Proceedings of the ASME 2009 International Design Engineering Technical Conferences &
Computers and Information in Engineering Conference
IDETC/CIE 2009
August 30-September 2, 2009, San Diego, USA

DRAFT: DETC2009-87402

RECOGNIZING NETWORK-LIKE HAND-DRAWN SKETCHES - A CONVOLUTIONAL NEURAL NETWORK APPROACH

Luoting Fu
Mechanical Engineering Dept.
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: luotingfu@cmu.edu

Levent Burak Kara[*]
Mechanical Engineering Dept.
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: lkara@andrew.cmu.edu

ABSTRACT

Hand-drawn sketches are powerful cognitive devices for the efficient exploration, visualization and communication of emerging ideas in engineering design. It is desirable that CAD/CAE tools be able to recognize the back-of-the-envelope sketches and extract the intended engineering models. Yet this is a non-trivial task for freehand sketches. Here we present a novel, neural network-based approach designed for the recognition of network-like sketches. Our approach leverages a trainable, detector/recognizer and an autonomous procedure for the generation of training samples. Prior to deployment, a Convolutional Neural Network is trained on a few labeled prototypical sketches and learns the definitions of the visual objects. When deployed, the trained network scans the input sketch at different resolutions with a fixed-size sliding window, detects instances of defined symbols and outputs an engineering model. We demonstrate the effectiveness of the proposed approach in different engineering domains with different types of sketching inputs.

INTRODUCTION

Freehand engineering sketches have unique traits that differ themselves from the formal CAD/CAE models: the former are *sketchy*, informal and with minimal commitment to details and precision, while the latter are quite the opposite. Sketches are powerful cognitive devices in the early stages of engineering design where their minimalist traits and ease of construction enable the engineers to efficiently and conveniently explore emerging ideas [1]. They also relieve the engineers of the mental burdens of attending to less relevant intricacies such as the precise size, shape, location and color, allowing them to focus on the more central issues and thus facilitate creativity for conceptual design [2]. Moreover, sketches are compact visual representations of rich engineering information regarding the spatial configurations and hierarchy of objects, temporal flow and/or causal relations of events, and they enhance the communication of abstract ideas in a collaborative setting. Several empirical studies have evidenced the strong positive correlation between the quality of the final design outcome and the number of sketches drawn in various stages of the design process [3–7].

Researchers agree that, based on the important role of sketches in design, it is desirable that CAD/CAE tools be able to work with freehand sketching inputs and extract the intended engineering models. For instance, Ullman *et al.* [8] observed that sketches are particularly useful graphical representations and proposed that "CAD systems must allow for sketching input." Schutze *et al.* [4] concluded that "digital sketching tools... can create potentially large time and cost savings for computer-aided design in mechanical engineering." Indeed, the rapid, autonomous conversion from the back-of-the-envelop sketches to formal CAD/CAE models, if achieved, would enable the engineers to harness the computational power of sophisticated CAD/CAE tools at the very early stages of the design process while still enjoying the natural fluidity and ease of sketched input.

[*]Address all correspondence to this author.

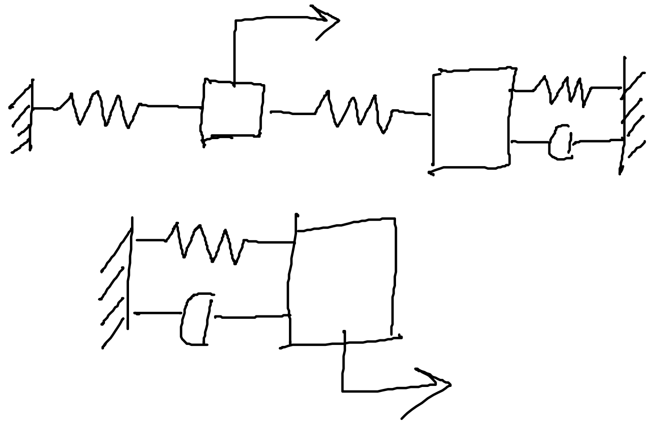


Figure 1. Sketches of mechanical vibration systems that exhibits high variability in drawing styles, scales and over-tracing strokes

However, the conversion from sketches to CAD/CAE models, *i.e.*, the task of *sketch recognition*, is non-trivial. Kara *et al.* [9] summarized the two major challenges in sketch recognition as *sketch parsing,* (*i.e.*, the sub-task of grouping strokes or pixels to form intended symbols, or, in other words, locating the bounding box or convex hull of each symbol) and *symbol recognition* (*i.e.*, the sub-task of mapping each stroke/pixel group obtained in *parsing* into defined symbols). Sketch parsing is complicated by the fact that a freehand sketch contains unknown number of symbols, symbols can be drawn at virtually any location on the canvas and they may be of different sizes. Symbol recognition is rendered difficult because of the inherently informal nature of sketches and the variabilities in personal drawing skills and styles. For example, Figure 1 highlights those challenges: although a body mass is customarily defined as a rectangle, none of the sketches in Figure 1 features perfect rectangular shapes; instead, they exhibit different drawing styles, scale and some even have extraneous, over-tracing strokes. Additionally, we propose the third challenge here: the co-dependence of the sketch parsing and symbol recognition. Sketch parsing would require certain amount of recognition in order to group together strokes or pixels that form defined symbols, meanwhile symbol recognition is predicated on the parsing process that clusters the strokes or pixels in a meaningful way and isolate a symbol from others. Should parsing and recognition be completely de-coupled? Which should precede the other? Should they be tackled iteratively? Those are the challenging questions yet to be investigated.

In this paper, we address the triple challenges by a general approach designed for the recognition of network-like sketches which encompass a wide range of graphical, engineering models, including but not limited to mechanical linkages, multi-body vibration systems, electric circuits, algorithmic flowcharts, control systems, UML diagrams, *etc*. Our approach is based on the Convolutional Neural Network introduced by LeCun *et al.* [10] and also inspired by the success of Convolutional Neural Network for face detection in digital photographs [11, 12]. At the core of our approach is a multi-layer Convolutional Neural Network, acting as a synergetic detector/recognizer for hand-drawn symbols. By applying a sliding window to the input sketch at different resolutions and feeding the content inside the sliding window to the Convolutional Neural Network recognizer, sketch parsing is performed implicitly in conjunction with symbol recognition.

