the art approaches to sketch recognition are predominantly based on machine learning technologies. However, the development and evaluation of these algorithms have traditionally been carried out with strong assumptions that do not hold in practice.

For example, it is generally assumed that sufficiently large set of annotated symbols are readily available for training classifiers. In practice, however, such data is generally unavailable. Moving into a new domain requires the designer of the sketch recognition system to create an annotated data set. This is done either by collecting isolated instances of symbols from users [1, 2, 3, 4, 5], or by annotating full sketches [6, 7] (i.e., sketches consisting of multiple symbols). Both cases require substantial annotation effort. In this paper, we propose methods for training sketch recognizers using only a few (1-3) labeled examples. We do so by leveraging large sets of unlabeled examples. This ability of the proposed framework allows users of the system to define their own classes for an unlabeled data set on-fly, which offers great flexibility.

Although our main contribution addresses learning with few examples, our setup also challenges other assumptions in the field. It is generally assumed that recognizers will only be tested on symbols strictly within the domain of interest. This assumption manifests itself through the use of crisp multi-class data sets, or in the form of drawing instructions for users where they are first briefed about the set of available domain objects, and told not to use any symbols outside this restricted set. Hence, evaluation results in the literature are all reported in a multi-class classification setting where the knowledge of all classes are available. However, real drawings usually contain a large number of objects, marks, and writing that are irrelevant for the domain, and act as outliers. The learning framework we describe explicitly abstains from crisp data assumptions, and is evaluated with realistic sketch data containing many outliers.

Our approach is technically a semi-supervised method performing *self-learning*. Self-learning refers to using some amount of labeled data to label unlabeled instances, and training a classifier with the extended set of labeled instances. Generally self-

Email address: `kyesilbek@ku.edu.tr`, `mtsezgin@ku.edu.tr` (Koc University)

learners start with an initial seed set of 10 or more labeled examples per class, and extend the training data. However, we target very few examples (1-3 labeled examples). This results in two main challenges. First, with only 1-3 items in the initial list of labeled examples, it becomes essential that any additional items brought into the list do indeed belong to the correct class. Even a few incorrectly labeled examples can cause catastrophic drops in recognizer performance. Second, it is extremely important to ensure that the additional labeled items are not too similar to the existing examples. New labeled examples help only if they are diverse and carry variations. We show that a context-based selection criterion promotes diversity. The key insight that we bring is to give precedence to candidate examples that not only have the appearance of the class of interest, but also appear in contexts that are typically observed for the object of interest. This scheme favors diversity.

Learning from few examples also poses a data imbalance challenge. The number of positive examples are multiple orders of magnitude smaller than the number of negative and unlabeled examples. We address this issue through bagging (bootstrap aggregation).

Finally, we successfully adopt a Viola-Jones-like filtering scheme to speed up the self-learning process for large data sets. The filtering acts as a conservative rejection mechanism that excludes irrelevant unlabeled instances from the self-learning pipeline.

The focus on learning from very few examples distinguishes our work from others. The context-based self learning method is our main contribution. We demonstrate the utility of this approach through its ability to accurately select diverse examples for training sketch recognizers. Successful incorporation of bagging and conservative rejection serve as two additional contributions.

In the rest of the paper, we first put our work into perspective by discussing the related work from the sketch recognition domain. Since the use of realistic data is one of the core contributions of our work, we describe the in-the-wild sketch data set that we use in Section 3. We measure the feasibility of self-learning through many repeated experiments designed to mimic what would have happened if the process had started with various initial conditions. The Experimental Setup section describes the end-to-end pipeline for self learning, including the details of data preparation, and metrics for performance measurement. Section 5 describes the details of our context-based self-learning algorithm, along with three others. We report our findings in the Results section, conclude with a discussion of the main findings, a summary of our contributions and directions for future work.

2. Related Work

The historical progression of interest in sketch recognition started with investigation of knowledge-based and model-based recognition systems with no elements of machine learning [8, 9, 10, 11]. The focus later shifted to approaches based on machine learning. These methods proved to be superior, and the field enjoyed steady progress in feature representations and recognition architectures. It is only recently that the interest has shifted to alleviating the difficulties associated with approaches based on machine learning. Below we discuss how our work fits in this vast body of work on sketch recognition.

The early work on sketch recognition focused on building rule-based recognition algorithms. These approaches combined structural descriptions of symbols with efficient matching algorithms and rule-based interpretation architectures for recognition [8, 9, 10, 11]. Rather than learning from examples, they use knowledge based object models. For example, Mahoney et-al. [8] propose structural descriptions that describe domain objects in terms of connections and constraints defined over line segments, and use sub graph isomorphism for recognition [8]. Sezgin et-al. propose automatic generation of recognizer code from structural descriptions of domain objects [9]. Veselova and Hammond et-al. take the idea of structural descriptions further by defining a formal symbol representation language [11] and a perceptually inspired method for generating object descriptions from single hand-drawn examples [10]. The work of Veselova et-al. is in the same spirit as ours in the attempt to learn from few examples, however we operate within a machine-learning-based framework, and try to exploit unlabeled data.

With the development of powerful feature representations for sketches, recognition frameworks based on machine learning gained dominance [12, 13, 14, 2, 3]. These methods were developed and evaluated within the standard train/validate/test machine learning pipeline, and our work aims to address the limitations induced by the assumptions of these systems. These and many others ([1, 4, 5]) assume fully labeled training data sets consisting of isolated hand-drawn symbols instances. They assume a predetermined set of object categories, and focus on performance indicators measured over isolated symbols or scenes consisting of domain objects only. In contrast, we focus on learning from few examples, while symbols are not isolated (i.e. there exists multiple symbols in a sketch), and exploiting unlabeled data. Most of the work supporting sketch scenes with multiple objects assume that each object is drawn with a single stroke [15, 16]. While this assumption both reduces the complexity and increases the success rate of the techniques, it forces users to change their sketching style which affects usability negatively. To address this issue, our system follows a fragment-and-combine approach similar to [17].

The most relevant pieces of work to ours are those that try to exploit unlabeled examples [15, 18, 19]. All these systems assume a small seed set of labeled examples, and try to extend the number of labeled instances by automatically labeling unlabeled examples with the user in the loop. Technically these methods are active learning approaches, since they require user supervision. They starts with a low number of labeled instances, and allow the labeling of the mis-recognized instances [15], or ask for specific instances to be labeled [19] by the user. Unlike these, we do not rely on the user for labeling. We start with very few labeled instances and continue in a fully automated fashion. This makes the problem more challenging, since no user intervention is possible in case of errors in automatic instance labeling. Furthermore, these approaches mostly assume that the unlabeled data is already segmented, an assumption we explicitly avoid.

Within the machine learning and computer vision literature, there are plenty of approaches for zero shot learning, one shot learning, and transfer learning [20]. These approaches rely on attributes that serve as reusable models of object properties. Models for new objects are subsequently defined in terms of the previously learned attributes [21, 20]. Examples of work along these lines in the sketch recognition community include the work of Alvarado and Shilman et-al. [22, 23]. They model subparts of domain objects using distributions over features and reuse this information to build generative graphical models. These approaches have been disadvantaged by high computational requirements, and lower recognition rates compared to the learning-based approaches that came later (e.g., [12, 3, 24]). Furthermore, the inherently sparse, and ambiguous nature of sketches renders the tuning process of these generative models an art.

One notable transfer learning technique by Miller et-al. [25] proposes an alignment-based technique that works with a single example. The technique learns a probability density over the parameters of a family of affine transforms computed for a data set of many known symbols, and uses the estimated density to build a single-example classifier. This approach does not utilize unlabeled examples, however we include an algorithm inspired by the technique as a baseline in our evaluation.

Our method is similar to a number of other methods in their use of context, (e.g., [26, 27, 28]). However, we perform self-learning and not recognition.

