Computers & Graphics 35 (2011) 955–966

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Neural network-based symbol recognition using a few labeled samples

Luoting Fu, Levent Burak Kara *

Carnegie Mellon University, Mechanical Engineering Department, 5000 Forbes Ave, Pittsburgh, PA 15213, USA

ARTICLE INFO

Article history:
Received 20 December 2010
Received in revised form
15 June 2011
Accepted 12 July 2011
Available online 23 July 2011

Keywords:
Sketch recognition
Symbol recognition
Deep Belief Network
Unsupervised training
Neural nets
Synthetic training samples

ABSTRACT

The recognition of pen-based visual patterns such as sketched symbols is amenable to supervised machine learning models such as neural networks. However, a sizable, labeled training corpus is often required to learn the high variations of freehand sketches. To circumvent the costs associated with creating a large training corpus, improve the recognition accuracy with only a limited amount of training samples and accelerate the development of sketch recognition system for novel sketch domains, we present a neural network training protocol that consists of three steps. First, a large pool of unlabeled, synthetic samples are generated from a small set of existing, labeled training samples. Then, a Deep Belief Network (DBN) is pre-trained with those synthetic, unlabeled samples. Finally, the pre-trained DBN is fine-tuned using the limited amount of labeled samples for classification. The training protocol is evaluated against supervised baseline approaches such as the nearest neighbor classifier and the neural network classifier. The benchmark data sets used are partitioned such that there are only a few labeled samples for training, yet a large number of labeled test cases featuring rich variations. Results suggest that our training protocol leads to a significant error reduction compared to the baseline approaches.

1. Introduction

Sketch understanding [1,2] aims to enable the computers to interpret man-made, freehand sketches and extract the intended information underlying the input strokes. Fig. 1 shows two exemplary sketches depicting two engineering systems and the corresponding engineering model. If successful, sketch understanding could provide a natural human-computer interface for scenarios in which physical, pen-and-paper sketches have been routinely used, such as the early ideation process or the classroom instruction. Moreover, sketch understanding could automate the mining, organization, search and critique of the information embedded in freehand sketches, potentially resulting in a myriad of intelligent agents, such as a web spider that crawls through the drawings in online textbooks and lecture notes to learn the design rules of electrical systems, an archiver that indexes brainstorming sketches for later retrieval and reuse, and a computer grader for the free-body diagrams that students draw in their statics homework.

One of the core problems in sketch understanding is to devise a symbol recognizer to compute a categorical label for each segment of the input sketch. Used in conjunction with a sketch parser that divides the input sketch into segments and possibly a post-processor that ensures the consistency of the recognition, an interpretation of the input sketch can be produced. Such problem decomposition is recurrent in recent sketch recognition systems [3–8]. For example, in [3], a Convolutional Neural Network recognizer is used with a sliding windows segmenter. The recognition output of the Convolutional Neural Network is post-processed to merge the overlapping and non-maximal labels. In [5], a heuristic-based segmenter is used in conjunction with a Gaussian Bayes classifier. Because of the particular choice of the segmentation heuristics for that domain, each segmentation corresponds to an isolated symbol in the sketch and post-processing is not required. In [7], the up and down motion of the pen-tip is utilized to segment the sketch into a number of strokes, and a Conditional Random Field model plays the dual roles of the recognizer and the post-processor to output a globally consistent interpretation of the input.

Neural network classifiers [9,10] are particularly appealing candidates for the recognition of sketched symbols. They feature a feed-forward classification algorithm capable of rapid classification of the input, and a supervised back-propagation training algorithm capable of the data-driven learning of highly complex decision boundaries between multiple categories. Neural network classifiers are known for the high accuracy achieved in various domains such as freehand-sketched symbols [3,11], handwritten digits [12] and human faces [13]. However, a large, correctly *labeled* training data set with rich, in-class variations is required to train a highly accurate neural network classifier. Such a requirement, arising from the rich stylistic variations of unconstrained user inputs, is reported in empirical [14–16] and theoretical

* Corresponding author.
E-mail addresses: luoting.fu@cmu.edu (L. Fu), lkara@cmu.edu (L.B. Kara).

0097-8493/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2011.07.001

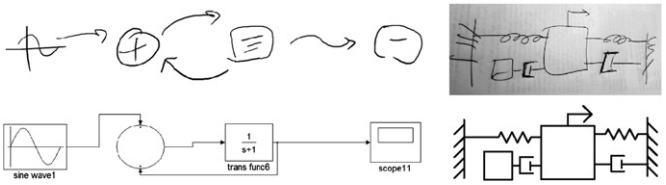

Fig. 1. Top row: two sketches drawn on a tablet PC and on a piece of paper. Bottom row: the control system and mechanical vibratory system models corresponding to the sketches above.

[17] studies. As a rule of thumb in visual classification in which complex, non-linear decision boundaries separate the pattern categories of interest, the performance of a classifier would degrade, if the size of the training data set is significantly below the dimension of the input patterns[1] [18].

Collecting a large number of training samples and manually labeling each instance, is time-consuming, costly, laborious and hence undesirable, especially when targeting a new sketch domain with new symbol definitions. To circumvent such difficulties and accelerate the training and deployment of neural network classifiers for such scenarios, we propose a novel training protocol that relies on only a handful of training samples[2] and achieves higher accuracy than purely supervised approaches.

The proposed training protocol consists of three steps, as illustrated in Fig. 2. Prior to applying our approach, a small, labeled data set of training samples, called the *seed* samples, is collected from the users. At the beginning of our approach, a large pool of synthetic, *unlabeled* samples are generated from the seed samples. Then, a Deep Belief Network (DBN, a neural network variant introduced in [19]) is trained using the unlabeled, synthetic samples. The learning objective of this step is to maximize the probability for the DBN to generate the training samples, such that it would discover the structural regularities underlying the input patterns and learn a non-linear, hierarchical representation of them. Finally, the pre-trained DBN from the previous step is used as a feature extractor and concatenated with an additional layer of decision units. Together they are fine-tuned as a deep, feed-forward neural network classifier using the labeled seed samples. The learning objective here is to minimize the classification errors on the labeled training set, so as to learn the decision boundaries between the pattern classes.

The proposed training protocol can be seen as supervised training preceded and enhanced by the synthesis of unlabeled training samples and the unsupervised pre-training. The incorporation of unsupervised pre-training is inspired by the recent progress on semi-supervised, transfer learning [20,21] and Deep Belief Network [19,22–25].

Our contribution is a protocol suitable for training neural network-based symbol recognizers in novel sketch domains where it is difficult to employ other existing training approaches. Specifically, in the scenario we target, the initial number of labeled samples is limited, no labeled sample synthesizer is available and no relevant domain with similar symbols exists for transfer learning. Our approach reduces the need for labeled training samples and in turn reduces the time or efforts needed to collect or label such samples from the users, thus enabling accelerated deployment of sketch understanding systems. The neural network-based recognizer can work with image-based, off-line sketches as well as trajectory-based, online sketches. We focus on the image-based, off-line sketches in this work for two reasons. First, such off-line sketches can be drawn on a broader range of digital or physical drawing surfaces and then captured using a variety of image or ink acquisition devices, whereas the acquisition of online sketches relies on tablet PCs or multi-touch devices. Second, online sketch recognition is confounded by the issues of stroke-level drawing order and stroke interspersions [26], whereas such issues do not exist in the image-based representation.

The rest of this paper is organized as follows: Section 2 reviews existing training approaches for neural network-based image classifiers, and shows that our approach is suitable for a scenario not covered by the existing approaches. Section 3 presents the step-by-step details of the proposed training protocol. Section 4 evaluates the proposed protocol with three different data sets of sketched symbols against purely supervised baseline techniques, and discusses the implications and future works.

2. Related work

Here we present a summary of related work in Fig. 3 and show their applicability in the form of a decision tree. All the work reviewed here pertains to the classification task defined in the root node of the tree, that is, given a collection of user-generated, labeled training samples $\mathbf{X}^L$, classify unlabeled, user-generated test samples $\mathbf{X}^?$.

If the number of user-generated, labeled training samples $|\mathbf{X}^L|$ far exceeds the dimensionality of a sample $\mathrm{Dim}(x)$, $x \in \mathbf{X}$, it is then straightforward to perform the purely supervised training of a neural network classifier using back-propagation [10]. In this case, the large volume of training set ensures the inclusion of rich stylistic variations within each class, which in turn results in accurate classifiers [14–16]. Otherwise, if the sample size is small, the various approaches reviewed in the subsequent sections can be utilized.

2.1. Synthetic training samples

In case the number of labeled training samples $|\mathbf{X}^L|$ is below the dimensionality of each input, if there exists a sample generator $\mathcal{G}$ that takes existing seed samples $\mathbf{X}^L$ as the inputs and outputs labeled, synthetics samples $\tilde{\mathbf{X}}^L$, then a viable strategy is to computationally generate many supplemental training samples by $\mathcal{G}$, rather than collecting and labeling additional samples by the users. With such synthetic samples $\tilde{\mathbf{X}}^L$, supervised training can be performed on the expanded training set $\tilde{\mathbf{X}}^L \cup \mathbf{X}^L$.