Our approach differs from previous sketch recognition systems in that (1) it is a generalized approach not tailored towards a particular domain, unlike previous works that employ heuristics specific to a certain domain, and (2) it is purely based on the spatial layout of pixel intensities of the input sketches and therefore is applicable to sketches represented as images (*e.g.*, scanned images, screen captures), while most previous works rely on the explicit information of strokes and hence are limited to sketches drawn with a digitizer on a tablet PC.

In the reminder of this paper, we first review the relevant works in sketch recognition and Convolutional Neural Network. Afterwards we present an overview and the details of our approach, followed by evaluations of the proposed approach in two engineering domains with different types of sketching inputs. Finally, we discuss the current limitations, compensations and possible future extensions of our approach and present conclusions.

RELATED WORK

Sketch Recognition

The advent of tablet PC technology and recent surges of interests in pen-enabled software applications have spawned a number of sketch recognition systems. Depending on the internal representations of sketches, those systems can be relegated to two categories: stroke-based or image-based. The former category of systems view sketches as a stream of time-stamped, sample points on the trajectory of the pen tip, captured on a tablet PC or a digital whiteboard. The latter category of systems view a sketch as an image which can be obtained from virtually any image acquisition device (*e.g.*, scanner, digital camera, webcam, screenshot). Below is a brief review of each category with emphasis on systems that recognizes the sketches in engineering domains.

Earlier works in stroke-based recognition require each symbol to be drawn in a single stroke [13, 14] or in predefined order [15], or require explicit user input (*e.g.*, gesturing with the digitizer or pressing a key) to demarcate the various symbols [16]. Such constraints reduce the complexity of the parsing and recognition tasks but compromise the natural fluidity of sketching.

Probabilistic graphical models and inference algorithms have been used for stroke-based sketch recognition. Alvarado and Davis [17] developed a parsing approach that evaluates a set of competing structural hypothesis by conducting inference in dynamically constructed Bayesian networks. Missing or extra strokes can adversely affect the performance, due to the dependence on precise structural shape descriptions. Sezgin and Davis [18] described a statistical framework for the recognition of hand-drawn electric circuits based on Dynamic Bayesian Networks (a generalization of Hidden Markov Model [19], a mathematical model widely used in speech recognition) that allows temporal interspersing of strokes from different symbols. Cowan and Szummer [20] presented an approach based on Conditional Random Fields for simultaneous parsing and recognition and demonstrated their approach in the domain of sketched organizational charts. Both [18, 20] made simplifying assumptions regarding the maximum clique size, so as to make the computation tractable. However, such assumptions may limit the use of their approaches for complex sketches in practice.

Kara *et al.* introduced the *mark-group-recognize* approach and applied to several sketch domains, including control system diagram [21] and multi-body mechanical vibratory system [9]. In the *mark* step, landmark symbols or regions of interest are detected using domain-specific heuristics, so as to provide cues to the subsequent steps. In the *group* step, strokes are grouped together to form clusters of candidate symbols, per their spatial layout relative to the landmarks. In the final step *recognition*, a symbol recognizer is applied to the stroke clusters to yield a full interpretation of the whole sketch. The manual design of domain-specific heuristics is crucial to the success of the entire recognition pipeline and it may require keen observations into the domain at hand. This precludes the easy re-targeting of this approach to new engineering domains.

All the aforementioned recognition approaches are stroke-based and hence dependent on hardware (*e.g.*, tablet PCs, digital whiteboards). They cannot be applied to cases where sketches are acquired as images that store only the spatial layout of pixels, not the temporal information of stroke trajectory.

Image-based recognition poses elevated challenge, due to the absence of *a priori* information of strokes. Saund *et al.* [22] presented a system that uses Gestalt principles to search for the perceptually closed paths in drawings and sketches. Their work only described the parsing technique, but have not employed recognition. Kara *et al.* [23] described an image-based technique for the recognition of isolated symbols. Their recognition technique computes several template matching distances to extract feature vectors from the input image and then uses a Gaussian Bayes classifier to recognize the symbol. It is designed to be used jointly with a stroke-based parser and therefore the entire sketch recognition process is still stroke-based, rather than purely image-based. Notowidigdo and Miller's [24] off-line sketch recognition technique used hand-coded rules to find shapes (*e.g.*, circles, rectangles) in an image rendered from a sketch and used empirical thresholds to filter false positives. It is not straightforward to extend beyond the simple domains of two or three simple geometric primitives. And the empirical thresholds, if not properly tuned, is observed to cause high rate of false positives.

Convolutional Neural Network for Visual Detection

Convolutional Neural Network was first introduced by LeCun *et al.* [10] as a general purpose mathematical model for machine learning. They differs from the conventional, fully-connected neural network [25, 26] in three key architectural aspects: *local receptive fields*, *shared weights* and *sub-sampling*. Local receptive fields describe the sparse, localized fashion that certain layers are connected to previous layers. Weights sharing results in a reduction in the total number of trainable weights and facilitate training. Dot-product of the input with shared weights on localized connections is equivalent to performing convolution on the input. Sub-sampling reduces the resolution of the input and produces a blurred version of the input. Such architectural differences ensure better robustness to moderate size variations and shape distortions in the inputs.

Convolutional Neural Networks have demonstrated record performance in recognizing handwritten digits [27]. Garcia and Delakis [11] developed Convolutional Neural Network face detection systems that outperformed other face detectors with alternative techniques (*e.g.*, hand-crafted features [28], conventional neural network [29]). Osadchy *et al.* [12] developed an improved Convolutional Neural Network face detector capable of estimating face poses while performing recognition, achieving a large degree of rotation invariance.

OVERVIEW

Our approach to freehand, network-like sketch recognition leverages a Convolutional Neural Network as a synergetic detector/recognizer that scans the input sketch at different resolutions with a fixed-size sliding window. We also propose an autonomous procedure for the generation of training samples.

At the heart of our approach is a six-layer Convolutional Neural Network. It maps a fixed-size input image to a code that indicates the symbol category (*i.e.*, the *label*) of the input image. The Convolutional Neural Network takes as input pixel intensity values in a fixed-size region of an input image. The first few layers of the network are alternating convolutional layers and sub-sampling layers that can be trained to extract distinctive local features from the input. The remainder of the network are fully-connected layers that classify the feature matrices extracted by previous layers from input images and produce a human-readable output.

