2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Improving Machine Vision via Incorporating Expectation-Maximization into Deep Spatio-Temporal Learning

Min JIANG*, Yulong DING*, Ben GOERTZEL†, Zhongqiang HUANG*, Changle ZHOU *,Fei CHAO * ‡
*Department of Cognitive Science and Fujian Provincial Key Laboratory of Brain-like Intelligent Systems
Xiamen University, P. R. China
{minjiang, dozero, fchao}@xmu.edu.cn
†Novamente LLC, 1405 Bernerd Place, Rockville,MD, 20851, USA
ben@goertzel.org

***Abstract*—The Deep Spatio-Temporal Inference Network (DeSTIN) is a deep learning architecture which combines unsupervised learning and Bayesian inference. The original version of DeSTIN incorporates k-means clustering inside each processing node. Here we propose to replace k-means with a more sophisticated algorithm, online EM (Expectation Maximization), and show that this improves DeSTIN's performance on image classification and restoration tasks.**

I. Introduction

Deep learning systems are receiving increasing attention lately, for two good reasons. There is escalating biological evidence [1, 2] supporting these networks as models of neural information processing, especially in the perceptual cortices. Further, an increasing variety of applications [3–5] have shown that deep networks can greatly increase accuracy, within reasonable computational performance bounds. A number of influential and successful deep learning models have been introduced, including Deep Belief Networks (DBN) [6], Stacked Autoencoders (SAE) [7] and Convolutional Neural Nets (CNN) [8]. Here we focus on the DeSTIN deep learning system [9, 10], which differs from the most common deep learning frameworks in multiple respects. DeSTIN bears the closest resemblance to Jeff Hawkins' Hierarchical Temporal Memory (HTM) system [11], but differs from HTM in many ways, some of which (such as the differing mechanisms for top-down feedback) are critical to its successful functionality.

DeSTIN originated as a mathematical model of the conceptual structure and dynamics of the human visual and auditory cortex; and, since its release as open source software, has acquired a variety of sophisticated features. Particular strengths of DeSTIN are its ability to handle temporal data, to learn from a small number of training examples, and to interface with external cognitive software systems [12]. While DeSTIN is in principle intended as a general model of perception processing, prior practical work with DeSTIN has focused on vision processing, and we will continue this focus here.

In an image classification context, it has been shown that use of DeSTIN to generate feature vectors can improve classification performance over standard methods [13]. However the speed of convergence and accuracy of DeSTIN is still far from perfect, particularly when the training examples contain noise. Hence, the DeSTIN research community has recently focused its efforts on techniques to improve DeSTIN's internal mechanisms, so as to create a more powerful machine vision system. The present paper reports one thrust in this direction, the replacement of k-means with online EM within DeSTIN's internal clustering, which appears to have a positive effect on DeSTIN performance.

DeSTIN's dynamics may be divided into feedforward and feedback aspects. In the former, a DeSTIN processing node in a certain layer handles the information passed from the lower layers by unsupervised clustering, and then passes the processed information to the higher level, thus enabling higher layers to encompass increasingly abstract data patterns. In the feedback aspect, each layer of DeSTIN can use feedback from the higher layers, which is regarded as "advice" about how to modify the beliefs in that layer. The advice is inserted into the "belief"data structures on which clustering operates, within each node.

From this capsule description, it is easy to see that improvement of the core clustering algorithm used within each DeSTIN node, has significant potential to improve DeSTIN's overall performance. This observation motivated the work reported here, the goal of which was to obtain a new deep network by replacing the k-means-like clustering method (Starvation Trace Winner-Take-ALL or ST-WTA) standardly used within DeSTIN, with a more sophisticated clustering algorithm, online Expectation-Maximization (EM). Our goals were to increase the accuracy and training speed of the DeSTIN system. Online EM was chosen for this purpose for theoretical reasons; it has been shown by Cappé et al. [14] to have a number of useful properties, e.g. fast and monotone convergence to a stable point of maximum likelihood.