Several distortion-based techniques [27–29] have been developed to generate synthetic samples through global, affine transformations and local, elastic deformations applied to the seed samples. Such distortions emulate the stylistic variations of pen-based input patterns naturally induced by the users. However, the amount of distortions has to be manually tuned through repeated trial-and-error. If too aggressive, the deformations would invalidate the labels of the synthetic training samples. For example, a letter "I" could be deformed into "J" if too much local bending is allowed to the bottom-half. A digit "9" could be subject to excessive rotation and becomes a digit "6". Studies [30,31] have shown that such invalid labels in the training set are detrimental to the performance of the supervised classifiers. If too conservative, the deformations only results in a redundant set of synthetic samples with little variations that does not contribute much to the training.

An alternative to distortion is to interpolate existing training samples that belong to the same class, in the hope that such synthetic samples will share the same, valid labels as the original ones [16,32]. One major limitation is that if the original training

[1] If the input is a feature vector extracted from the input pattern, then the dimension of the input is the number of features. If the input is an image patch, then the dimension of the input is the number of pixels.

[2] That is, the sample size is smaller than or on par with the input dimension.

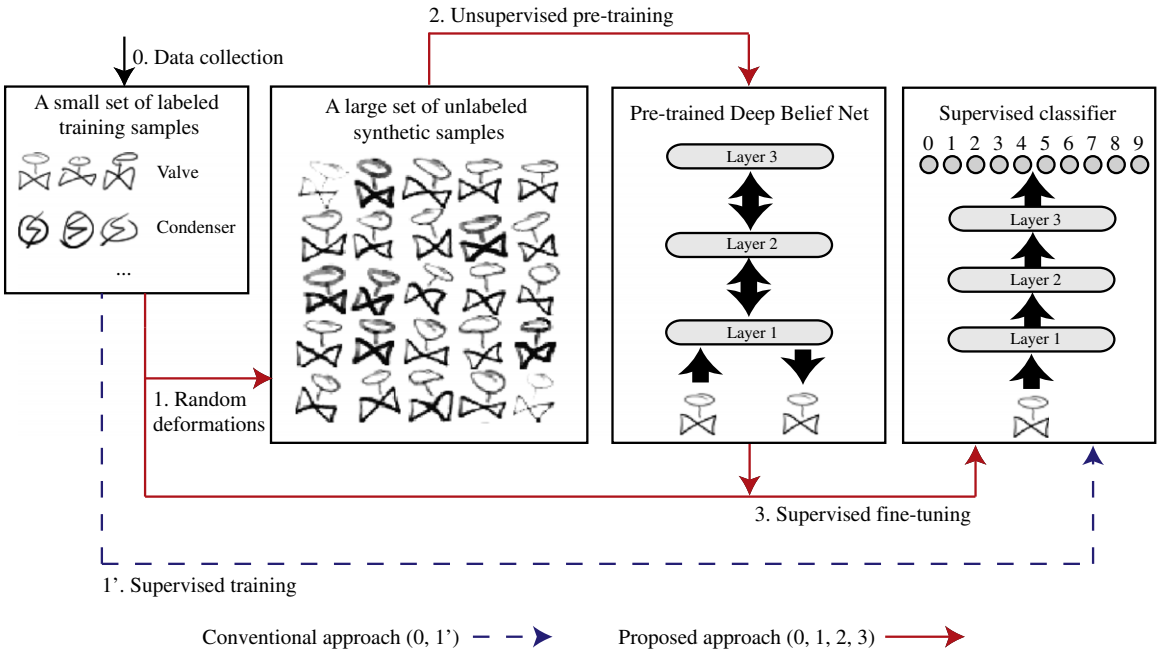

Fig. 2. Overview of the proposed training approach.

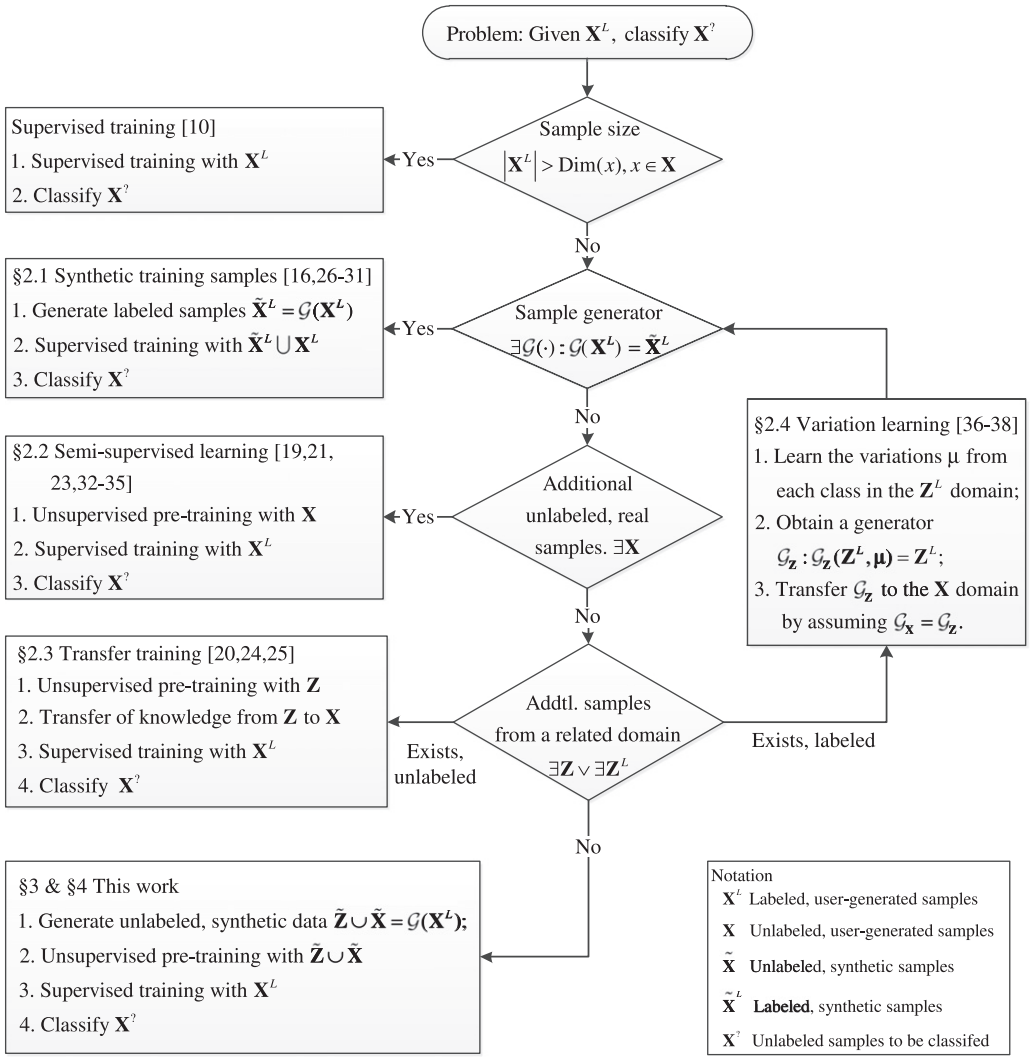

Fig. 3. Overview of the training approaches reviews in Section 2.

samples exhibit little variations among themselves, then interpolation results are similar to that of conservative deformations.

2.2. Semi-supervised learning

In case the aforementioned generator $\mathcal{G}: \mathcal{G}(\mathbf{X}^L) \rightarrow \tilde{\mathbf{X}}^L$ does not exist, if there exists a large set of user-generated, unlabeled samples $\mathbf{X}$ belonging to the same domain as the test set $\mathbf{X}^?$, then semi-supervised learning can utilize such unlabeled samples to improve the classification accuracy. Early examples of semi-supervised learning, exemplified by co-training [33], exploit the redundancy of features and require at least two pre-defined, disjoint feature sets that can potentially yield the same, accurate classification of the input. They are unsuitable for the classification of image patches where the features are the intensities of individual pixels, and the pre-determination of the two disjoint feature sets is overwhelmingly difficult, if not impossible. Universum [34] is a semi-supervised learning method that eliminates the requirement of multiple, disjoint feature sets, but at the cost of new constraint: the unlabeled samples must belong to an extra class not overlapping with the classes to be recognized. Recent approaches [19,23,35,36,21] have lifted such constraints, but still require a sizeable unlabeled training set $\mathbf{X}$.

2.3. Transfer learning

In the absence of a labeled sample synthesizer $\mathcal{G}$ or an unlabeled training set $\mathbf{X}$ from the same domain as the test set $\mathbf{X}^?$, transfer learning [24,25,20] utilizes additional, unlabeled samples $\mathbf{Z}$ from a different, but related domain to learn transferable knowledge that improves the performance in the classification domain. The transferrable knowledge includes feature detectors, parameter initializations and probability priors. For example in [20], handwritten digits is observed to facilitate the classification of handwritten letters, if the feature extractors (i.e., image filters) learned on digits are used on letters.