Prior to deployment, a Convolutional Neural Network needs to be trained on a few labeled prototypical sketches from which a training set is initialized. To cover a wider range shape variations of sketched symbols and reduce the incidence of false positive

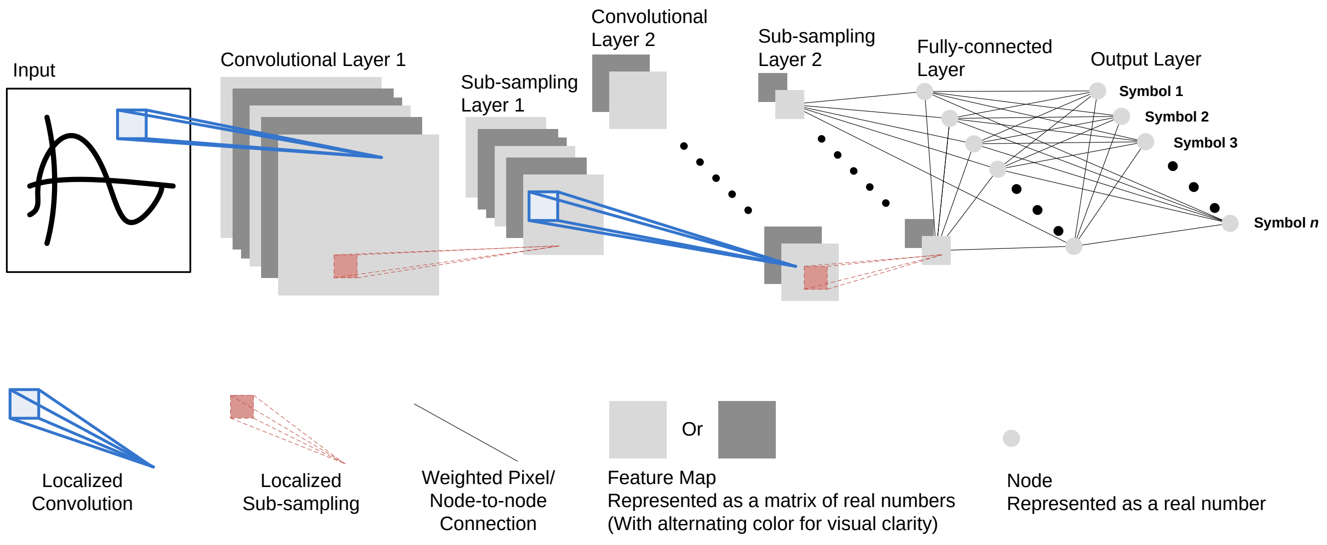


Figure 2. The architecture of the convolutional neural network

recognitions, the training set is then iteratively and autonomously expanded, demanding no further manual interventions from the user.

Once deployed, the trained network scans an input sketch at different resolutions with a fixed-size sliding window. Such exhaustive scanning seeks to cover every region of the input sketch and maximize the chance of detecting each instance of the defined symbol. Measures are taken to address certain computation redundancy during scanning and accelerate the process. Finally, outputs from different sub-windows and different resolutions are merged to build an engineering model.

DETAILS OF THE PROPOSED APPROACH

Network Architecture

The Convolutional Neural Network we propose for sketch recognition has six layers, as shown in Figure 2. The first four layers are alternating convolutional layers and sub-sampling layers, which are architectures unique to Convolutional Neural Networks. The last two layers are fully connected layers, like those of the conventional, fully-connected neural network.

The input to the network (hence the input to the first layer of the Convolutional Neural Network, *i.e.*, Convolutional Layer 1) is an image of fixed size. To ease subsequent discussions and without losing generality, let us assume that the input is an image of 32×32 pixels. The input image is represented as a matrix where each element is the intensity value of the pixel at the corresponding location.

In the Convolutional Layer 1, each output element is connected to a small region (for instance, 5×5 pixels) of the input image. The adjacent elements in the Convolutional Layer 1 are connected to regions in the input image that are largely overlapping, with one pixel offset. This forms the *local receptive fields*. To obtain the value of one output element of this layer, a dot product is performed between pixel values in the 5×5 regions connected to this output element in question and the 25 weights on the connections, added by a bias term. For all output elements, the 25 weights and the bias term involved in the computations share the same values and this architectural characteristic is known as the *shared weights*. The above operation of dot-products across the input image with shared weights amounts to applying a convolutional *kernel* of 5×5 pixels to a 32×32 image and results in a 28×28 real-valued matrix. The 5×5 kernel can be seen as trainable feature extractors, detecting local features such as oriented strokes, strokes ends and junctions. Those feature extractors are to be trained in a data-driven fashion to adapt to the visual features of a specific problem domain, instead of being manually coded. The output matrix of the convolution represents the spatial layout of responses to the feature detectors and are hence called *feature maps*. In practice, more than one sets of 5×5 kernels (feature detectors) are convolved with the input and multiple sets of feature maps are obtained. In our case of Convolutional Layer 1, five feature maps are obtained.

Sub-sampling Layer 1 takes every feature map produced by Convolutional Layer 1 as input, and applies weighted local averaging and sub-sampling to reduce the resolution of the feature maps by half. This serves to blur the feature maps such that the exact locations of the detected features are no longer important. In doing so, the sensitivity of subsequent layers to moderate scale variation and shape distortions are reduced. After Sub-sampling

Layer 1, the number of feature maps (in this case, five) remain the same and the sizes are reduced to 14×14.

The purposes of Convolutional Layer 2 and Sub-sampling Layer 2 are mostly the same with the previous layers, except for two important aspects: first, they take feature maps as input, therefore they extract features of features, namely, the higher-order features (for instance, the co-occurrence of a junction in the center and a sharp corner on the right); second, they produce more feature maps than do the first pair of convolutional/sub-sampling layers, which is empirically shown to benefit the recognition performance [10]. After the second round of convolution and sub-sampling, the number of feature maps is grown to thirty and the sizes are reduced to 5×5, which is too small for further convolution or sub-sampling.