There have been two previous attempts to improve DeSTIN's clustering component. Young, S.R. [15] replaced the WTA with an incremental version. However, his work retained the mechanism of WTA, so the inherent limitations of WTA were not overcome. Rose D.C. [12] evaluated the results of the unsupervised clustering algorithm in every layer of DeSTIN by a pseudo-entropy method and then adjusted the learning rate. We feel our current approach, replacing WTA with online EM, is more promising than putting band-aids on the fundamentally limited WTA algorithm.

There have also been other papers improving and applying the original DeSTIN algorithm, in ways largely orthogonal to the present contribution. Zhang, Y. [13] made interesting use of DeSTIN's recognition ability by combining DeSTIN with KNN, thus achieving good performance. Based on DeSTIN

. The Corresponding author: Fei CHAO, email: fchao@xmu.edu.cn.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

, Goertzel [16] proposed Uniform DeSTIN, the basic idea being that all the nodes on the same level of the DeSTIN hierarchy should share the same library of patterns. That is, all the nodes on the same level should share the same list of centroids. This change helps to deal with large scale data since it is computationally feasible to have a much larger library of patterns utilized by each node.

The paper is organized as follows. In Section II, we will briefly introduce DeSTIN and its training process, including ST-WTA, the clustering approach used in DeSTIN traditionally. In Section III, after a short account of Online EM, we will make a comparison between ST-WTA and Stepwise Online EM from a theoretical point of view, and introduce EM-DeSTIN. Comparative analysis of the ST-WTA and Online EM approaches in the DeSTIN context gives insight into why the standard DeSTIN version's performance declines sharply when noisy training data are present. We will prove that, under certain assumptions, EM-DeSTIN, possesses better properties than the original DeSTIN network.

In Section IV, experimental results will be presented. These experiments were carried out to show the performance of EM-DeSTIN in different aspects, such as accuracy, speed of convergence, and the ability to handle noisy training examples. As well as comparing EM-DeSTIN to traditional DeSTIN on classification and image restoration tasks, we will also compare it to other mainline Deep Networks, e.g. Deep Belief Network (DBN), Convolutional Neural Network (CNN) and Autoencoder. Finally, Section V gives a brief conclusion.

II. Background

In this section, we give a basic overview of DeSTIN, and then briefly analyze the reason why DeSTIN, in its traditional form, cannot deal with noisy data as well as one might hope.

A. DeSTIN

At the macroscopic level, as shown in Fig. 1, the structure of DeSTIN is pyramidal. The pyramid has the following meaning: at the $n-1$-th level, every 2*2 nodes forms a group, and the outputs of the nodes are fed to a node on the n-th level. Similarly, on the n-th level, every 2*2 nodes will form a group, and these nodes are connected to a node on the $n+1$-th level. Each layer of DeSTIN contains a square grid of nodes and every node in the intermediate layers (hidden layers) contains a number of centroids [1]. In the lowest level of DeSTIN, raw data feeds into the corresponding nodes directly. We can consider that DeSTIN abstracts the raw input data to a different extent in the different levels.

The centroids associated with a node depict each possible "state"s for the node. At the beginning, the estimated centroids are initialized to random values. DeSTIN employs a special online algorithm, ST-WTA, to carry out a clustering operation on the input (or observation) at a given time, based on the centroids for that node at that time. It means that an observation o coming from the lower level is assigned to a single estimated centroid χ based on the minimum distance. Given observation o, a belief state s, and belief state of a higher level node a [2], DeSTIN uses the following formula to update the belief state of a node from s to s':