However, it is unclear how to choose the related domain $\mathbf{Z}$ given the seed samples $\mathbf{X}^L$. A quantified measure of feature relevance or learning transferability between two domains $\mathbf{Z}$ and $\mathbf{X}$, preferably computable without having to actually perform the transfer learning from $\mathbf{Z}$ to $\mathbf{X}$, has yet to be proposed. In addition, the related domain $\mathbf{Z}$ must by itself feature a large training set.

2.4. Variation learning

If the additional samples from the related domain are correctly labeled, i.e., $\exists \mathbf{Z}^L$, it is then possible to learn the variations within each class of $\mathbf{Z}^L$, and then port such knowledge to the domain $\mathbf{X}^L$ to parameterize a sample generator $\mathcal{G}$ that generate a large set of plausible synthetic samples with valid labels. Existing approaches in this line are built upon techniques that learn the variations manifested in a pair [37] or a class [38] of training samples. However, the learning of transformations is by itself dependent on large labeled data sets. For example, the handwritten digit classifier of [38], performing well even with one training sample per class, is crucially dependent on the variations extracted from 1000 handwritten letters belonging to 10 letter classes.

In [37,38], the learning of the variations and the synthesis of labeled samples precedes the supervised training. Alternatively, in [39], the variation learning and the sample synthesis are informed by the feedbacks outputted during the supervised training. Their approach is specialized for the synthesis of positive training samples in binary classification scenarios (e.g., face *versus* non-face patterns), and the large set of negative training samples (e.g., face-like background clutters) still needs to be collected. Also, the alternations between synthesis and the retraining of the classifier are computationally expensive.

2.5. Relations to our approach

Having walked through the decision tree in Fig. 3, it is now clear that our approach addresses a scenario not covered by previously reviewed approaches. In such a scenario the following conditions hold: The labeled training samples are scarce; there is no labeled sample synthesizer; and there is no other sample sets, labeled or unlabeled, to resort to.

Though seemingly rare and constrained among all possible scenarios shown in Fig. 3, the above scenario is likely in practice. Consider the case of developing sketch recognition systems with only a handful of samples from a limited group of pilot users, especially when the sketch recognition system under development is an exploratory prototype intended to recognize a set of newly defined symbols. As such, the novel symbols may share little visual or structural commonalities with existing data sets. And the shareholders, undecided whether to adopt the sketch recognition prototype or its non-pen-based alternatives in the final product, could be unwilling to allocate the resources for large-scale data collection from many human participants. Those barriers prevent the use of the other approaches shown in Fig. 3, leaving only the bottom-left scenario as viable.

3. Training a symbol classifier with a few labeled examples

To circumvent the difficulties of collecting and labeling a large training data set, and to enable the rapid training and deployment of neural network classifiers with limited training samples, we present in the following subsections a three-step training protocol that utilizes synthetic training samples and the unsupervised training [19,22] of DBN models.

3.1. Generating synthetic samples

In this step, the proposed approach starts with a small number of seed samples $\mathbf{X}^L$, i.e., user-generated image patches with categorical labels. A large number of unlabeled, synthetic training images are generated by randomly deforming the seed samples. Various image deformation schemes are incorporated to emulate the stylistic variations inherent in pen-based patterns. For example, the smoothed Gaussian random distortion field [28] mimics the local oscillations of the pen tip along the stroke trajectory. To generate such a distortion field, each pixel is first assigned a vector with each component uniformly drawn from the interval of $(-1, +1)$. Then the random vector field is smoothed by a Gaussian filter with a standard deviation σ. The smoothed random field is then magnified by a factor of α. A distorted image can be then computed by deforming the original image per the distortion vector associated with each pixel. Other transformation, such as the affine transformations (e.g., rotation, scaling and shear), capture variations at the larger, global scale. Morphological dilation and erosion of pixels emulate the variations of ink width due to different drawing scale, different types of pen tips and different pressure applied to the drawing surface during sketching.

The control parameters of the random transformations are tuned by increasing the values used in [28,40], while visually and interactively inspecting a limited subset of resulting image patches. The final parameters are obtained through a few iterations of bracketing, that is, increasing or decreasing the magnitude of the deformations per visual inspection, until the resulting images exhibit rich variations, while a small portion (up to $\approx 10\%$) may

be overly distorted into patterns not belonging to any classes. Such over-distorted, non-symbol samples are not manually removed from the results. The resulting set of synthetic samples, containing synthetic symbol patterns belonging to the domain as well as non-symbol patterns, can therefore be denoted as $\tilde{\mathbf{X}} \cup \tilde{\mathbf{Z}}$.

Unlike previous works [28,40] that generate labeled samples $\tilde{\mathbf{X}}^L$ for supervised training, here the resulting synthetic samples $\tilde{\mathbf{X}} \cup \tilde{\mathbf{Z}}$ are treated as unlabeled samples in subsequent processing. Therefore, it is not required to guarantee the validity of all the labels of the synthetic samples, or manually identify and remove excessively distorted, non-symbol patterns from the synthetic samples, or customize the deformation parameters for each individual symbol class. As a result, the allowed range of deformations are relaxed and more variations are instilled to the synthetic data set, while less human time is needed to find the precise deformation parameters that ensure the correctness of the labels of the synthetic samples.

In the reminder of our paper, we use the following set of deformation parameters:

- random distortion field smoothed by a Gaussian filter with $\sigma = 8$;
- random distortion field magnified by the factor of $\alpha = 45$;
- rotation within the range of $\pm 20^\circ$;
- shearing with the ratio between $\pm \tan 20^\circ$;
- morphological dilation or erosion of 1 or 2 pixels.

Using the above parameters and a user-sketched symbol of "actuated valve", we generate 99 synthetic samples, as shown in Fig. 4. Most of the resulting synthetic symbols appear valid and exhibit a broad range of shape variations perceptually similar to what human users would produce. A small portion (less than 10%) of the resulting patterns are obviously too distorted and lose too much distinguishing features to be recognized as an "actuated valve", mainly due to the random morphological erosion that erases the top part of the valve. However, such overly distorted samples are not discarded from the synthetic training set. They are just treated as unlabeled ones. In Section 4.3, the results from benchmark tests indicate that the excessively deformed samples often cause higher error rates on average, if they are used with the incorrect labels in supervised training. And it is shown that the proposed approach, using those samples as unlabeled ones, produces lower error rates.

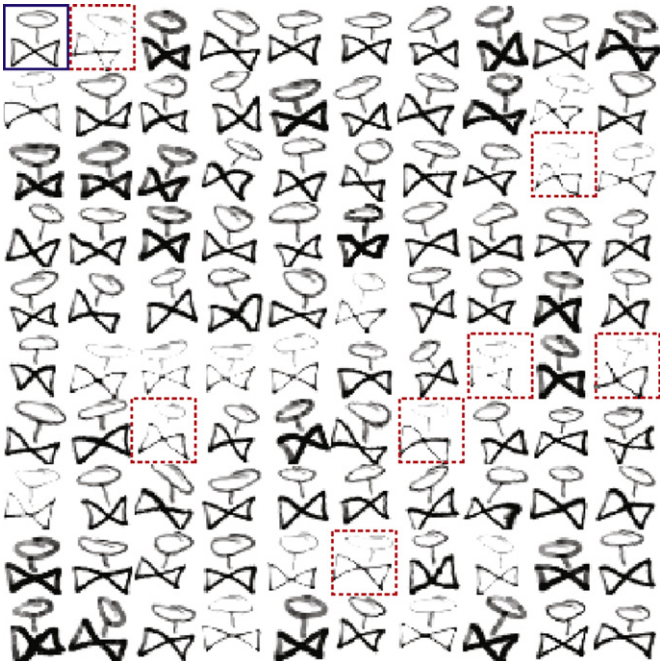

Fig. 4. From one user-sketched symbol of actuated valve (top-left corner, in the solid bounding box), 99 synthetic samples (the rest) are automatically generated through random transformations. Most of the synthetic samples are realistic-looking and feature rich shape variations. Those in the dotted bounding boxes are overly distorted due to the random erosion applied to the pixels, but not removed from the synthetic training set.

3.2. Unsupervised pre-training of DBN

In this step, a DBN model is pre-trained in an unsupervised fashion with the unlabeled training set $\tilde{\mathbf{X}} \cup \tilde{\mathbf{Z}}$ previously synthesized. The learning objective for the DBN is to model the probability of the input images using multiple layers of non-linearities. This objective guides the DBN to learn a generative model of the training set and, in doing so, discover the regularities underlying the training data. The principles and learning algorithm of DBN have been provided in [22,41]. Interested readers are referred to those papers for details. Only a brief overview of DBN is provided below.

A DBN consists of several layers called the Restricted Boltzmann Machines (RBM). Each RBM is a bipartite graphical model of one layer of visible binary nodes $\mathbf{v}$, usually set to the pixels intensities of an input pattern, and one layer of hidden binary nodes $\mathbf{h}$, usually corresponding to the activation of feature detectors. Weighted bi-directional connections exist between both layers, but not within a layer.