The final two layer are fully-connected to classify the feature maps outputted by Sub-sampling Layer 2 into a human-readable label of symbol categories. The input to Fully-connected Layer is a long vector constructed by concatenating rows of the thirty 5×5 feature maps outputted by Sub-sampling Layer 2. To compute the output of Fully-connected Layer, the input vector is taken a dot-product with the weight vector of each connection in Fully-connected Layer, to which a bias term is added. A similar process is repeated to computer the output of Output Layer. The number of output elements in Output Layer equals the the number of defined symbol categories plus one. The extra category is used to represent the *negative* symbol category, namely, pixel patterns that do not form any defined symbols. The output values are normalized to mimic probabilities. To interpret the final output of the Convolutional Neural Network, the index of the node with the maximum output value is taken as the recognition result. For example, an output of $(0.6, 0.1, 0.1, 0.0, 0.2)$ means that the input image belongs to the first out of four defined symbol categories, while an output of $(0.1, 0.1, 0.1, 0.1, 0.6)$ indicate that the input image belongs none of the four defined categories. This format is also known as the 1-of-n or indicator code.

In sum, the Convolutional Neural Network works as follows: a 32×32 input image is convolved with five sets of 5×5 kernels to obtain five 28×28 feature maps. The five feature maps undergo weighted local averaging and sub-sampling, resulting in five 14×14 feature maps. Afterwards, those five 14×14 feature maps are convolved with 150 (30×5, *i.e.*, five sets of kernels per output feature map) sets of 5×5 kernels to yield thirty 10×10 feature maps. After the second sub-sampling layer, those feature maps are reduced to 5×5. Then they are reshaped to construct a feature vector and taken as input by the last two fully-connected layers, which produces a 1-of-n code to indicate the symbol category of the input.

Training

Stochastic Gradient Descent The Convolutional Neural Network is trained with the back-propagation algorithm [25]. During training, weights are updated to minimize an optimization target called the *loss function*. Here we use a particular formulation of loss function called the *cross-entropy* or *negative log likelihood* loss function [26], which is designed to reflect the discrepancy between the desired output and the actual output on the training set with the current set of weights.

Our method uses a stochastic version of the back-propagation training algorithm known as Stochastic Gradient Descent. A stochastic, approximate gradient is evaluated in term of each training instance and network weights are thereby updated in proportion to those gradient at the frequency of once per training sample. In contrast, the alternative, namely *batch Gradient Descent*, defers the weight update after iterating through the entire training set and calculating the exact gradient of the loss function. The advantages of the stochastic are reliable convergence and speed-up over the batch training [10].

Interested readers are referred to [30] for detailed mathematical derivations of the gradients and the weight update equations in Convolutional Neural Networks.

Obtaining Initial Training Data To initiate the training of the network, a user is supposed to provide a few *labeled* prototypical sketches to the training procedure. A labeled sketch refers to a sketch on which the user has drawn the rectangular bounding box for each symbol and attached descriptive labels (*e.g.*, spring, damper, mass) to the bounding box. This is the only manual intervention required from the user. The rest of the training procedure is autonomous.

In our approach, a training sample is a 32×32 gray-scale image patch with its label encoded as a 1-of-n code. The input size and label encoding is designed to be compatible with the input and output format of the Convolutional Neural Network described in the previous section. For instance, a label of $(0, 1, 0, 0, 0, 0)$ means that the input image belongs to the second category of the six defined categories. A collection of training samples constitute a training set.

Training samples can be generally divided into two super-categories: the positives and the negatives, denoting defined and undefined pixel patterns that are meant to be detected and rejected by the recognizer, respectively.

Each category of positive training samples is obtained by cropping labeled region from the prototypical sketches and resizing the cropped image patch to the size of 32×32, in line with the input size of the Convolutional Neural Network. Because the user has supplied the rectangular bounding box and label of each defined symbol in the prototypical sketches, the creation of the positive training sets is straightforward.

The training samples of the negative category is initially obtained by randomly cropping unlabeled region from the prototypical sketches and resizing the resultant image patches to the size of 32×32. Regions in the prototypical sketches that are with-

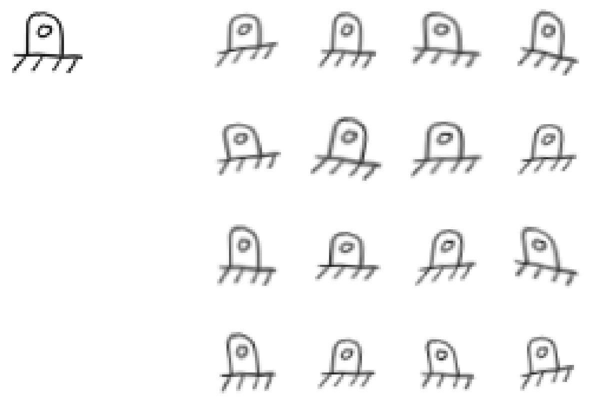


Figure 3. A seed training sample (left) and additional training samples generated from it by random affine transformations (right), aliased due to quantization

out positive labels or has only a small overlap with the positively labeled regions are considered negative regions and cropped as initial samples of the negative training set.

The size of the initial training set is dependent on the number of prototypical sketches that the user provides and the number of defined symbols contained in those sketches. Drawing a large number of sketches and labeling each of them by hand is a tedious task for the user and may incur undesired time and computing cost. Therefore, the size of the initial training set is kept small, usually on the level of dozens of samples per category.

Generating Additional Training Samples Researchers [10, 31] have found that the training of a neural network would generally require a large number of samples in order to yield robust recognition performance that are insensitive to moderate shift, rotation or distortion in the inputs. The larger the training set is, the higher recognition accuracy and robustness can be attained. Therefore, we propose an autonomous procedure to easily generate hundreds or thousands of additional, variant training samples from the dozens of initial training samples without further user intervention.

New training samples in the positive categories are generated by applying a set of random affine transformations to the initial samples. Those transformations are defined to mimic the natural style variations and shape distortions of free-hand sketches and include one or more of the follow: rotation within -2 to 2 degrees, non-uniform shearing with shear factors between -0.2 and 0.2, and/or non-uniform down scaling with scale factors between 0.8 and 1. Figure 3 shows an example of sixteen additional training samples generated from one seed training sample for the ground symbol in sketches of mechanical linkages.