Finally, there are feature representations and distance-based approaches for single stroke [24, 29] and multi-stroke [30, 31, 32, 33, 12, 2] gesture/symbol recognition. These methods do not exploit unlabeled data, and are generally used with many training examples within the traditional machine learning setup. However, they also serve as good feature representations. Hence, they can conceivably be modified to compute distances to build robust single-example classifiers in a nearest neighbor classification setup. To shed light into their efficacy in recognition, we include methods based on the well established Image Deformation Model (IDM) feature [34] in our evaluation.

Question type	Number of sketches
Balance	463
Money	379
Reflection	289
Circuit 1	112
Circuit 2	110
Tree	47
Box-Pointer	122
Total	1522

Table 1: Number of sketches for various types of drawings.

Class	Drawing	Class	Drawing
Circle		Upside Down Triangle	
Triangle		Plus	
Square		Resistor	
Diamond		Battery	
Star Bullet		Parallelogram Right	
Number	1 , 15	Parallelogram Left	
Arrow Right		Trapezoid Up	
Arrow Down		Trapezoid Down	
Double Box		Cross	
Star		Minus	

Table 2: Target symbol classes used in our experiments.

3. Data set

In this work, we make a conscious effort to use realistic sketches collected under naturalistic settings and used the Q&A data set [35]. This data set contains a total of 1522 sketches produced in response to 7 basic maths, physics, and computer science questions collected from groups of high-school and college students (number of drawings per question type and example sketches are given in Tables 1, and Figure 1).

The data set has five key properties making it realistic, hence amplifying the creditability and validity of our results. First, the sketches were collected *in the wild* at schools, from students who were asked to answer questions accordant with their grade level in their natural environments. Hence, they are more representative compared to drawings collected through controlled laboratory settings or mechanical Turk setups [1, 3, 4]. Second, sketches in this data set were collected using tablets equipped with proper styluses, and not painted with a mouse as in the case of mechanical Turk setups. Third, the students were given written questions and asked to produce freehand drawings describing their answer, resulting in substantial variation in sketches very much in the spirit of the work by Adler and Davis [36]. Fourth, the data set contains only one sketch per question from each participant, hence avoids duplicates. Fifth, the data set contains full sketches and not individual symbols. Furthermore, since there was no restriction on the set of symbols used in the drawings, they contain substantial amount of handwriting and outlier symbols. The self-learning experiments in this paper were carried out for symbol classes shown in Table 2.

Our system makes three assumptions on the data set. (1) First assumption is interspersing. As we utilize both temporal and spatial information of sketches, interspersing is not allowed (i.e. starting a symbol before finishing another). (2) We assume that two instances are members of the same symbol only if they share the same, or very similar, orientation. This is essential for the data set we use as the same shapes with different orientation have different semantics. (3) The last assumption we make on data is over-tracing. We assume that there are no over-traced symbols in the data set. Although there are studies handling over-tracing, our data set is a solid example showing that people do not over-trace much in certain kinds of sketches.

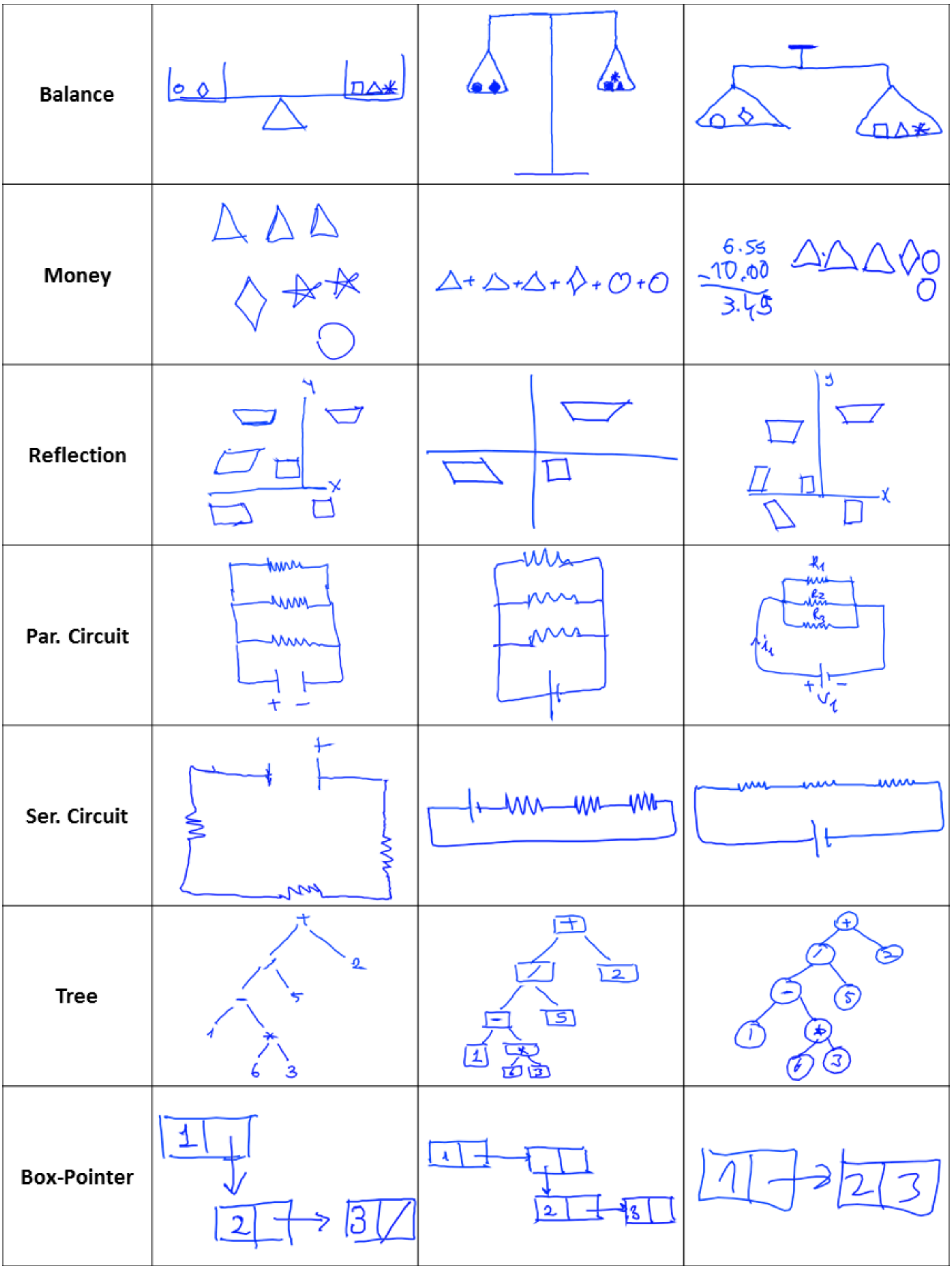


Figure 1: Example sketches from the data set used in our experiments.

4. Experimental Setup

Our main contribution is a context-based self learning algorithm for learning from few examples. We compare this algorithm to a host of other alternatives, including variants of nearest neighbor self learners combined with the state of the art feature representations, and an approach based on artificial instance generation. In order to assess the relative merits of these approaches, we run several experiments initialized with the same initial conditions. All these experiments are carried out with the execution pipeline shown in Figure 2. The pipeline consists of four stages: 1) Candidate Extraction, 2) Conservative Rejection, 3) Self-learning 4) Performance Measurement.

Technically, the job of the self learner is to train a binary classifier for a target symbol (one from Table 2) using unlabeled full sketches. As a first step, we extract symbol candidates from full sketches for use in the subsequent self-learning stages. Stage two discards a subset of the unlabeled symbol candidates that we can confidently declare as not representing the symbol of interest. In the third stage, we perform self-learning, and in the fourth stage we measure the performance of the classifier obtained through self learning. Now we describe each stage in detail.

4.1. Candidate Extraction

The input to the self-learning pipeline is 1-3 instances of the target symbol class, and a group of unlabeled sketches. The goal is to find further instances of the target symbol in the unlabeled sketches, and train a binary classifier on all the instances.