Given an input image reshaped as a vector, namely, given the values of the visible layer $\mathbf{v}$, the values of the hidden nodes are computed stochastically by

$$P(h_j = 1 \mid \mathbf{v}) = \sigma\left(b_j + \sum_i v_i w_{ij}\right), \tag{1}$$

where h_j is the j-th element of $\mathbf{h}$, v_i is the i-th element of $\mathbf{v}$, w_{ij} is the weight between visible node v_i and hidden node h_j, b_j is the bias term for hidden node h_j, and $\sigma(z) = 1/(1+\exp(-z))$ is the sigmoid non-linearity. The set of weights w_{*j} from all visible nodes in $\mathbf{v}$ to a single hidden node h_j has the same dimensionality as the input image $\mathbf{v}$. Given the element-wise multiplications between w_{*j} and $\mathbf{v}$, w_{*j} actually serves as a global feature detector or a template for the input image patch.

Given the states of the hidden nodes, the input, namely the states of the visible nodes v_i, can be reconstructed after computing the stochastic states of the visible nodes by

$$P(v_i = 1 \mid \mathbf{h}) = \sigma\left(b_i + \sum_j h_j w_{ij}\right), \tag{2}$$

where b_i is the bias term for visible node v_i,

With the visible nodes clamped to those reconstructions, the states of the hidden layer can be computed again using Eq. (1).

The RBM can be trained to increase the likelihood of reconstructing the input by one-step Contractive Divergence [42]. The weight update equation is

$$\Delta w_{ij} = \varepsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon}), \tag{3}$$

where $\langle v_i h_j \rangle_{data}$ is the probability of the visible node v_i and the hidden node h_j both taking the value of 1 when the visible units are clamped to the input image from the training data set, $\langle v_i h_j \rangle_{recon}$ is the probability of visible node v_i and hidden node h_j both taking the value of 1 when the visible nodes are clamped to the reconstructions, and ε is the learning rate. A similar update equation can be derived for the bias terms.

The unsupervised pre-training of an entire DBN is actually the layer-wise training of each constituent RBM from bottom to top using Contrastive Divergence, followed by the composition of all

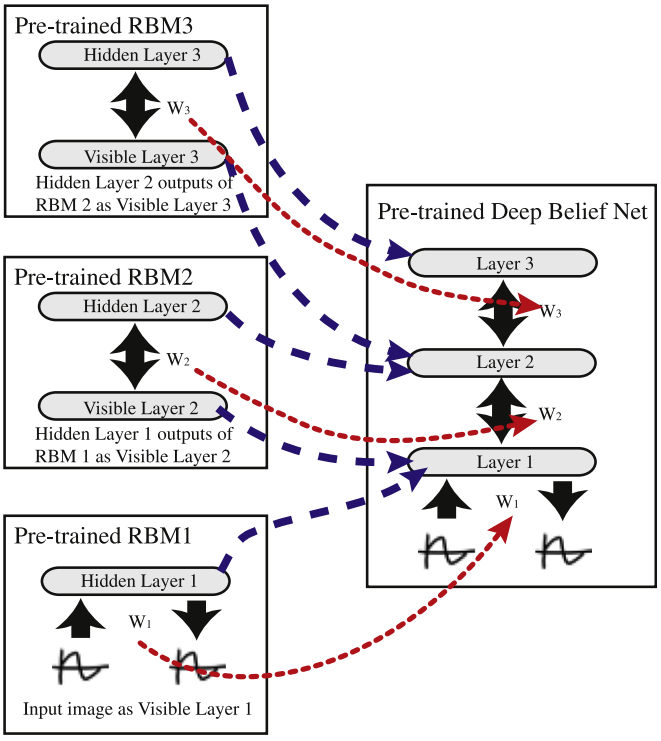

Fig. 5. By merging the shared layers between the three pre-trained RBMs and retaining the trained weights, a three-layer pre-trained DBN is obtained.

RBMs into a single DBN, as illustrated in Fig. 5. The RBM 1 clamped to the input image on the lowest layer of the entire DBN is trained first. Then the states of the hidden nodes in RBM 1 are taken as the inputs or visible units of the next layer RBM 2. For this to happen, the visible layer of RBM 2 should have the same number of units as the hidden layer of RBM 1. The same process repeats until all three RBMs are individually trained and composed together.

The pre-training performed in this step is entirely unsupervised. Ground truth labels of the input patches are not used for training.

3.3. Fine-tuning

At the beginning of the fine-tuning step, a randomly initialized, soft-max output layer with n units is superimposed on the pre-trained DBN to form an extended DBN. Here n equals the number of classes in the recognition task. Each node of the soft-max layer is fully connected to the penultimate layer of the extended DBN (previously the topmost layer of the pre-trained DBN). The activation of each node of the soft-max layer encodes the detection of a pattern class and the outputs are normalized with regard to their sum, thus mimicking probability values.

Then, the DBN is fine-tuned with the training set of labeled seed samples $\mathbf{X}^L$. The learning objective for this step is to minimize the DBN's classification error of the labeled training data set. The optimization is performed through back-propagation, a standard algorithm to train neural networks for supervised tasks and extensively detailed in [10]. To accelerate training, stochastic conjugate gradient is used for parameter updates in lieu of batch gradient descent. Fig. 6 illustrates the process.

After fine-tuning, the DBN can be used for classification and it functions equivalent to a multi-layer, feed-forward neural network. The recognition pipeline involves a series of vector-based operations such as dot-products and vector sums, and it is faster compared to alternative probabilistic inference models that require expensive likelihood maximization or nearest neighbor queries.

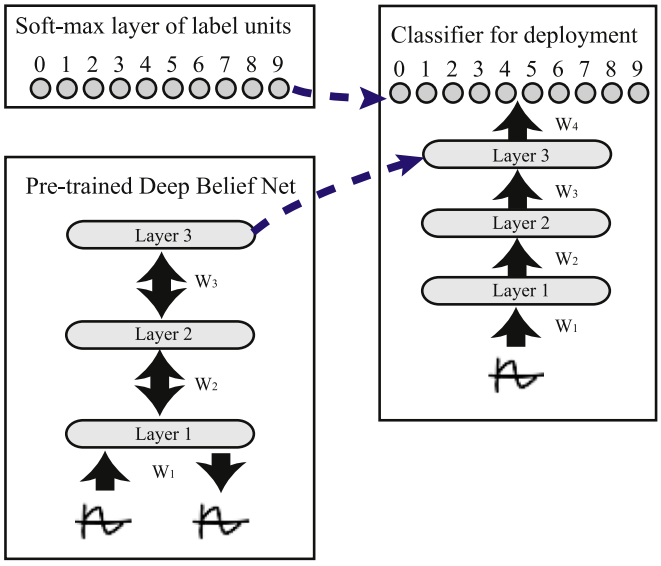

Fig. 6. By adding a soft-max layer on top of the pre-trained three-layer DBN and performing back-propagation fine-tuning, a neural net classifier is obtained.

Table 1
Data sets used for evaluation.

Name	Number of contributing users	Number of symbol classes	Number of training samples	Number of testing samples
HHreco	19	13	650	7141
ThermoFluid12	10	12	120	720
Sym24	19	24	1200	3000
MNIST	Approximately 250	10	60,000	10,000
NicIcon	35	14	9234	10,769

Fig. 7. Sketched shapes from the HHreco data set. Top row: ellipse, heart, trapezoid, pentagon, arch, hexagon, square. Bottom row: triangle, parallelogram, moon, call-out, cube, cylinder.

Working examples of DBNs for image classification are documented in [19,22,43]. A detailed practical guide to train the DBN is provided in [44].

4. Experiments

4.1. Data sets

To provide with a comprehensive benchmark suite, we have chosen to evaluate the proposed training protocol with the data sets summarized in Table 1 and detailed below. Each of the chosen data set contains a large number of test cases generated by different users. And together they cover a broad range of pen-based symbols, varying from uni-stroke numerals to multi-stroke engineering symbols. Among them, the MNIST and HHreco data sets are well-known benchmark cases frequently used in the computer vision and the sketch recognition literature.

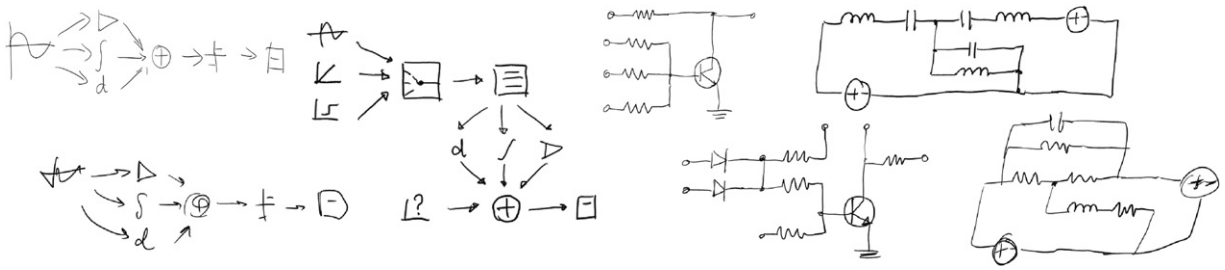

Fig. 8. Various sketches from the Sym24 data set, showing PID control systems, RLC circuits and diode-transistor logic circuits.