To generate more negative samples and effectively suppress false positives, iterative re-training of the Convolutional Neural Network and iterative sampling of additional negative image patches are performed as follows. The Convolutional Neural Network is first trained with the small-sized, initial training sets and then tested on the prototypical sketches to produce recognitions using the sketch recognition techniques described in the next section. The recognition results are validated against user-supplied labels. Regions of false positives are then identified and ranked according to their probability-like output values. Significant negative samples are added to the negative training set per multiple criteria such as whether the rank order is within top n or whether the output value is above a threshold of P_t. As the negative training set is thus expanded, a commensurate number of additional positive training samples are generated using random affine transformations, so as to keep the number of samples in each category balanced. The Convolutional Neural Network is re-trained with this expanded training set. The above process of re-training and sampling can be re-iterated for several times to ensure that the network is trained with an adequate number of samples. False positive rate and accuracy of the Convolutional Neural Network can be monitored to decide the termination of the iteration.

Sketch Recognition

With a trained Convolutional Neural Network, the recognition of freehand, network-like sketches proceeds as follows.

Sliding Window and The Image Pyramid The input sketch is converted to a gray-scale image and successively down-sampled by a constant factor of 1.25 for several times to produce a series of images with increasingly smaller resolutions than the original one. Those images form a multi-scale representation known as the *image pyramid*, as exemplified in Figure 4. A fixed-size sliding window is applied to each level of the image pyramid with a step size of one in both axis directions and traverse every position within the image. The region inside the window is fed to the Convolutional Neural Network for recognition. Because of the exhaustive manner that the sliding window scans the image at different resolutions, the chance for the Convolutional Neural Network to detect each sketched symbol is maximized. For instance, the left-most body mass in Figure 4, oversized for the sliding window at the original resolution, is gradually shrunk to fit into the sliding window at resolution 4.

Here we have chosen the smallest image of the pyramid to be no smaller than to 56×40 pixels, while the base level of image pyramid is 800×600 pixels. This constructs a 12-level image pyramid. With a 32×32 sliding window, symbols sketched within the range of approximately 32×32 and 480×480 can fit into the sliding window at some level of the pyramid and will be detected if the Convolutional Neural Network is trained to attain a high degree of accuracy.

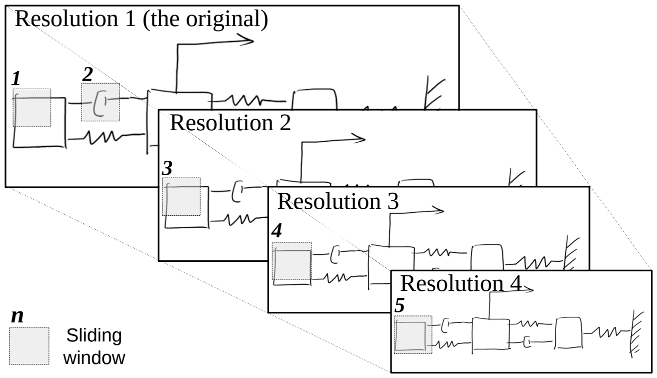


Figure 4. Sliding windows on different levels of the image pyramid

Table 1. The architectural parameters used in evaluations

Parameter	Value
Convolutional kernel size	5×5
Sub-sampling factor	2
Output of Convolutional Layer 1	5 feature maps
Output of Convolutional Layer 2	30 feature maps
Output of Fully-connected Layer	50 nodes
Sliding window size	32×32
Input sketch size	800×600
Level of the image pyramid	12

The exhaustive scanning and recognition amounts to performing sketch parsing and symbol recognition in conjunction, because whenever the Convolutional Neural Network outputs a non-negative label, a symbol is detected inside the sliding window and the bounding box of the symbol can be localized from the current position and the size of the sliding window. This eliminates the need for a separate sketch parsing step and fundamentally differs from previous sketch recognition works [9,21] in which a heuristic parsing step (or segmentation) precedes symbol recognition.

Reducing Computational Redundancy Obviously a large number of computations performed by the Convolutional Neural Network for overlapping input regions are redundant, due to the convolutional nature of the first four layers of the network. For better computational efficiency, the sliding window is actually implemented as follows. The first four layers of the Convolutional Neural Network are applied to each entire level of the image pyramid (rather than the individual image patch inside the sliding window) for once, producing feature maps approximately (due the boundary effect of convolution) four times smaller than the input image. Then a sliding window four times smaller than the original sliding window scans the feature maps and feeds the contents inside the window to the last two layers of the network for recognition. Such operations would produce the same detection results as with the previous approach because of the weight sharing in the convolutional and sub-sampling layers, and it is more efficient to perform all the convolutions only once and then reuse the computation between overlapping sliding windows.

Merging Multiple Detections Some symbols are detected within adjacent sliding windows or at multiple resolutions, because the Convolutional Neural Network is robust to modest shift and scale variation. In such cases, multiple detections outputted by the network are grouped according to their spatial proximity and the amount of overlap between the input windows, and the one with the highest output value is retained and the rest are discarded. After the detection of symbols and, if necessary, some post-processing that analyzes the connectivity between detected symbols, an engineering model can be constructed.

EVALUATIONS

Recognizing Sketches of Mechanical Linkages

Implementation This is a simple, proof-of-concept domain. Symbol definitions include 4 categories: the *ground* symbol drawn in a canonical style as shown in Figure **??**, the *ground* symbol drawn in a simplified way as a triangle, the *pivot* symbol defined as a circle, and the *negative*. Bars that connect pivots and grounds are represented by line segments.

The architectural parameters of the Convolutional Neural Network, the sliding window and the image pyramid are the same as described in foregoing sections and are summarized in Table 1. Unless otherwise specified, the same implementation is used by default throughout subsequent evaluations.

For this domain, the mere detection of ground and pivots is insufficient to extract the intended engineering model. The connectivity between symbols (*i.e.*, the presence of linkages) also needs to be determined. Towards this end, all pixels inside the bounding boxes of detected symbols are masked by background pixels so that the post-recognition canvas only shows the linkages as isolated instances of lines. Then a fast image processing algorithm [32] is utilized to locate the extrema points of each line (*i.e.*, linkage). Each extrema point is assigned to the nearest detected symbol and the connectivity between symbols is thus determined.

With the information regarding the detected symbols and their connectivity, the sketch recognition is accomplished and the intended engineering model can be created by calling the programming interface of the appropriate CAD/CAE packages (*e.g.*, MSC.Adams, MATLAB SimuMechanics), which is out of the scope of this work.