However, identifying further instances of the target symbol in sketches is hard, primarily because the sketch is simply a collection of strokes, and conceivably any subset of the strokes could be representing an instance of the target symbol. It is not known a priori which subsets of the strokes represent meaningful objects, hence technically each and every possible grouping of the strokes is a potential instance of the target class.

The purpose of candidate extraction is to build a list of symbol candidates by extracting groups of ink from the unlabeled sketches that can conceivably be an instance of the target class. We perform these groupings over straight line segments (primitives) extracted from the unlabeled sketch using the Douglas-Peucker algorithm [37] as illustrated in Figure 3. This is performed in the combination generation step shown in Fig. 2. Groupings created over primitives are more flexible compared to those created over strokes, and allow us to support multi-object strokes and multi-stroke objects as defined in [7].

The complexity of this method depends on the number of sketches in the dataset, complexity of the fragmentation method (Douglas-Peucker algorithm in our case), and min-max values of the number of primitives per combination to be generated, which are set by the user. The worst case complexity of Douglas-Peucker algorithm is $O(n^2)$, where n is the number of sketch points per stroke. The complexity of generating combinations for is $O(p \times (l - s + 1))$, where p is the average number of primitives produced by the Douglas-Peucker algorithm, l is the number of primitives in the longest combination, and s is the number of primitives in the shortest combination. As a result, the complexity of this method is the combination of these steps, which is : $O(D \times K \times (n^2 + p \times (l - s + 1)))$, where D is the number of sketches in the dataset, and K is the average number of strokes per sketch.

Enumeration of primitives to obtain symbol candidates is costly, and has exponential time complexity in the number of primitives. To keep this step tractable, we limit the number of primitives in each group to 2-15, and assume the primitives to be temporally adjacent.

Once all the symbol candidates are extracted, we compute IDM features [34] for them using the feature extraction settings recommended by Sezgin and Tumen [14] in the feature extraction step in Fig. 2. IDM feature extraction method transforms sketches into five feature images. Four of those feature images contain orientation and one of those contains stroke endpoint information of a sketch. After extraction of the feature images, IDM applies smoothing and down-sampling followed by a concatenation operation to form the feature vector. Next, a few (1-3) sketches are randomly selected and only the positive instances are labeled to mimic user input. Feature vectors representing the symbol candidate are then passed to the conservative rejection step to discard those candidates that are unlikely to be instances of the symbol of interest.

4.2. Conservative Rejection

In a typical sketch, candidate extraction yields thousands of primitive groupings, only a few of which will be of the target symbol class. This creates scalability concerns for the subsequent steps. Hence, we discard any candidates that we can confidently declare belonging to the negative class (i.e., class other than the symbol of interest).

We filter out some of the negative instances by training a simple but fast classifier following a strategy inspired by the work of Viola et al. [38]. Instances that get classified into the negative class are filtered out of the data set. The filtered out instances are labeled negative in the final evaluation step. The proportion of the instances to be filtered out is determined by a free parameter. In our experiments, we set this parameter to 25%, meaning the quarter of the instances that are predicted negative will be filtered out.

A good value for this parameter depends on the dataset characteristics as this parameter controls the trade-off between the speed of the later processes and the false omission rate. In our dataset, negative instances dominate over the positives in numbers thus setting the parameter value to 25% has a positive impact on speed of the later processes while keeping the false omissions rate very low as presented in the results section.

In our experiments, we train a classifier using a linear kernel SVM with labeled instances (assuming that unlabeled instances in annotated sketches belong to negative class) for conservative rejection [39]. The linear kernel SVM is trained with a high C hyper-parameter value to prevent training errors as it is done in one-class-classification frameworks. Next, we predict the classes of unlabeled instances using this classifier. The instances that are predicted negative with highest confidences are

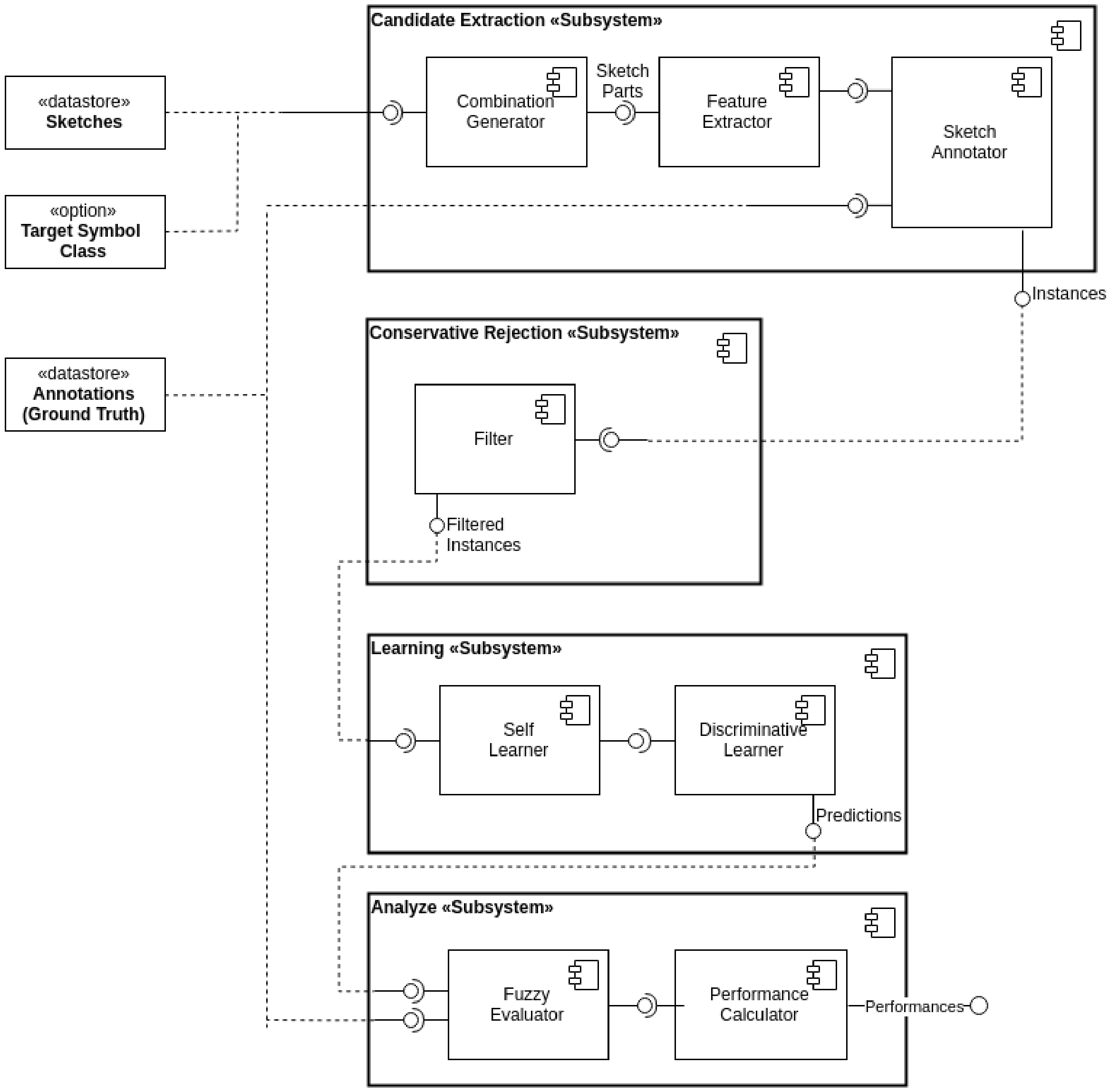


Figure 2: Component Diagram of the Framework

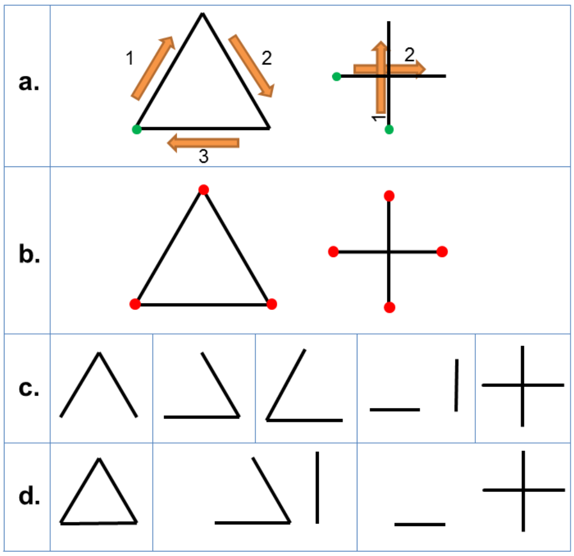


Figure 3: Illustration of Sketch Fragmentation and Combination: a) Original sketch with two symbols (△ and +) with arrows describing the drawing order. b) Primitives (lines) extracted through fragmentation. c) Combinations containing two primitives, d) Combinations containing three primitives

filtered out. We choose to use linear kernel for speed and its superior generalization ability for small data sets.