4.1.1. HHreco

The HHreco data set of sketched symbols [45] includes 7791 PowerPoint shapes in 13 classes including boxes, ellipses, cylinders, call-outs, *etc.*, as shown in Fig. 7. The symbols were collected from 19 different users. Originally the data was recorded as time series of pen tip motions. Here they are rendered as gray-scale ink the on a 28×28 black canvas. The HHreco data set is not originally partitioned into training and testing subsets. In this paper, 7141 symbols are picked as testing samples and the remaining 650 are used for training. Here the partition of testing and training subsets is performed to maximize user-independence, such that all but one user's contribution would appear in only one of the subsets. The top performance on HHreco is an error rate of 1.8% reported in [11], where a convolutional neural network classifier was trained with training data from 18 users and tested on one user, a partition significantly different from ours.

4.1.2. ThermoFluid12

The ThermoFluid12 data set[3] is a data set of hand-sketched engineering symbols in the domain of thermofluid systems. It contains 840 symbols in 12 categories contributed by 10 users. The stroke data of each symbol are rendered as gray-scale ink on a 32×32 black canvas. Seven hundred and twenty symbols are selected as testing set and the rest 120 are set aside for training. Exemplary symbols and their labels from this data set are shown in Fig. 9. Users contributing those symbols were instructed to copy individual symbols several times on a pressure-sensitive tablet PCs. This paper is the first to report the performance benchmarks on this new data set.

4.1.3. Sym24

The Sym24 data set[4] is a data set of hand-sketched engineering symbols that we collected to benchmark sketch recognition systems. It contains 4200 symbols in 24 categories contributed by 19 users. The symbols are rendered as gray-scale ink on a 28×28 black canvas. Three thousand symbols are selected as testing set and the rest 1200 are set aside for training. Exemplary symbols and their labels from this data set are shown in Fig. 10. The symbols are collected on pressure-sensitive tablet PCs. At first, users contributing those symbols were instructed to copy individual symbols several times to memorize the shapes. Then, they were shown brief textual descriptions or printed block diagrams of control systems or analogous circuits, and asked to create freehand sketches depicting those diagrams. Only data from the later stage entered into this data set.[5] As a result, the users had the freedom to introduce variations such as scale variations (resulting in different ink widths), over-tracing or pen-drag (i.e., drawing multiple symbols in one stroke). Fig. 8 shows such sketches from which the symbols of Sym24 are labeled and extracted as image patches. This paper is the first to report the performance benchmarks on this new data set.

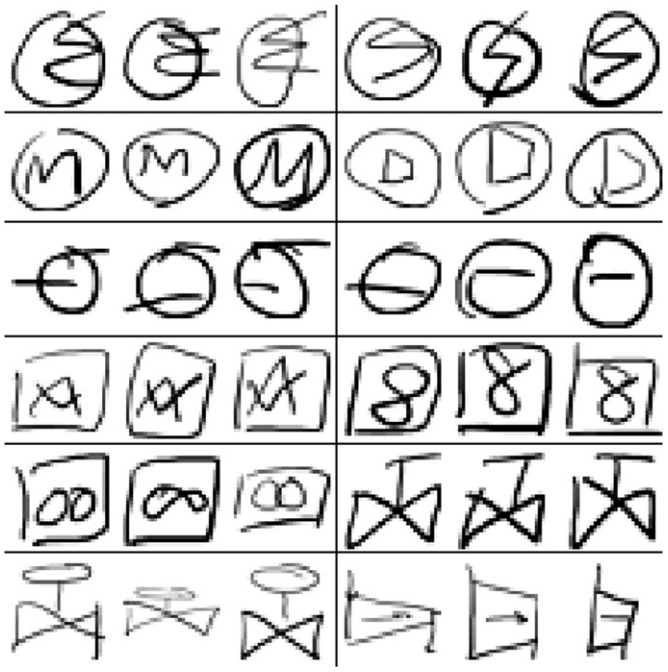

Fig. 9. Sketched symbols from the ThermoFluid12 data set, showing three examples per class. From left to right, top to bottom: heat exchanger, condenser, motor, motor-driven turbine, pump, fluid-level indicator, mixer, axial fan, flow meter, manual valve, actuated valve, turbine.

4.1.4. MNIST

The MNIST data set of handwritten digits [12] is a popular benchmark data set for machine learning models. It contains 60,000 training samples and 10,000 testing samples of digits 0–9. All digits are gray-scale, size-normalized and centered images of 28×28 pixels. Representative samples are randomly drawn from the testing set and shown in Fig. 11. The state-of-the-art

[3] Available online at http://www.andrew.cmu.edu/user/luotingf/SketchRecog/datasets/.

[4] Available online at http://www.andrew.cmu.edu/user/luotingf/SketchRecog/datasets/.

[5] This two-stage data collection process was initially motivated by our hypothesis that the users might exhibit different levels of stylistic variations while sketching, if (1) their familiarities with the symbols differ (i.e., unfamiliar versus familiar) and (2) the sketching tasks differ (e.g., creating a facsimile of existing sketch versus open-ended synthesis, or externalizing ideas by way of a sketch). We further hypothesized that the difference in the levels of stylistic variations could affect the effectiveness of the training: low-variation data may not lead to highly accurate classifier. The test of this hypothesis is beyond the scope of this work. Alternatively we would like to point to the recent discovery of [46], where the authors report no significant difference in the testing accuracy, when the training data was collected through different tasks.

Fig. 10. Sketched symbols from the Sym24 data set, showing three examples per class. From left to right, top to bottom: scope, gain, mux, constant, sum, transfer function, clock, sine wave, step, derivative, integration, sign function, Columbus friction, random number, AC source, V source, capacitor, resistor, inductor, diode, transistor, ground.

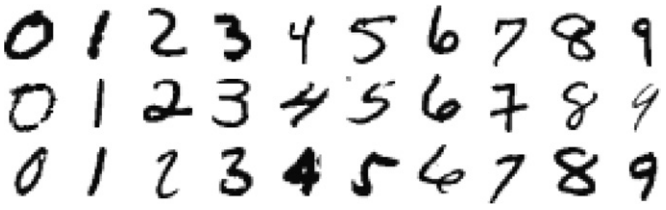

Fig. 11. Digits from the MNIST data set.

performance trained on full MNIST is an overall error rate of 0.32% reported in [40]. Top performing MNIST classifier trained with only a subset of the 60,000 training samples was reported in [47]. Their results are shown in the last column of Table 7.

4.1.5. NicIcon

The NicIcon data set of sketched symbols [48] contains 26,163 examples belonging to 14 classes, each representing a specific object or event in the context of crisis management. Representative examples from this data set are shown in Fig. 12. This data set is partitioned into the training set, the validation set and the testing set, each containing approximately 36% (9234), 24% (6169) and 40% (10,769) of the samples. The original data set was recorded in the online format as streams of the pen-tip trajectories and the corresponding time stamp and normal pressure value at each sample point. Here we have rendered the ink as 48×48 gray-scale images. To our knowledge, no prior result has been reported regarding the offline, image-based recognition performance on this data set.

4.2. Procedure

4.2.1. Model and hyper-parameters

A four-layer neural nets, namely a DBN composed of three RBMs and one soft-max layer, is used in all test cases. The input of the DBN is clamped to the pixels of the input image. Each layer of

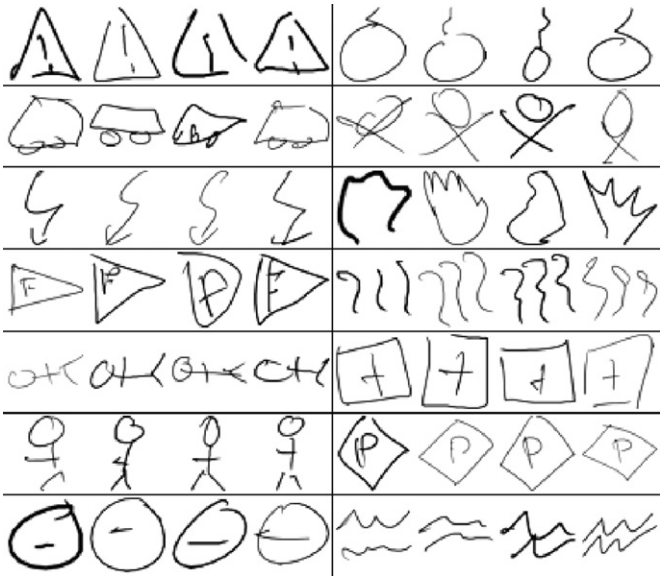

Fig. 12. Sketched symbols from the NicIcon data set, showing four examples per class. From left to right, top to bottom: accident, bomb, car, casualty, electricity, fire, fire brigade, flood, gas, injury, paramedics, person, police, road block.

the DBN has 500, 500, 2000, n_{class} units, respectively, where n_{class} is the number of classes in the data set. The hyper-parameters used with all data sets are summarized in Table 2.