Benchmarks The initial training set includes only two sketches drawn by a tester using a digital tablet. On average approximately 20 symbols per category are drawn and labeled in these sketches as initial samples. During training, the automated procedure responsible for generating additional training samples expands the training set to 300 samples per category after 5 times of re-training, which takes a total of approximately four minutes on a 2.0 GHz CPU.

Once trained, the recognition of a sketch takes approximately 2 to 3 seconds if the input sketch is rendered directly from a digital tablet, or up to 6 seconds if the input is captured from image acquisition devices, during which 2 to 3 seconds are spent on image preprocessing (*e.g.*, edge detection, enhancing).

The test set contains 18 mechanical linkage sketches with various degrees of complexity and a total of 117 symbols. Six of the sketches are drawn with a digital tablet and the rest are drawn on A4 paper with a blue ballpoint pen and then captured using a commodity webcam. All symbols except 3 canonical style ground symbols and 2 pivots are correctly detected and no false positive is produced, amounting to an error rate of 5.1%. Figure 5 shows a correctly recognized test case of a single sketch featuring three linkages: a Peaucellier-Lipkin linkage, a Crank-Rocker four-bar linkage and a Watt's linkage. The style, shape and size variations are noticeable but have not confounded the proposed approach.

Recognizing Sketches of Multi-body Vibratory System

Implementation This section aims to evaluate the generality of our approach, namely, the ease of adapting to a new engineering domain. Also evaluated is the scaling up of the recognition performance, because the defined symbols now fall into 6 categories: *ground*, *body mass*, *spring*, *damper*, *excitation (arrow)* and *negative*. The number of outputs from the Convolutional Neural Network is therefore changed to 6. The rest of the architectural parameters remain unchanged.

By assuming that springs and dampers are not allowed to connect end to end, the connectivity analysis can be skipped, because the intended connectivity can be inferred from the spatial proximity of detected symbols without tracing the lines connecting them [9].

Benchmarks The initial training set includes only one sketch drawn by a tester using a digital tablet. Around 6 to 8 symbols per category are drawn and labeled in this sketch as initial samples. During training, the automated procedure responsible for generating additional training samples expands the training set to 400 per category after 5 times of re-training, which takes a total of approximately 10 minutes on a 2.0 GHz CPU.

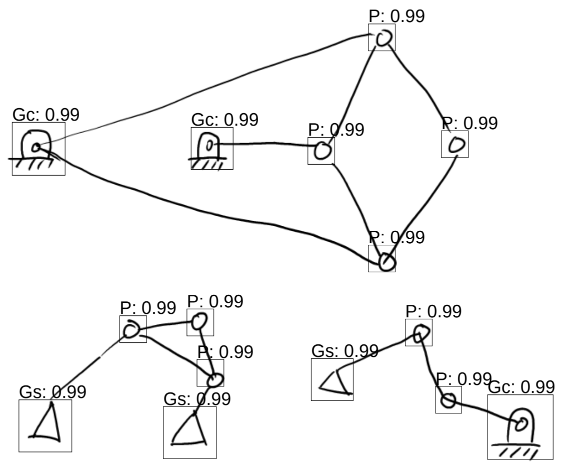


Figure 5. A digital sketch of mechanical linkages that is correctly recognized (Bounding boxes and text annotations indicate the locations and categories of symbols: GC for canonical ground, GS for simplified ground, P for pivot. The real numbers are the probability-like output values.)

Once trained, the recognition of a sketch takes approximately 3 to 4 seconds if the input sketch is rendered directly from a digital tablet, or up to 7 seconds if the input is captured from image acquisition devices, during which 2 to 3 seconds are spent on image preprocessing (*e.g.*, edge detection, enhancing). It can be observed that the scaling up of the number of symbol categories from 4 to 6 leads to increased amount of computation and thereby noticeable longer running time for recognition.

The test set contains 30 images of various multi-body vibration systems with 271 symbols. Twelve of the sketches are drawn with a digital tablet, another fourteen are drawn on A4 paper with a blue ballpoint pen and then captured using a commodity webcam and the remaining four are screen captures of formal, beautified diagrams taken from the electronic versions of relevant technical publications. Figure 6 (a), (b) and (c) demonstrate three exemplary cases of correctly recognized sketches (*e.g.*, a pen-and-paper sketch captured using webcam, a digital sketch and a screenshot of a formal drawing from technical publications). A total of 8 (2.9%) recognition errors occurred, including 5 cases of false negatives (*i.e.*, overly distorted or small symbols that are undetected) and 2 cases of symbol misclassifications (in both cases dampers are misrecognized as body masses) and 1 case of false positive is triggered (Pixels from several nearby symbols are mistaken as a body mass symbol). These error are shown in Figure 6 (d). Overall, the recognition performance is not significantly degraded as the number of symbol categories are doubled. Also note that although the recognizer is trained using images rendered from digital sketches, it is applicable to formal, beautified diagrams as well.

Discussions

Generality and Effectiveness The evaluations suggest that our approach performs reasonably well for both test domains. The Convolutional Neural Network is shown to be robust to shape distortions inherent in freehand sketches and efficient for the exhaustive scanning on input images. The autonomous procedure to expand the training set automates the generation of additional positive and negative training samples, relieving users the burden of drawing and hand labeling many training sketches. The sliding window on the image pyramid enables the detection of symbols drawn at different scales.

Because of the generic nature of this approach, re-targeting a new engineering domain does not require substantial changes to the approach. All that needed is to provide new training samples (*i.e.*, labeled prototypical sketches of the target domain) and retrain the Convolutional Neural Network.

The approach works effectively with various sketching inputs, including both digital ink collected on a tablet PC and traditional sketching media (*e.g.*, pen and paper).

Limitations and Future Work Two major limitations of the current approach are identified. First, the Convolutional Neural Network lacks complete rotation invariance. In other words, it may not recognize an input image that is a significantly rotated version of the training samples. For instance, the resistors of a bridge circuit are not axis-aligned and exhibit such patterns of large rotations. Presently this limitation can be compensated in an *ad hoc* way by including rotated symbols to the training set, which will slow down the training process and may have negative effects on the recognition performance. Future work that formally address this issue may include the polar transformation of input symbols [21] or new mathematical models [12].