4.3. *Self-Learning*

Self-learning tries to expand the initial set of 1-3 positive examples with new ones selected from the candidates that pass the conservative rejection step. We perform self-learning using our context-based self-learning method, and a host of other alternatives.

In practice self-learning has usually been used with larger seed sets (larger than 10) [40]. This is primarily due to the difficulty of generalizing with few examples. Hence, in our experiments, we also include alternative methods capable of working with very few examples to serve as baselines. In particular, we include variants of nearest neighbor self learners combined with state of the art feature representations (i.e. Instance-wise Nearest Neighbor (IW NN) (section 5.1), Mean of Distanced Nearest Neighbor (MoD NN) (section 5.2)), and an approach based on artificial instance generation (AIG). The details of these self-learners are described in Section 5.

To keep our experiments tractable, we cut off self-learning after a total of 15 instances have been labeled.

4.4. *Discriminative Learning*

In this stage, we train final binary classifiers using the self-labeled instances. For this purpose, we use nearest neighbor classifiers and linear SVMs with bagging. Bagging is a well known approach in machine learning literature to overcome overfitting and high variance. This approach selects sub samples from the data set randomly with repetition, trains a model with each set, and predicts the class of an instance by combining the predictions of each individual models.

While nearest neighbor classifier is a very simple classifier, it is extremely hard to beat in sparse data sets (fewer than 5 examples) [41, 42]. Linear SVM with bagging is a meta-learning method [43]. Here, we generate 50 random subsets of the data set and train models for each of those subsets. The subsets are generated by randomly selecting half of the instances from the data set. For prediction, we perform majority voting.

The classifiers obtained in this step are used to make the final predictions on all the unlabeled instances to measure classifier performance.

4.5. *Performance Measurement*

We follow the standard confusion matrix approach in our study to measure system performance. However, since the granularity of labeling is at the level of primitives, we adopt a fuzzy evaluation scheme that addresses issues that can arise from over-fragmentation.

Over-fragmented sketch data sets include instances that are very similar to positive sketch objects yet labeled as negative in ground truth. An example is presented in 4. In this example, there are four instances each is a subset of the positive labeled sketch object. The instance that has 90% match is also a clear example of a circle, however as it is a subset of the annotated sketch object, it is not labeled as positive in the ground truth annotation. In order to address this issue, we calculate the confusion matrix using fuzzy evaluator.

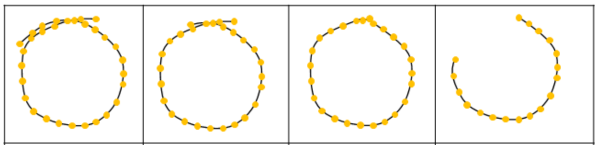
Fuzzy evaluator is a simple matching algorithm that labels a prediction as true positive if its overlap with the annotated sketch is over a certain threshold. We set this free parameter to 90%. In cases where there are multiple positive predictions for the same object that exceed the threshold, only the one with the highest amount of overlap is counted as true positive and other predictions are counted as false positives.

Although we report fuzzy evaluation results, we conducted additional experiments to compare predictions directly to the ground truth for the top methods (context based self-learning and nearest neighbor). Direct comparison experiments produced results very similar to fuzzy evaluation (less than 0.01% difference in performance). The direct and fuzzy evaluations produce very close results, because fuzzy evaluation only kicks in if an object has more than ten primitives. In our dataset, most objects have fewer than ten primitives, thus fuzzy evaluation performance closely follows direct comparison.

The set of initially labeled examples affect the performance of the system. For example, while some sets may have diverse instances, others may consist of highly similar and redundant examples. In this case, we would expect the diverse set to perform better than the similar set as it carries more information about the class. In order to account for such variations in our evaluation, we performed each experiment with randomized sets of labeled sketches for a total of 10 repetitions.

5. Self-Learners

We propose two self-learners that combine state of the art feature representations with nearest neighbor classifiers. A third



Annotated Sketch Object	Partial Sketch Object	Partial Sketch Object	Partial Sketch Object
30 Sketch Points	27 Sketch Points	25 Sketch Points	18 Sketch Points
100% Match	90% Match	85% Match	60% Match

Figure 4: Conceptual example of over-fragmentation.

self-learner is inspired by the work of Miller et at.[25], and is based on artificial instance generation. The fourth method is our novel context based self-learning method.

5.1. *Instance-wise Nearest Neighbor (IW NN)*

Here we extend the positive instance set by labeling the unlabeled instances closest to the existing positive labeled instances where the distance is defined as the Euclidean distance between feature vectors. Each positive instance contributes equal number of additional positive instances.

5.2. *Mean of Distances Nearest Neighbor (MoD NN)*

This method extends the positive instance set by labeling instances with the lowest mean distance to all of the positive labeled instances where distance is defined as the Euclidean distance between feature vectors. This method effectively favors points closest to the mean of the existing data points. It has the potential advantage of finding more diverse instances compared to the instance-wise nearest neighbor method.

5.3. *Artificial Instance Generation (AIG)*

Formally, artificial instance generation is not a self-learning method. While self-learning methods extend the labeled set via labeling unlabeled instances, artificial instance generation method extends the labeled set by generating novel instances from existing ones.

We generate artificial instances by applying linear geometric transformations on positive labeled instances. We limit the set of transformations to rotation and shearing. We generate 10 novel instances for each positive labeled instance, randomly apply transformations using a rotation parameter in the range $[-\pi/12, \pi/12]$, and a shearing parameter in the range $[0, 0.25]$. This approach is inspired by the work of Miller et al.[25], which tries to generalize from a single example by combining it with a probability density defined over the parameters of a family of affine transforms.

5.4. *Context Based self-learning*

It is known that sketches contain rich spatial patterns. For example, elements in charts [44], nodes and connectors in a binary trees [22], components in a circuit diagram have prototypical spatial co-occurrence patterns [7, 13]. The context-based self-learning algorithm that we propose is based on this observation.

The key insight is to favor unlabeled symbol candidates that not only have the **appearance** of the class of interest, but also appear in **contexts** that are typically observed for objects of interest. Hence we calculate appearance and context scores for symbol candidates.

5.4.1. *Calculation of the appearance score*

The appearance score measures the visual similarity between an unlabeled instance and the positive labeled instance that is closest to it. The score is calculated based on the feature-space distance between the unlabeled and the labeled instances (i.e. Euclidean distance between the feature vectors of labeled and unlabeled instances). After calculations, we normalize the appearance scores to the range [0, 1] using Platt scaling [45].