4.2.2. Tested approaches and abbreviations

The proposed approach is compared against several baseline approaches shown in Table 3. All the approaches are tagged using an abbreviation system that highlights the key differences from the proposed approach.

The proposed training approach is tagged as "SynHdHo → UnPre → TuneSeed", to represent the following three key steps.

- SynHdHo: generating a large number of synthetic samples which exhibit high deformation and therefore possibly high percentage of overly distorted outliers.
- UnPre: performing unsupervised DBN pre-training.
- TuneSeed: supervised fine-tuning using only the seed samples.

Comparable baseline approaches are enumerated by varying the three steps of the above approach. For the first step ("SynHdHo"), a valid alternative is to avoid generating synthetic data, but only resort to the seed samples in the subsequent neural network training. This amounts to simply supervised training with a limited amount of seed samples. For this condition, we note that a neural network classifier is not expected to perform well, given the small number of training samples. We therefore augment the results by reporting the better accuracy of the neural network classifier and a nearest-neighbor classifier trained on the seed data. This condition is tagged as "NoSyn".

If alternatively "Syn" is chosen, further variations are obtained by departing from "HdHo", resulting in an alternative such as "HdLo". "HdLo" represents the "ideal", well-engineered synthesizer that introduces high deformation while keeping the outliers low, which is non-trivial to obtain in practice. Here we emulate the ideal "HdLo" synthesizer by selecting more user-generated, labeled samples from the large data sets such as MNIST and NicIcon. In other words, we utilize the abundant extra samples in those data sets to simulate the existence of an ideal synthesizer that can generate the "HdLo" samples. It will be shown later that

Table 2
Hyper-parameters that control the training process.

Hyper-parameter	Value	
	Pre-training	Fine-tuning
Number of iterations	50 per layer	100
Batch size	Minimum of 100 or all samples	Minimum of 500 or all samples
Learning rate	0.01	Automatic
Weight decay	0.002	None

Table 3
Abbreviations and details of the tested approaches.

Abbreviation	Generate synthetic training samples?	How much deformation/outliers during synthesis?	Unsupervised DBN pre-training?	Fine-tuning with synthetic data, or with seed samples only?
NoSyn	No	–	–	–
SynHdHo → UnPre → TuneSyn	Yes	High deformation, high outliers	Yes	Synthetic samples
SynHdLo → UnPre → TuneSeed	Yes	High deformation, low outliers	Yes	Seed samples
SynHdHo → UnPre → TuneSeed (Proposed approach)	Yes	High deformation, high outliers	Yes	Seed samples

"HdLo" produces on average lower error rates than the proposed "HdHo", but the majority of the differences are not significant.

Another alternative "LdHo", however, is not included, because it is naturally counter-productive to have low deformation yielding high percentage of outliers in the training data. The last possible alternative "LdLo", indicating a conservative sample synthesizer that introduces low deformation and low outliers, is not included either, because empirical evidence suggests the need for high deformations to achieve high accuracy [28,40].

For the second step, namely pre-training, we choose to keep "UnPre" (Unsupervised DBN Pre-training), rather than skipping it, because literature has shown the performance improvement brought by unsupervised pre-training [19,23,35,36,21]. We also note that DBN pre-training is unsupervised by nature, as described in Section 3.

In the final step of fine-tuning, "TuneSeed" is switched to "TuneSyn", in which supervised fine-tuning is performed using all the synthetic samples, with the assumption that they all inherit the labels of their ancestral seed samples during synthesis, and without removing the overly distorted outliers.

To sum up, the baseline approaches, shown in Table 3, are selected to be compared with the proposed approach. They all correspond to valid options dealing with limited training samples. For example, "SynHdHo → UnPre → TuneSyn" describes a process in which the user starts from a limited amount of seed samples, generates a large number of synthetic samples using high magnitude of deformations, treat all synthetic samples as labeled (thus possibly polluting the synthetic data set with resulting outliers in the process), and use all synthetic data for the unsupervised pre-training and supervised training of the DBN classifier.

While comparing the above approaches, we have also compared the number of labeled, seed samples initially available in each test condition, varying among 1, 10 and 50 per class, if the data size permits. The seed samples are randomly selected from the training subsets of each data set.

The target number of synthetic samples are approximately 30,000. The exact number varies for each data set, such that the total number of training samples, including the initially available seed samples and the subsequently synthesized samples, sums up to 30,000, which is an artificially set threshold larger than the dimensionality of the input image patch, yet bounded so that the time required for training remains acceptable.

Table 4
The error rates (%) with HHreco test set. Results are obtained on 7141 test symbols belonging to 13 classes.

Number of labeled samples per class	NoSyn	SynHdHo → UnPre → TuneSeed
1	13.76 ± 5.26	8.15 ± 2.83
10	10.63 ± 4.47	6.90 ± 2.78
50	7.62 ± 1.31	5.75 ± 2.44

Table 5
The error rates (%) with the ThermoFluid12 test set. Results are obtained on 720 test symbols belonging to 12 classes.

Number of labeled samples per class	NoSyn	SynHdHo → UnPre → TuneSeed
1	26.13 ± 4.18	18.71 ± 4.22
10	16.51 ± 2.39	10.34 ± 3.47

Table 6
The error rates (%) with the Sym24 test set. Results are obtained on 3000 test symbols belonging to 24 classes.

Number of labeled samples per class	NoSyn	SynHdHo → UnPre → TuneSyn	SynHdHo → UnPre → TuneSeed
1	52.91 ± 4.82	47.82 ± 7.80	37.19 ± 5.72
10	40.02 ± 5.33	31.47 ± 5.82	24.62 ± 4.51
50	20.57 ± 5.29	15.58 ± 6.19	13.96 ± 4.24

Repeated trials and random initial weights are used to rule out the biases from chance factors such as luckily initializing one of the tested approaches with random weights close to a good local optimal, or picking a highly variable and representative subset of the training data for some approaches.

4.3. Results and discussion

The error rates with the test data sets are summarized in Tables 4–8, and plotted in Fig. 13. In each cell of the tables, we report the average performance of 10 repeated trials on the large

Table 7
The error rates (%) on the MNIST test set. Results are obtained on 10,000 test digits belonging to 10 classes.

Number of labeled samples per class	NoSyn	SynHdHo → UnPre → TuneSyn	SynHdLo → UnPre → TuneSeed	SynHdHo → UnPre → TuneSeed	Best of reported results from [47]
1	63.46 ± 5.27	49.83 ± 4.23	43.61 ± 5.23	47.08 ± 4.65	–
10	29.26 ± 2.40	15.71 ± 5.69	14.89 ± 2.01	17.35 ± 2.59	6.5
50	15.74 ± 1.79	7.98 ± 3.78	7.13 ± 4.25	7.84 ± 2.13	2.46

Table 8
The error rates (%) with NicIcon test set. Results are obtained on 10,769 test symbols belonging to 14 classes.

Number of labeled samples per class	NoSyn	SynHdHo → UnPre → TuneSyn	SynHdLo → UnPre → TuneSeed	SynHdHo → UnPre → TuneSeed
1	57.12 ± 8.46	48.39 ± 7.82	44.82 ± 9.96	47.11 ± 6.83
10	41.83 ± 12.05	32.87 ± 6.99	22.15 ± 5.06	27.49 ± 7.38
50	25.36 ± 11.48	23.52 ± 7.56	21.43 ± 7.72	22.70 ± 4.71

data set of MNIST and up to 5 repeated trials on the other data sets.

4.3.1. *The synthetic training samples*

In the "Syn-" conditions, synthetic samples are generated by global and local deformations. With our approach, the magnitude of those deformations are roughly tuned such that the majority of the synthetic samples exhibit high magnitude of visual variations. A number of the synthetic symbols are severely deformed into non-symbol outliers, but are tolerated and retained. This corresponds to the "HdHo" cases as shown in the Tables 4–8.

It can be seen from the results that approaches with synthetic samples ("Syn-") systematically outperform the approaches with only supervised training with the small seed data sets "NoSyn". Compared with the emulated "HdLo" cases, the error rates of the "HdHo" are not significantly lower, although the mean error rates appear lower. This is not unexpected, given that the "HdLo" cases are simulated by selecting more user-generated, correctly labeled samples from the large data sets.