Second, the Convolutional Neural Network lacks the capability to exploit contextual cues because recognition is entirely based on the pixel patterns inside the sliding window. Interactions between neighboring windows are not fully taken into account of. The possible negative impact is that the recognizer can be confused by inherently ambiguous symbol definitions. For examples, in the case of handwritten digit recognition, a digit 0 resembles the top or bottom half of a digit 8, and thus a false positive of digit 0 will be produced whenever the top or bottom half of a digit 8 enters the sliding window. None of domains under investigation in this paper have such ambiguous symbol definitions and currently there is no measures taken to compensate for this limitation. In future, relational machine learning model such as Conditional Random Fields [33] may be incorporated to perform recognitions with both local features and contextual features.

Additional future works include the integration with Optical Character Recognizers in order to recognize sketches composed of both shapes and text annotations and the parallelization of the recognition pipeline for faster processing.

SUMMARY AND CONCLUSIONS

We present a general approach to recognize network-like, freehand sketches prevalent in the early design stages. This approach leverages a Convolutional Neural Network as a synergetic detector/recognizer and features an autonomous procedure for the generation of training samples. Evaluations in different domains with different types of input demonstrate the effectiveness of our approach as a general purpose recognizer for network-like sketches.

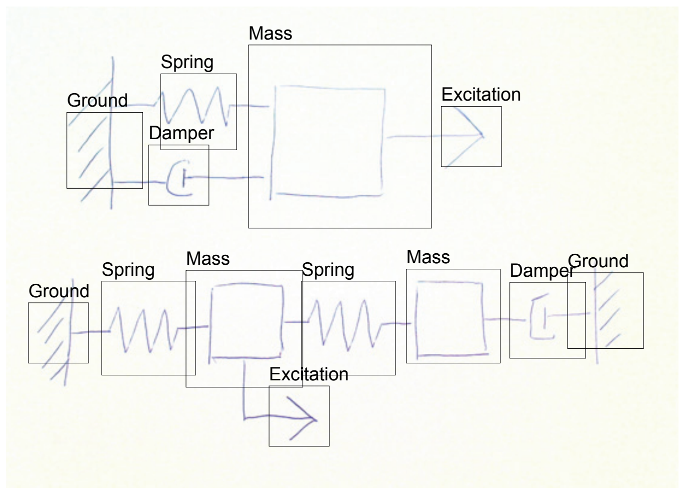
Our approach can be utilized as a building block in various sketch-based application scenarios. If implemented to take full advantage of the computing power of modern multi-core processors and integrated with a gesture recognizer, our approach can be used to build a sketch-based front end for engineering design tools. It may also be used, in conjunction with an Optical Character Recognizer, as a tool for digitizing hard-copy or image-based archival design documents (*e.g.*, legacy patents stored entirely as images that features network-like sketches), which may facilitate content-based information retrieval and design reuse.

Recent work by Silva *et al.* [34] has suggested that pen-based interfaces can be effective instructional tools. Therefore, we believe that our approach also has niches in engineering education. For instance, our software could enhance the lecture experience by allowing an instructor to simply sketch on a conventional blackboard, take a snapshot using a webcam and then simulate and animate the engineering models via a computer and a projector. Similarly, our approach would enable students to model and solve engineering problems graphically and intuitively using pencil and paper.

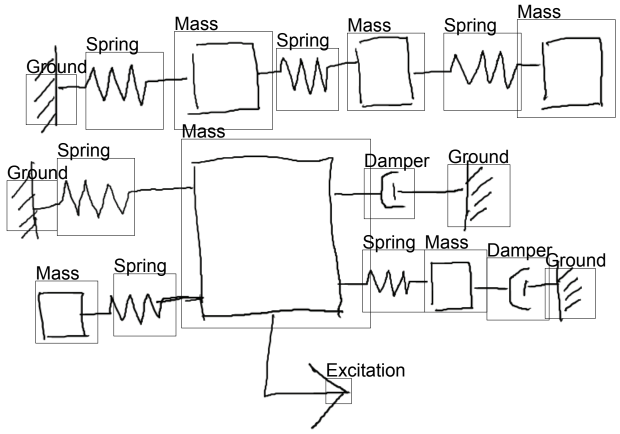
This work is a sequel of a series of past works [9, 21, 23, 35] towards intelligent, sketch-based interface for design. Like Ullman *et al.* [8] and Schutze *et al.* [4], the authors are proponents of integrating the natural and intuitive input modality of sketching with the simulation power of sophisticated CAD/CAE tools. We hope that such integration will gradually instill computational vitality to the relatively lifeless pen and paper and that, in future, the pen will be mightier than the mouse-and-keyboard.

ACKNOWLEDGMENT

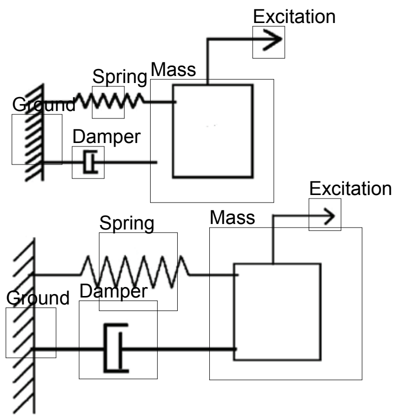
Thanks go to the Jay Jung, Wenyan Wang, Xinhuang Liang and Jin Qian for provided prototypical sketches for training and to Sophia Yang for assistance with capturing pen-and-paper sketches.



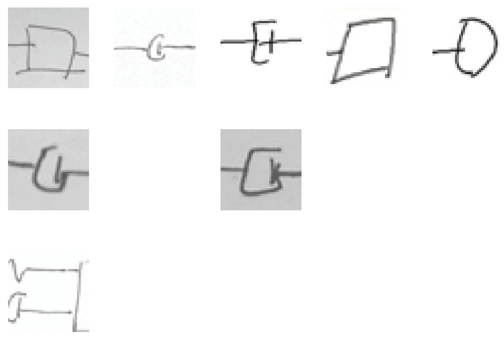
(a)



(b)



(c)



(d)

Figure 6. Exemplary sketches/symbols that are correctly/incorrectly recognized. (a): A correctly recognized pen-and-paper sketch; (b): A correctly recognized digital sketch; (c): A correctly recognized screenshot of a beautified diagram; (d): Pixel patterns that are undetected (first row), misclassified (second row) or trigger false-positives (third row)

REFERENCES

[1] Shpitalni, M., and Lipson, H., 1995. "Classification of sketch strokes and corner detection using conic sections and adaptive clustering". *ASME Journal of Mechanical Design,* **119**, pp. 131–135.