5.4.2. *Calculation of the context score*

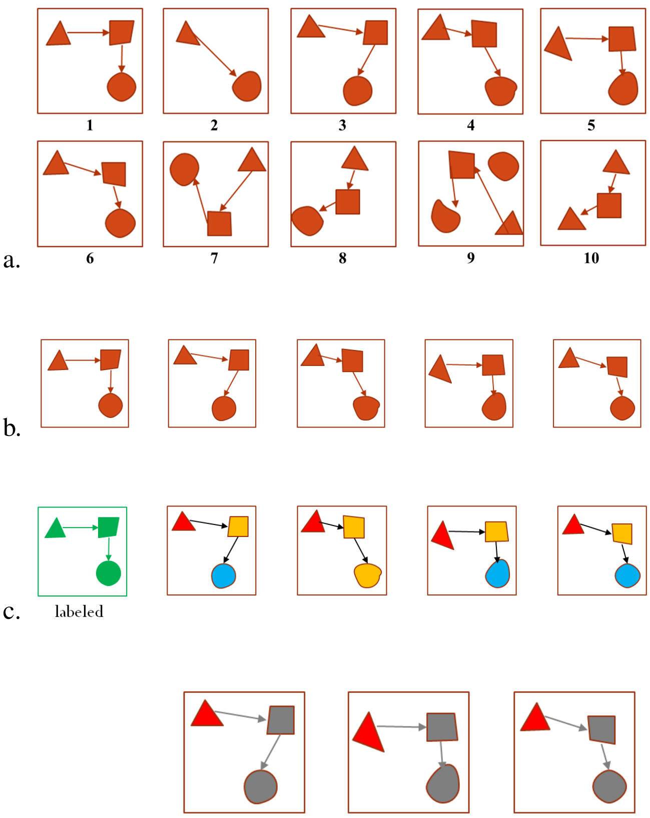
The context score measures the agreement of the pairwise spatial relationships of candidate symbols and the already labeled examples. It is computed in five steps: clustering, prediction, score calculation, score scaling, and instance selection.

In the clustering step, we cluster sketches by their appearance into sufficiently large number of clusters (20 in our case) to achieve within cluster homogeneity using hierarchical clustering. Sketch appearances are encoded using IDM features (i.e. each whole sketch is represented by a set of IDM features). During clustering, Euclidean distance in feature space is used as similarity metric. In the prediction step, object instances in unlabeled sketches are located via nearest neighbor classifiers using the annotated instances.

To serve as a toy example, consider the set of unlabeled sketches in Fig. 5-a. The process starts by **clustering** all unlabeled sketches based on their appearance. This allows us to find sufficiently large clusters of sketches that are likely to share similar contexts (one such cluster shown in Fig. 5-b). Subsequent operations focus on these clusters, hence we save computational resources.

Next we bootstrap the self-learning process by obtaining labels for objects in one of these sketches. These are the few examples that we require from a user. We use these examples as nearest neighbor classifiers to **predict** labels on the unlabeled instances (Fig. 5-c). Note that these classifiers don't have to be very accurate. In particular, they can be allowed to have large false positive rates. An overwhelming portion of the false positives will not have the expected contextual relationships, hence they will not inhibit the subsequent steps. Appearance scores of the positively predicted instances are calculated at this step, which is inversely proportional to the distance between the instance and its labeled nearest neighbor. As an example, triangle in second sketch at Fig. 5-d has a low appearance score as its appearance is less similar to the appearance of the labeled triangle.

Next, **matching scores** are calculated by comparing the spatial relationships between pairs of predicted symbols in the labeled and unlabeled sketches. Continuing with the example in



d.

Context	0.5	0.9	0.8
Appearance	0.8	0.4	0.8
Final score	0.68	0.6	0.8

Figure 5: A toy example illustrating the calculation of the context score: a) The unlabeled input sketches, b) A cluster consisting of five similar sketches obtained through hierarchical agglomerative clustering (sketches 1, 3, 4, 5, 6), c) Objects in one of the sketches are labeled (green sketch), and classes predicted on the others indicated in color. Note the prediction error on the third sketch from left. d) The appearance, context and final scores computed.

Fig. 5-c, this amounts to comparing the pairwise spatial relationships between the triangle, square and circle in the labeled sketch to the spatial relationships of the same symbols predicted in the unlabeled sketches. In our examples, the triangle in the second sketch in Fig. 5-d to has a high context score as its spatial relationship to other objects is similar to what we observe in the labeled sketch. Spatial relationship is defined by the length and orientations of imaginary vectors originating from the source object and extending to the target object (for example, an imaginary vector in Fig. 5-c from the triangle to the square). The spatial relationship in this example would be captured by the length and orientation of the imaginary vector. The context scores will be higher for pairs that match the spatial relationship in the labeled sketch (Fig. 5-d). Pseudo-code for calculation of context score for a single sketch object is presented in algorithm 1. These scores are also normalized using Platt scaling. After scaling scores to the range $[0, 1]$, all the scores are subtracted from 1 to represent similarity instead of dissimilarity.

The complexity of calculating context score for a single object is $O(o + G \times e_p)$, where o is the number of sketch objects in the sketch, G is the number of labeled sketches in the dataset, and e_p is the size of patterns in the labeled sketches (proportional to number of sketch objects in the labeled sketches).

After the context score S_C and appearance score S_A are calculated for all candidate sketch objects, the scores are combined to obtain a final score S_{final} (Fig. 5-d). We combine scores using a linear combination where an α parameter controls the relative dominance of the context and appearance. In particular: $S_{final} = S_C * \alpha + S_A * (1 - \alpha)$. We **select** unlabeled instance candidates with the highest scores, and add them to our seed list of labeled examples, hence achieve self-learning. Examples of failure (false positive) and success (true positive) of this method are presented in Figure 6.

This method is able to work with different sketches ranging from having a single, to tens of different sketch objects. For sketches that embodies only single sketch object, the context score will be calculated based on the location of the object in the sketch. In this case if a candidate sketch object is located in a different place compared to labeled object, it will have a lower context score. When there exist multiple objects in a sketch, the method will compare the placement patterns of the objects with the placement patterns in the labeled sketches. The context score yielded by comparing placement patterns formed with multiple objects will be more informative as the placement patterns with more objects supply more information. As a result, the datasets with sketches that embodies multiple objects will benefit the context method better compared to the ones that embodies single objects. However this does not indicate that as the number of objects per sketch increases, the benefit gained from context method will increase.

6. Results

In the previous sections, we introduced a conservative rejection scheme to improve scalability, bagging to address data

Algorithm 1 Context Score Calculation Algorithm

procedure CONTEXT SCORE CALCULATION(target object t, labeled s placement patterns G, sketch s)
  Let placement pattern p for t within s be PlacementPattern(t, s)
  Let $minscore$ be +Inf
  for each placement pattern g in G **do**
    if PatternDissimilarity(g, p) is less than $minscore$ **then**
      $minscore$ = PatternDissimilarity(g, p)
    end if
  end for
  return $minscore$
end procedure
procedure PLACEMENTPATTERN(target object t, sketch s)
  Let h be the holder for placement information
  for each object o in sketch s **do**
    Let a be the angle between t and o
    Let d be the distance of the object centers between t and o in s
    Let c be the predicted class of o
    Add a, d, and c to h
  end for
  return h
end procedure
procedure PATTERNDISSIMILARITY(pattern p, pattern g)
  Let h be the holder for entry differences
  for each entry e_p in pattern p **do**
    Let entry e_g be the entry in g that is sharing same object class with entry e_p
    Add angle and distance differences between e_g and e_p to h

  end for
  Let the average angle difference be a_a
  Let the average distance difference be a_d
  return $a_a + a_d$
end procedure

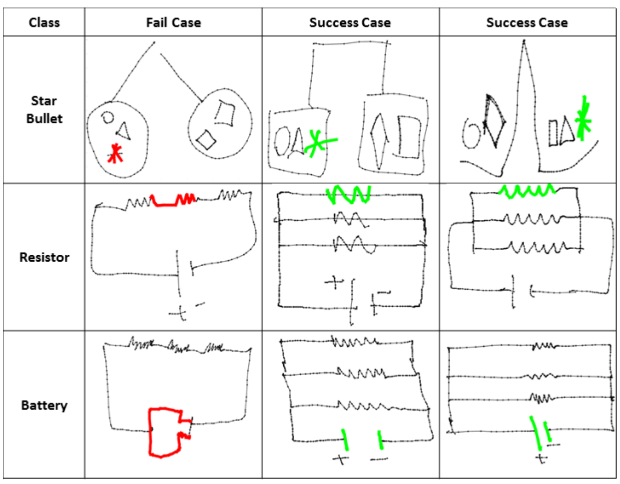


Figure 6: Failure and success examples for the context-based self-learner.