The practical implication here is that our training protocol can tolerate outlier symbols that are outside the subspace of valid training samples. Users deploying the proposed training protocol for their application do not have to carefully tune the parametric deformations to ensure that all the synthetic samples belong to the

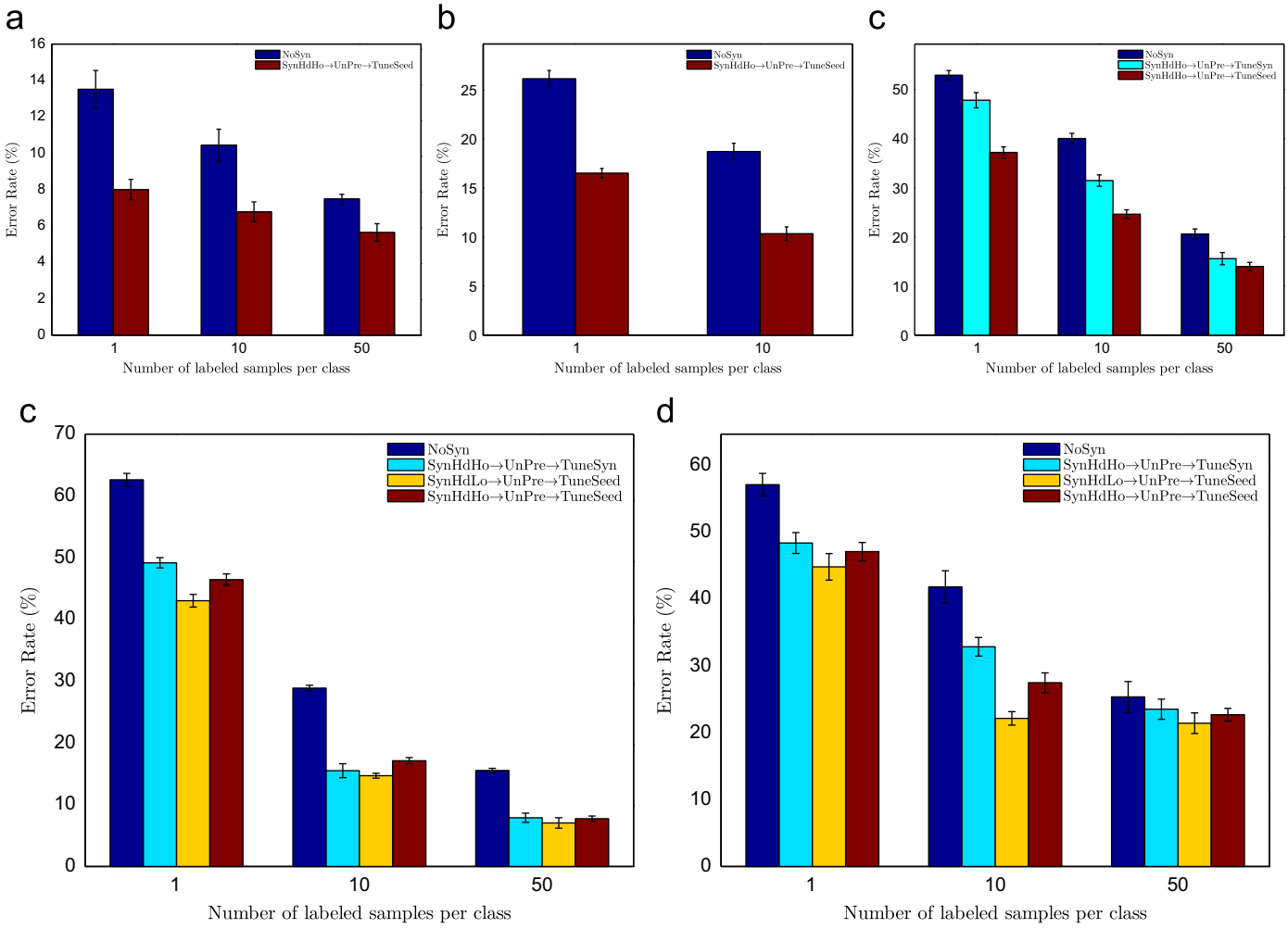

Fig. 13. The error rates (%) obtained with the different test sets. NoSyn: purely supervised training, without using synthetic data; SynHdLo/SynHdHo: using synthetic data with high deformation and low/high percentage of outliers; UnPre: unsupervised DBN pre-training; TuneSeed: supervised fine-tuning using seed examples only; TuneSyn: supervised fine-tuning using seed examples plus synthetic samples. Error bars represent standard error. (a) HHreco. (b) ThermoFluid12. (c) Sym24. (d) MNIST. (e) NicIcon.

classes of interest. Nor do they have to sift through the synthetic training data set and remove excessively deformed samples. Upon a visual inspection of a subset of the synthesized samples, users can quickly determine a rough range of the deformation parameters by bracketing and use them for the subsequent training. That being said, there is nothing to stop the user from looping through all synthetic samples and manually picking out the severely deformed ones that are not recognizable as valid symbols. The visual inspection of samples and the removal of invalid shapes equates to labeling pre-segmented image patches and can be conveniently completed by a few users with a well-designed human-computer interface.

4.3.2. Fine-tuning with only seed samples

When samples are synthesized through deformation, one might wonder why not simply inherit the labels from the seed samples and use both seed and synthetic samples during the supervised fine-tuning. Our results show that this may lead to worse recognition rates, as reflected in the comparison between the "HdHo → TuneSyn" and "HdHo → TuneSeed" conditions on the Sym24, MNIST and NicIcon data set. We hypothesize that the performance deterioration might be caused by the excessively distorted outliers. Specifically, we believe that in "TuneSeed", the small number of samples plays as a regularizer on the decision boundaries between classes, whereas in "TuneSyn" the distorted samples affects the decision boundaries and causing over-fitting to the synthetic deformation, rather than the true distribution of user-generated symbols. This raises the question: what regularization techniques could be utilized to mitigate the effects of outliers in this context? The test of the above hypothesis and the open issue of regularization are to be investigated.

One practical implication of fine-tuning with only seed samples is that the time for DBN fine-tuning can be shortened, compared to training with the large number synthetic samples, because the training algorithm iterates each training sample and the time needed is proportional to the sample size.

4.3.3. The effect of the deformations

One might wonder, how do the deformations used in example synthesis affect the recognition accuracy in our approach? There seems to be two parallel effects and no simple correlation. On one hand, a larger deformation during synthesis causes higher percentage of outliers, which we have shown is detrimental for the performance of the trained recognizer. On the other hand, this is confounded by the opposing effect that a larger deformation produces richer in-class variations which is desirable for training. In the training scenario that we target, there is not an explicit model for the in-class variations, nor a fine-tuned deformation generator that produces rich variations and no outlier. As a result, there is no trivial way to decouple the two effects or control them individually.

4.3.4. Relations to existing approaches and results

The observed improvements with synthetic data and unsupervised pre-training complement previous findings such as [23–25]. In this paper, the improvement of classification accuracy is observed when labeled samples are scarce, whereas the reported improvements of semi-supervised learning over supervised learning are observed in a different performance regime where training samples are already available in large quantities. Altogether, they empirically show the consistent advantage of unsupervised pre-training over conventional supervised learning regardless of the availability of labeled samples.

We note that our main goal here is not to obtain the record-breaking classifier on the tested data sets, but to study the effects of synthetic samples and unsupervised pre-training against comparable baselines when labeled data is scarce. Techniques known to facilitate visual recognition, such as convolution-like weight sharing [12] or kernel methods, are not incorporated here. Therefore, our error rates reported on those data sets are not expected to be on par with the state-of-the-art and we believe that there is room for future improvement.

To the best knowledge of the authors, the top results on MNIST subsets, shown in the last column of Table 7, was obtained by the Patchwork of Parts (POP) model [47] which is a well engineered, specialized visual learning model enhanced by the prior human knowledge that visual objects can be described by deformable models of binary oriented edges. However, the POP model has made the simplifying assumption that parts can only be translated, not rotated. Therefore, it is not robust against large global rotations ($\pm 15^{\circ}$, [47]) which can be prominent in freehand sketched symbols. In contrast, techniques have been developed incorporate large rotation invariance into the neural network classifiers [49].

The best result on HHreco, an error rate of 1.8% reported by [11], was obtained using a convolutional neural network model trained on a majority fraction of the whole data set (i.e., samples contributed by all but one user) and cannot be compared with our results which is by contrast trained with only a small portion of the data set.

5. Conclusion

We present a training protocol for neural network-based symbol recognizers. The proposed protocol consists of three-steps: the synthesis of supplemental, unlabeled training samples, the unsupervised pre-training of a DBN and the supervised back-propagation fine-tuning that turns the pre-trained DBN into a neural network classifier. Systematic improvements over purely supervised training are observed, suggesting that the proposed protocol is able to exploit the unlabeled samples synthesized from a few labeled samples, and alleviates the dependence on a large, labeled training set.

With our approach, the training and deployment of the symbol recognizers could proceed after collecting only 10–50 training samples from a small group of the users, shortening the development cycle. This is particularly useful when introducing the recognizer for a new domain where existing, labeled training symbol data set is scarce.

It remains open to ascertain the dependency of this approach on Deep Belief Networks and the possibility of extending to learning models of simpler parametric structure, such as logistic regression and Support Vector Machines, or models of non-parametric nature, such as the nearest neighbor classifier.