[2] Landay, J. A., and Myers, B. A., 2001. "Sketching interfaces: Toward more human interface design". *IEEE Computer,* **34**(3), pp. 56–64.

[3] Yang, M. C., 2003. "Concept generation and sketching: Correlations with design outcome". In ASME Design Engineering Technical Conferences and Design Theory and Methodology Conference.

[4] Schutze, M., Sachse, P., and Romer, A., 2003. "Support value of sketching in the design process". *Research in Engineering Design,* **14**, pp. 89–97.

[5] Song, S., and Agogino, A. M., 2004. "Insights on designers' sketching activities in new product design teams". In ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference.

[6] Chusilp, P., and Jin, Y., 2006. "Impact of mental iteration on concept generation". *Journal of Mechanical Design,* **128**(1), pp. 14–25.

[7] Yang, M. C., and Cham, J. G., 2007. "An analysis of sketching skill and its role in early stage engineering design".

Journal of Mechanical Design, **129**(5), pp. 476–482.

[8] Ullman, D. G., Wood, S., and Craig, D., 1990. "The importance of drawing in the mechanical design process". *Computer and Graphics,* **14**(2), pp. 263–274.

[9] Kara, L. B., Gennari, L., and Stahovich, T. F., 2008. "A sketch-based tool for analyzing vibratory mechanical systems". *Journal of Mechanical Design,* **130**(10).

[10] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., 1998. "Gradient-based learning applied to document recognition". *Proceedings of the IEEE,* **86**(11), November, pp. 2278–2324.

[11] Garcia, C., and Delakis, M., 2004. "Convolutional face finder: A neural architecture for fast and robust face detection". *IEEE Transactions On Pattern Analysis and Machine intelligence,* **26**(11), pp. 1408–1423.

[12] Osadchy, M., LeCun, Y., and Miller, M., 2007. "Synergistic face detection and pose estimation with energy-based models". *Journal of Machine Learning Research,* **8**, May, pp. 1197–1215.

[13] Ozer, O. F., Ozun, O., Tuzel, C. O., Atalay, V., and Cetin, A. E., 2001. "Vision-based single-stroke character recognition for wearable computing". *IEEE Intelligent Systems and Applications,* **16**(3), pp. 33–37.

[14] Rubine, D., 1991. "Specifying gestures by example". *Computer Graphics,* **25**, pp. 329–337.

[15] Yasuda, H., Takahashi, K., and Matsumoto, T., 2000. "A discrete hmm for online handwriting recognition". *International Journal of Pattern Recognition and Articial Intelligence,* **14**(5), pp. 675–688.

[16] LaViola, J., and Zeleznik, R., 2004. "Mathpad2: A system for the creation and exploration of mathematical sketches". In Proceedings of SIGGRAPH, Vol. 23, pp. 432–440.

[17] Alvarado, C., and Davis, R., 2005. "Dynamically constructed bayes nets for multi-domain sketch understanding". In International Joint Conference on Artificial Intelligence.

[18] Sezgin, T. M., and Davis, R., 2008. "Sketch recognition in interspersed drawings using time-based graphical models". *Computers and Graphics Journal,* **32**(5), pp. 500–510.

[19] Rabiner, L. R., 1989. "A tutorial on hidden markov models and selected applications in speech recognition". In Proceedings of the IEEE, pp. 257–286.

[20] Cowans, P. J., and Szummer, M., 2005. "A graphical model for simultaneous partitioning and labeling". In AI and Statistics.

[21] Kara, L. B., and Stahovich, T. F., 2004. "Hierarchical parsing and recognition of hand-sketched diagrams". In User Interface Software Technology.

[22] Saund, E., 2003. "Finding perceptually closed paths in sketches and drawings". *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **25**, pp. 475–491.

[23] Kara, L. B., and Stahovich, T. F., 2005. "An image-based, trainable symbol recognizer for hand-drawn sketches". *Computers and Graphics,* **29**(4), pp. 501–517.

[24] Notowidigdo, M., and Miller, R. C., 2004. "Off-line sketch interpretation". In AAAI Fall Symposium Series 2004: Making Pen-Based Interaction Intelligent and Natural.

[25] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986. "Learning representations by back-propagating errors". *Nature,* **323**(6088), pp. 533–536.

[26] Bishop, C. M., 1995. *Neural Networks for Pattern Recognition*. Oxford University Press, November.

[27] Simard, P. Y., Steinkraus, D., and Platt, J. C., 2003. "Best practice for convolutional neural networks applied to visual document analysis". In International Conference on Document Analysis and Recogntion, pp. 958–962.

[28] Viola, P., and Jones, M., 2001. "Rapid object detection using a boosted cascade of simple features". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 905–910.

[29] Rowley, H. A., Member, S., Baluja, S., and Kanade, T., 1998. "Neural network-based face detection". *IEEE Transactions on Pattern Analysis and Machine intelligence,* **20**, pp. 23–38.

[30] Bouvrie, J., 2006. "Notes on convolutional neural networks". In MIT CBCL Tech Report, pp. 38–44.

[31] Vapnik, V. N., 1998. *Statistical Learning Theory*. Wiley, New York.

[32] Haralick, R. M., and Shapiro, L. G., 1992. *Computer and Robot Vision*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[33] Quattoni, A., Collins, M., and Darrell, T., 2004. "Conditional random fields for object recognition". In Neural Information Processing Systems, MIT Press, pp. 1097–1104.

[34] Silva, R. d., Bischel, D. T., Lee, W., Peterson, E. J., Calfee, R. C., and Stahovich, T. F., 2007. "Kirchhoff's pen: A pen-based circuit analysis tutor". In Eurographics Workshop on Sketch-Based Interfaces and Modeling.

[35] Gennari, L., Kara, L. B., and Stahovich, T. F., 2005. "Combining geometry and domain knowledge to interpret hand-drawn diagrams". *Computers and Graphics,* **29**(4), pp. 547–562.