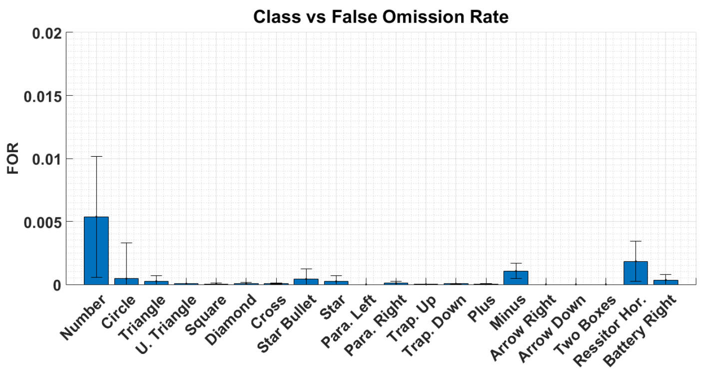


Figure 7: Performance of conservative rejection measured through the false omission rate. (FOR = FN / (FN+TN))

imbalance, and four self-learning methods to learn from few examples. Here we report the performance of these techniques.

6.1. Conservative Rejection Performance

We used conservative rejection to remove irrelevant candidate objects. However, since instances of the symbol of interest may already be too few, we would like the avoid discarding them. Hence, we need to assess the number of relevant examples that have been inadvertently removed in this step. This is measured through the false omission rate.

The false omission rate gives the number of positive candidates that have been inadvertently discarded, normalized by the number of true negatives. As seen in Fig. 7, even if we filter out a large portion of the instances, as we do in our case, the false omission rate is quite low.

6.2. Effect of Bagging

In order to assess the utility of bagging, we compare the performance of the self-learning methods that have bagging variants. We performed a 3-factor repeated measures ANOVA to study the effects of three factors on self-learning: **1)** the initial number of positive instances (1, 2 or 3), **2)** the presence or absence of bagging method (using bagging vs. using a single classifier), and **3)** the self-learning method (MoD NN, IW NN, AIG). The profile plots in Fig. 8 clearly demonstrate the advantage of performing bagging. Green-house-Geisser corrected values computed following the Mauchly's test of sphericity did not indicate three-factor interactions ($p < 0.05$). The analysis indicated statistically significant interactions for the first and the third, as well as the second and third factors ($p < 0.05$). The presence of an interaction between factors implies that the setting of one parameter has an effect on the way changing the setting of the other parameter will affect the system performance. These interactions are *quantitative interactions*, hence we look at the main effects. The main effects show that all three factors have a statistically significant effect on performance. Most importantly, bagging results in significant improvements in performance. This result serves as the first demonstration of the utility of bagging for sketch recognition with few examples in the literature.

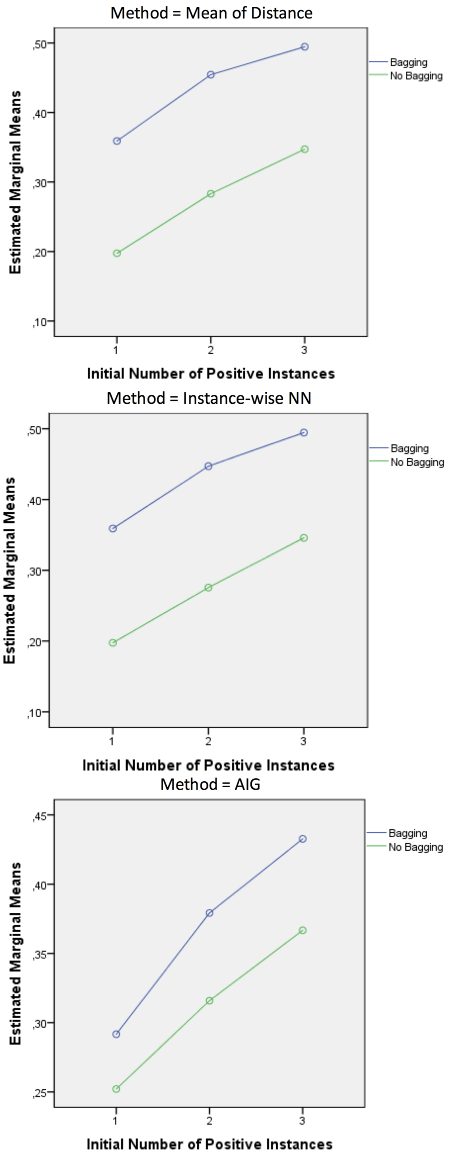


Figure 8: Performance (F-measure) of bagging for the three methods with and without bagging. For all three methods, bagging results in a statistically significant improvement in performance.

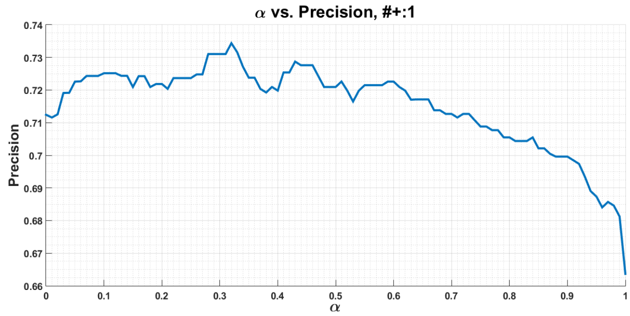


Figure 9: Precision for combinations of appearance and context scores shown as a function of α

6.3. *Self-Learning Performance*

We assess the performance of the self-learning methods in two fronts. First, we assess the precision of each method in finding new examples that are indeed of the desired symbol class. Next, we measure if the new examples are successfully converted into higher recognition accuracies.

The precision of the context-based self learner depends on the α parameter. Since this parameter controls the mixture of context and appearance scores, we expect it to have a peak towards the middle. Figure 9 agrees with this expectation. A value of $\alpha = 0$ favors appearance, and $\alpha = 1$ favors context. However, note that since the context score is only calculated for sketches which already have symbols with plausible appearances, the precision does not drop too low on the far end of the graph as one might expect.

We report the classification performance of the final classifiers using F-measure. F-measure serves as a reliable metric for imbalanced data sets, hence its use is appropriate. Fig. 10 presents the overall performance of all methods for varying number of initial examples.

In order to assess statistical significance, we performed a multi-factor repeated-measures ANOVA test. We took the number of positive instances (NoPI), and the self-learning method as the two factors. Since we have analyzed the effects of bagging separately, here we treat each of the 8 methods independently. The profile plots from our analysis Fig. 10 shows consistent ordering of the performances for the methods under question. A pronounced superiority of the context-based self-learning is also evident. The Greenhouse-Geisser corrected values computed following the Mauchly's test of sphericity show statistically significant interactions (Table 3, and Fig. 10). Hence, we tested for simple main effects of the self-learning methods for 1, 2 and 3 initial positive instances. The results show that the context-based self-learner dominates over the other methods for all choices of the number of initial examples ($p < 0.05$). The 95% confidence intervals are shown in Fig. 11.

Note that since the profile plots from our analysis (Fig. 10) shows consistent ordering of *quantitative interactions*, consulting the main effect statistics is safe. The statistics indicate significant main effects ($p < 0.05$) for both factors under investigation (NoPI & Method) shown in Table 3. Results of the pairwise comparisons for the number of positive instances and the 95%

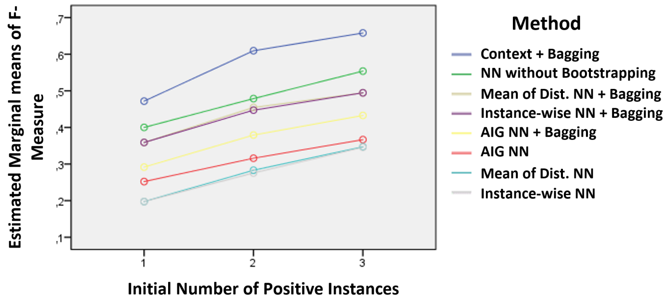


Figure 10: Estimated marginal means of the performances measured through F-Measure values. A multi-factor repeated measures ANOVA test shows that our context-based self learner with linear SVM bagging performs significantly better. In the legend, the label before the + symbol indicates the self-learning method, the label after the + indicates the classifier that was used. See Table 3, and Table 4 for the quantitative details.