References

[1] Davis R. Magic paper: sketch-understanding research. Computer 2007;40(9):34–41.
[2] Olsen L, Samavati FF, Sousa MC, Jorge JA. Sketch-based modeling: a survey. Computers & Graphics 2009;33(1):85–103.
[3] Fu L, Kara LB. Recognizing network-like hand-drawn sketches: a convolutional neural network approach. In: Proceedings of ASME 2009 international design engineering technical conferences (IDETC) & computers and information in engineering conference (CIE), no. DETC2009-87402, San Diego, CA, 2009. p. 1–8.
[4] Ouyang TY, Davis R. Learning from neighboring strokes: combining appearance and context for multi-domain sketch recognition. In: Bengio Y, Schuurmans D, Lafferty J, Williams CKI, Culotta A, editors. Advances in neural information processing systems. Cambridge, MA: MIT Press; 2009. p. 1401–9.
[5] Kara LB, Gennari L, Stahovich TF. A sketch-based tool for analyzing vibratory mechanical systems. Journal of Mechanical Design 2008;130(10):101101.

[6] Gennari L, Kara LB, Stahovich TF. Combining geometry and domain knowledge to interpret hand-drawn diagrams. Computers & Graphics 2005;29(4):547–62.
[7] Cowans PJ, Szummer M. A graphical model for simultaneous partitioning and labeling. In: Cowell RG, Ghahramani Z, editors. Proceedings of the 10th international workshop on artificial intelligence and statistics, Society for Artificial Intelligence and Statistics, Barbados, 2005. p. 73–80.
[8] Kara LB, Stahovich TF. An image-based, trainable symbol recognizer for hand-drawn sketches. Computers & Graphics 2005;29(4):501–17.
[9] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323(6088):533–6.
[10] Bishop CM. Neural networks for pattern recognition. Oxford, UK: Oxford University Press; 1995.
[11] Ouyang TY, Davis R. A visual approach to sketched symbol recognition. In: Proceedings of the 21st international joint conferences on artificial intelligence. Pasadena, CA: Morgan Kaufmann; 2009. p. 1463–8.
[12] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 1998;86(11):2278–324.
[13] Garcia C, Delakis M. Convolutional face finder: a neural architecture for fast and robust face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 2004;26(11):1408–23.
[14] Ho TK, Baird HS. Large-scale simulation studies in image pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 1997;19(10):1067–79.
[15] Banko M, Brill E. Mitigating the paucity-of-data problem: exploring the effect of training corpus size on classifier performance for natural language processing. In: Proceedings of the first international conference on human language technology research. Morristown, NJ, USA: Association for Computational Linguistics; 2001. p. 1–5.
[16] Varga T, Bunke H. Comparing natural and synthetic training data for off-line cursive handwriting recognition. In: Proceedings of the ninth international workshop on frontiers in handwriting recognition. Washington, DC, USA: IEEE Computer Society; 2004. p. 221–5.
[17] Vapnik VN. Statistical learning theory. New York: Wiley; 1998.
[18] Wang J, Plataniotis KN, Lu J, Venetsanopoulos AN. On solving the face recognition problem with one training sample per subject. Pattern Recognition 2006;39(9):1746–62.
[19] Hinton GE, Osindero S, Teh Y. A fast learning algorithm for deep belief nets. Neural Computation 2006;18(7):1527–54.
[20] Raina R, Battle A, Lee H, Packer B, Ng AY. Self-taught learning: transfer learning from unlabeled data. In: Proceedings of the 24th international conference on machine learning. New York, NY, USA: ACM; 2007. p. 759–66.
[21] Ranzato M, Huang FJ, Boureau Y-L, LeCun Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: Proceedings of computer vision and pattern recognition conference. Minneapolis, MN: IEEE Press; 2007. p. 1–8.
[22] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science 2006;313(5786):504–7.
[23] Salakhutdinov R, Hinton G. Using deep belief nets to learn covariance kernels for Gaussian processes. In: Platt J, Koller D, Singer Y, Roweis S, editors. Advances in neural information processing systems. Cambridge, MA: MIT Press; 2008. p. 1249–56.
[24] Salakhutdinov R, Hinton GE. Deep Boltzmann machines. In: Proceedings of the international conference on artificial intelligence and statistics, vol. 5, 2009. p. 448–55.
[25] Nair V, Hinton G. 3D object recognition with deep belief nets. In: Bengio Y, Schuurmans D, Lafferty J, Williams CKI, Culotta A, editors. Advances in neural information processing systems. Cambridge, MA: MIT Press; 2009. p. 1339–47.
[26] Sezgin TM, Davis R. Sketch recognition in interspersed drawings using time-based graphical models. Computers & Graphics 2008;32(5):500–10.
[27] Yaeger L, Lyon R, Webb B. Effective training of a neural network character classifier for word recognition. In: Mozer MC, Jordan MI, Petsche T, editors. Advances in neural information processing systems. Cambridge, MA: MIT Press; 1997. p. 807–13.
[28] Simard PY, Steinkraus D, Platt JC. Best practices for convolutional neural networks applied to visual document analysis. In: Proceedings of the seventh international conference on document analysis and recognition. Washington, DC, USA: IEEE Computer Society; 2003. p. 958–63.
[29] Thomas A. Govindaraju V. Generation and performance evaluation of synthetic handwritten CAPTCHAs. In: Proceedings of the 11th international conference on frontiers in handwriting recognition, no. 1071, Montréal, Québec, Canada, 2008.
[30] Brodley CE, Friedl MA. Identifying mislabeled training data. Journal of Artificial Intelligence Research 1999;11:131–67.
[31] Bouveyron C, Girard S. Robust supervised classification with mixture models: learning from data with uncertain labels. Pattern Recognition 2009;42(11):2649–58.
[32] Nonnemaker J, Baird HS. Using synthetic data safely in classification. In: Berkner K, Likforman-Sulem L, editors. Document recognition and retrieval XVI, vol. 7247. San Jose, CA, USA: SPIE; 2009. p. 72470G.
[33] Blum A, Mitchell T. Combining labeled and unlabeled data with co-training. In: Proceedings of the 11th annual conference on computational learning theory. New York, NY, USA: ACM; 1998. p. 92–100.
[34] Weston J, Collobert R, Sinz F, Bottou L, Vapnik V. Inference with the universum. In: Proceedings of the 23rd international conference on machine learning. New York, NY, USA: ACM; 2006. p. 1009–16.
[35] Ando RK, Zhang T. A framework for learning predictive structures from multiple tasks and unlabeled data. Journal of Machine Learning Research 2005;6:1817–53.
[36] Fei-Fei L, Fergus R, Perona P. One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence 2006;28(4):594–611.
[37] Memisevic R, Hinton GE. Unsupervised learning of image transformations. In: IEEE computer society conference on computer vision and pattern recognition. Los Alamitos, CA, USA: IEEE Computer Society; 2007. p. 1–8.
[38] Learned-Miller E. Data driven image models through continuous joint alignment. IEEE Transactions on Pattern Analysis and Machine Intelligence 2006;28(2):236–50.
[39] Chen J, Chen X, Yang J, Shan S, Wang R, Gao W. Optimization of a training set for more robust face detection. Pattern Recognition 2009;42(11):2828–40.
[40] Ciresan DC, Meier U, Gambardella LM, Schmidhuber J. Deep big simple neural nets excel on handwritten digit recognition. Technical Report 〈arXiv:1003.0358〉; 2010.
[41] Bengio Y. Learning deep architectures for AI. Foundations and Trends in Machine Learning 2009;2(1):1–127.
[42] Hinton GE. Training products of experts by minimizing contrastive divergence. Neural Computation 2002;14(8):1771–800.
[43] Hinton GE. To recognize shapes. In: Cisek P, Drew T, Kalaska J, editors. Computational neuroscience: theoretical insights into brain function. Elsevier; 2007.
[44] Hinton GE. A practical guide to training restricted Boltzmann machines. Technical Report, UTML TR 2010-003, Department of Computer Science, University of Toronto; 2010.
[45] Hse H, Newton AR. Sketched symbol recognition using Zernike moments. In: Proceedings of the 17th international conference on pattern recognition, vol. 1, Cambridge, UK, 2004. p. 367–70.
[46] Field M, Gordon S, Peterson E, Robinson R, Stahovich T, Alvarado C. Technical section: the effect of task on classification accuracy: using gesture recognition techniques in free-sketch recognition. Computers & Graphics 2010;34:499–512.
[47] Amit Y, Trouve A. POP: patchwork of parts models for object recognition. International Journal of Computer Vision 2007;75(2):267–82.
[48] Niels R, Willems D, Vuurpijl L. The NicIcon database of handwritten icons. In: 11th International conference on the frontiers of handwriting recognition (ICFHR 2008), Montreal, Canada, 2008.
[49] Osadchy M, LeCun Y, Miller M. Synergistic face detection and pose estimation with energy-based models. Journal of Machine Learning Research 2007;8:1197–215.