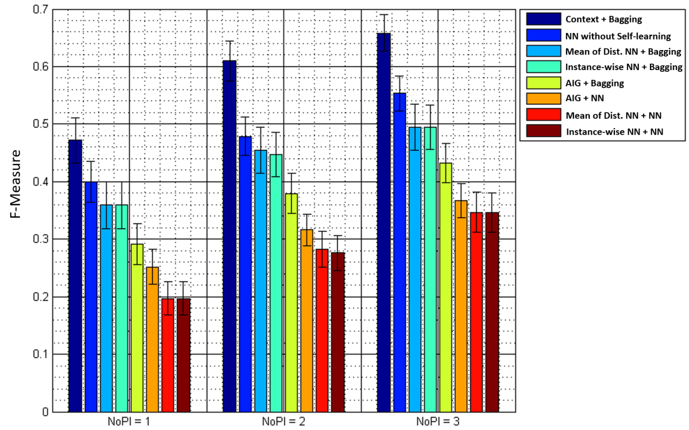


Figure 11: 95% confidence intervals (error-bars) and estimated marginal means (filled bars) for the methods. The context-based learner dominates over the others.

confidence intervals can be found in Table 4. This main effect is also evident in Fig. 10.

As can be seen from the results, our novel context based self-learning method with linear SVM bagging is superior compared to other methods. We attribute the superior performance of context based self-learning to its higher precision and diverse instance selection capability.

7. Discussion

The results presented above are surprising on many fronts. First, it is counter-intuitive to see that self-learning and artificial instance generation do not always yield better performance compared to a simple instance-based classifier.[1] This is the case for initial seed set sizes of 1, 2 and 3 for the instance-wise and mean-of-distance based nearest neighbor self-learners. Even though the precision of these methods improve with larger seed sizes (Fig. 10 shows steady improvement), the self-learned examples actually lead to inferior classifiers.

We believe this is due to lack of diversity of the newly added examples, and the adverse effects of mislabeled examples added to the seed set through self-learning. In order to verify the effects of diversity, we compared the diversity of the examples added through our context-based self-learner, which is the overall best achiever, and those added by the other self learners.

We assess diversity of a set of candidates by computing the minimum radius that encloses the candidates and the labeled examples from which the candidates were self-learned. We use linear kernel support vector data description to find the hypersphere [46]. Intuitively, a set of diverse instances will require a larger sphere for enclosure, while the less diverse ones will fit inside a small one. Figure 12 displays the relative diversity of the instances chosen by two underperforming self-learners to the ones chosen by the context-based self learner. Hence the y axis serves as an indicator of the difference in diversity. Those instances with positive value can be said to be more diverse than their respective counterparts learned through the use of context. As seen in this figure, the relative diversity of the candidates selected for labeling is mostly on the negative side for the underperforming methods. This is strong evidence that underpins the importance of diversity, and serves as a guide for further research in the direction of building better self-learners.

Another surprising result is the boost in performances obtained through combination of self-learning and bagging – first time such results are presented in the sketch recognition literature. Bagging in general appears to help even with few positively labeled examples. This has the potential to help with the data imbalance problems associated with large number of outliers in realistic sketches. Extrapolating these results, we can predict that classifiers with bagging will be a standard choice in simultaneous segmentation and recognition architectures where the classifiers are fed few instances of positive instances and many more examples of outliers and meaningless sketch fragments.

The problem that we are addressing is far more challenging than the traditional closed-set, many-examples setup adopted in the mainstream. Hence, comparing the accuracies directly is not appropriate. However, there is room for improvement. We see two future directions that can be taken to further improve classification accuracies. First, the labeling process can be organized more strategically. In our system, the sketch to be labeled by the user is selected randomly. There is evidence from the Active Learning literature that not all examples are equally useful, and directing the annotation effort to the more informative examples has the potential to yield better recognizers. Active learning gives a set of rules and guidelines on how these more informative examples can be found. We believe active learning with an emphasis on using few examples is a promising future direction to take. Second, we utilize context placement information for self-learning. However, we believe using context information for prediction can further boost system performance in general.

[1] See Fig. 10 where N.N. without self-learning achieves significantly better results than mean of distance N.N., and methods that utilize AIG.

Source	Type III SS	df	Mean Square	F	Sig.	Partial Eta Sq.
NoPI	13,811	1,794	7,699	92,517	,000	,368
Error(NoPI)	23,736	285,234	,083			
Method	38,833	3,023	12,848	189,069	,000	,543
Error(Method)	32,657	480,586	,68			
NoPI * Method	,388	8,382	,046	3,463	,000	,021
Error(NoPI*Method)	17,793	1332,743	,013			

Table 3: Tests of within-subjects effects.

					95% Confidence Interval	
(I) NoPI	(J) NoPI	Mean Difference (I-J)	Std. Error	Sig.	Low. Bound	Up. Bound
1	2	-0.089	0.012	000	-0.118	-0.061
	3	-0.146	0.012	000	-0.174	-0.118
2	1	0.089	0.012	000	0.061	0.118
	3	-0.056	0.009	000	-0.077	-0.035
3	1	0.146	0.012	000	0.118	0.174
	2	0.056	0.009	000	0.035	0.077

Table 4: Pairwise comparisons for the effect of the number or positive instances (NoPI) used for initiating self-learning. All differences are statistically significant ($p < 0.05$).

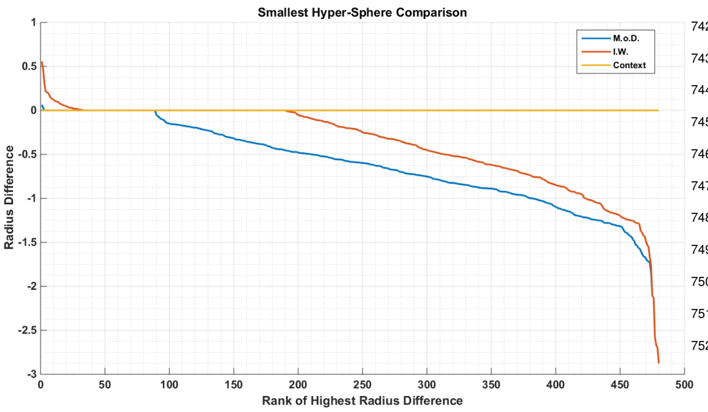


Figure 12: Smallest Hyper-Spheres Radius Differences of self-learning Methods

8. Contributions and Future Work

We presented a novel context-based self learning method that successfully learns from few examples. We demonstrated the utility of this approach through its ability to accurately select diverse examples for training sketch recognizers. Successful incorporation of bagging and conservative rejection serve as two additional contributions.

We believe that our work is open for further improvements. All of the subsystems we presented can be further studied independently. Apart from the methods we examined in this work, there are many methods proposed in literature which can be used to achieve better results in our system. We see two major directions for future work in sketch recognition with few instances.

One direction to study is active learning for sketch recognition with few instances. In our current system, the user annotates sketches selected in random. However, it is possible to increase performance rates both for self-learning and classification if the sketches to be annotated are chosen carefully as opposed to randomly. The work of Yanik et al. [19] on active learning for sketch recognition can serve as a guideline for such future work.

In this work, we utilize context information only for self-learning. However, context can be used to improve classification rates as well. There is already a body of work using context for recognition, and the insights gained from this work can lead to novel ways of using context, and adaptive models which know when and where to use context information.

Acknowledgments

The authors gratefully acknowledge the support and funding of TUBITAK under grant number 113E059.

References

[1] R. Arandjelovic, T. M. Sezgin, Sketch recognition by fusion of temporal and image-based features, Pattern Recognition 44 (6) (2011) 1225–1234.

[2] H. Hse, A. R. Newton, Sketched symbol recognition using zernike moments, in: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, Vol. 1, IEEE, 2004, pp. 367–370.

[3] M. Eitz, J. Hays, M. Alexa, How do humans sketch objects?, ACM Trans. Graph. 31 (4) (2012) 44–1.

[4] R. Niels, D. Willems, L. Vuurpijl, The nicicon database of handwritten icons for crisis management, Nijmegen Institute for Cognition and Information Radboud University Nijmegen, Nijmegen, The Netherlands 2.

[5] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, T. Hospedales, Sketch-a-net that beats humans, in: British Machine Vision Conference (BMVC), 2015.

[6] C. Alvarado, M. Lazzareschi, Properties of real-world digital logic diagrams, in: Pen-Based Learning Technologies, 2007. PLT 2007. First International Workshop on, IEEE, 2007, pp. 1–6.

[7] T. M. Sezgin, R. Davis, Sketch recognition in interspersed drawings using time-based graphical models, Computers & Graphics 32 (5) (2008) 500–510.

[8] J. V. Mahoney, M. P. Fromherz, Three main concerns in sketch recognition and an approach to addressing them, in: AAAI Spring Symposium on Sketch Understanding, 2002, pp. 105–112.

[9] T. M. Sezgin, R. Davis, Generating domain specific sketch recognizers from object descriptions, in: Student Oxygen Workshop, Vol. 17, 2002.

[10] O. Veselova, R. Davis, Perceptually based learning of shape descriptions for sketch recognition, in: 33. International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2006, Boston, Massachusetts, USA, July 30 - August 3, 2006, Courses, 2006, p. 28.

[11] T. A. Hammond, Ladder: A perceptually-based language to simplify sketch recognition user interface development, Ph.D. thesis, Massachusetts Institute of Technology (2007).

[12] T. Y. Ouyang, R. Davis, Recognition of hand drawn chemical diagrams, in: AAAI, Vol. 7, 2007, pp. 846–851.

[13] M. Oltmans, Envisioning sketch recognition: a local feature based approach to recognizing informal sketches, Ph.D. thesis, Massachusetts Institute of Technology (2007).

[14] R. S. Tumen, M. E. Acer, T. M. Sezgin, Feature extraction and classifier combination for image-based sketch recognition, in: Proc. of the 7^{th} Sketch-Based Interfaces and Modeling Symp.

[15] B. Plimmer, R. Blagojevic, S. H.-H. Chang, P. Schmieder, J. S. Zhen, Rata: codeless generation of gesture recognizers, in: Proceedings of the 26th Annual BCS Interaction Specialist Group Conference on People and Computers, British Computer Society, 2012, pp. 137–146.

[16] B. Plimmer, I. Freeman, A toolkit approach to sketched diagram recognition, in: Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 1, British Computer Society, 2007, pp. 205–213.

[17] C. Alvarado, R. Davis, Preserving the freedom of paper in a computer-based sketch tool, in: Proceedings of HCI International, Vol. 2001, 2001.

[18] J. S. Zhen, R. Blagojevic, B. Plimmer, Automated labeling of ink stroke data, in: Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling, Eurographics Association, 2012, pp. 67–75.

[19] E. Yanık, T. M. Sezgin, Active learning for sketch recognition, Computers & Graphics 52 (2015) 93–105.

[20] B. Romera-Paredes, P. Torr, An embarrassingly simple approach to zero-shot learning, in: Proceedings of The 32nd International Conference on Machine Learning, 2015, pp. 2152–2161.

[21] C. Tirkaz, J. Eisenstein, T. M. Sezgin, B. Yanikoglu, Identifying visual attributes for object recognition from text and taxonomy, Computer Vision and Image Understanding 137 (2015) 12–23.

[22] C. Alvarado, R. Davis, Sketchread: a multi-domain sketch recognition engine, in: Proceedings of the 17th annual ACM symposium on User interface software and technology, ACM, 2004, pp. 23–32.

[23] M. Shilman, H. Pasula, S. Russell, R. Newton, Statistical visual language models for ink parsing, in: AAAI spring symposium on sketch understanding, 2002, pp. 126–132.

[24] D. Rubine, Specifying gestures by example, SIGGRAPH Comput. Graph. 25 (4) (1991) 329–337.

[25] E. G. Miller, N. E. Matsakis, P. A. Viola, Learning from one example through shared densities on transforms, in: Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, Vol. 1, IEEE, 2000, pp. 464–471.

[26] Y. Song, R. Davis, K. Ma, D. L. Penny, Balancing appearance and context in sketch interpretation, in: 25th International Joint Conference on Artificial Intelligence, 2016.

[27] M. Szummer, Learning diagram parts with hidden random fields, in: Eighth International Conference on Document Analysis and Recognition (ICDAR'05), IEEE, 2005, pp. 1188–1193.

[28] T. Ouyang, R. Davis, Learning from neighboring strokes: Combining appearance and context for multi-domain sketch recognition, in: Advances in Neural Information Processing Systems, 2009, pp. 1401–1409.

[29] J. O. Wobbrock, A. D. Wilson, Y. Li, Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes, in: Proceedings of the 20th annual ACM symposium on User interface software and technology, ACM, 2007, pp. 159–168.

[30] L. Anthony, J. O. Wobbrock, $ n-protractor: a fast and accurate multistroke recognizer, in: Proceedings of Graphics Interface 2012, Canadian Information Processing Society, 2012, pp. 117–120.

[31] L. Anthony, J. O. Wobbrock, A lightweight multistroke recognizer for user interface prototypes, in: Proceedings of Graphics Interface 2010, Canadian Information Processing Society, 2010, pp. 245–252.

[32] E. M. Taranta II, J. J. LaViola Jr, Penny pincher: a blazing fast, highly accurate $-family recognizer, in: Proceedings of the 41st Graphics Interface Conference, Canadian Inf. Proc. Soc., 2015, pp. 195–202.

[33] R.-D. Vatavu, L. Anthony, J. O. Wobbrock, Gestures as point clouds: a $ p recognizer for user interface prototypes, in: Proc. of the 14th ACM Int. Conf. on Multimodal Interaction, ACM, 2012, pp. 273–280.

[34] T. Y. Ouyang, R. Davis, A visual approach to sketched symbol recognition.

[35] S. Cakmak, Clustering Free-Form Sketch Scenes through Perceptual Similarity, Master's thesis, Koc University, Turkey (2016).

[36] A. Adler, J. Eisenstein, M. Oltmans, L. Guttentag, R. Davis, Building the design studio of the future (2004) 1–7.

[37] D. H. Douglas, T. K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartographica: The International Journal for Geographic Information and Geovisualization 10 (2) (1973) 112–122.

[38] P. Viola, M. J. Jones, D. Snow, Detecting pedestrians using patterns of motion and appearance, International Journal of Computer Vision 63 (2) (2005) 153–161.

[39] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

[40] X. Zhu, Semi-supervised learning literature survey, Tech. rep., Computer Sciences, University of Wisconsin-Madison (2005).

[41] D. M. J. Tax, One-class classification, Ph.D. thesis, TU Delft, Delft University of Technology (2001).

[42] D. M. J. Tax, R. P. W. Duin, Data description in subspaces, in: Proceedings 15th International Conference on Pattern Recognition. ICPR-2000, Vol. 2, IEEE, 2000, pp. 672–675.

[43] M. Skurichina, R. P. W. Duin, Stabilizing classifiers for very small sample sizes, in: Proceedings 13th International Conference on Pattern Recognition. ICPR-1996, Vol. 2, IEEE, 1996, pp. 891–896.

[44] G. Costagliola, M. De Rosa, V. Fuccella, Local context-based recognition of sketched diagrams, Journal of Visual Languages & Computing 25 (6) (2014) 955–962.

[45] J. Platt, et al., Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, Advances in large margin classifiers 10 (3) (1999) 61–74.

[46] D. M. J. Tax, R. P. W. Duin, Support vector data description, Machine learning 54 (1) (2004) 45–66.