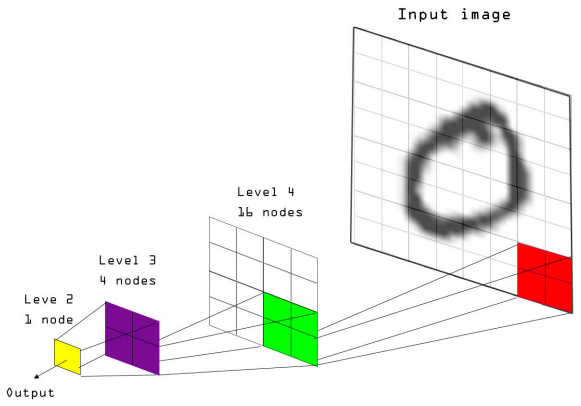

Fig. 1. The construction of DeSTIN

$$b(s'|a) \propto P(o|s')\{\sum_{s\in S} P(s'|s,a)b(s)\} \tag{1}$$

where $P(o|s') = \frac{d_j^{-1}}{\sum_j d_j^{-1}}$ is static pattern similarity. d_j means the distance between the observation and the centroid j. The thinking behind the formula is that: $P(s'|s,a)$ is used to characterize the system dynamics and it modulates the static pattern similarity, so that the belief state inherently captures both spatial and temporal information regarding the raw input data.

The training process of a layer in DeSTIN can be understood roughly as follows: At first, every node obtains observations from the corresponding lower layer nodes, and DeSTIN calculates the "belief state" of the winning centroid based on the clustering results and computes the belief values of the nodes according to the above formula 1, after that DeSTIN will feed the belief values to the corresponding nodes in the higher level. (and recalled that when computing the belief value of a node, DeSTIN will receive information, called "advice", from the higher level node.) Repeating this form bottom to top, and DeSTIN outputs a "belief" value at the top level. Those belief values obtained from higher levels [3] can be regarded as a special kind of feature derived and abstracted from the raw data. The feature could have different uses, for example it can be fed into different classifiers, says KNN or SVM, to carry out pattern recognition.Or they can be fed into a more general cognitive system, to be correlated with other types of inputs such as auditory, verbal or sensory input, or with abstract knowledge obtained from language or structured knowledge bases.

1. How many centroids in a node depends on a balance between resource limitation and representational capacity.

2. In DeSTIN, the belief state of a higher level node is called advice, which is the index of the winning centroid in the higher level node.

3. Depending on various applications, we can take the belief values that come from different numbers of levels as a "feature".

B. ST-WTA and its drawback

The Winner-Take-All approach is a well-known competitive learning algorithm. When the algorithm is used to carry out clustering in DeSTIN, only one centroid – the one which is most similar to the observation, based on minimum distance – is updated, and the rest of the centroids remain unchanged. Despite the fact that WTA is rapid and simple, if the centroids are in poor initial positions, it can give bad results. DeSTIN utilizes a variety of the normal WTA, starvation trace WTA (ST-WTA for short), to circumvent this problem. The key idea of the ST-WTA is depicted by the following formula:

$$d_{xi} = \|x_i - o\|(1 - \psi_i) \tag{2}$$

where x_i is a high-dimensional vector which be used to describe the i-th centroid and o represents an observation, so that d_i is the distance between the i-th centroid and the observation. ψ_i is called starvation factor. When a centroid, for example the i-th centroid, is updated, the starvation factors $\psi_j (j \neq i)$ of the other centroids are increased correspondingly. This was designed to avoid a certain centroid has the sole right to be updated which is caused by the special initial value.

However, all approaches based on the WTA, by nature, are following the logic of gradient descent, which has implications for the convergence speed of DeSTIN, and its robustness with regard to noisy data. We will analyze these aspects in more detail in the next section.

III. EM-DeSTIN

The key innovation described in the current paper, EM-DeSTIN, results from replacing ST-WTA with Online EM, an alternative unsupervised clustering algorithm. In the first section, we introduce the main ideas of Online EM and then analyze why Online EM possesses better convergence speed and robustness than ST-WTA. After that, we described the training steps of EM-DeSTIN in detail.

A. Online EM

A traditional EM algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models. Briefly, EM algorithms alternately carry out two steps: an Expectation (E) step and a Maximization (M) step. The E step computes an expectation of the likelihood estimate by including the latent variables as if they were observed, and the M step, computes the maximum likelihood estimates of the parameter by maximizing the expected likelihood obtained in the E step. The parameters found in the M step are then used to initiate another E step, and the process is repeated.

Traditional EM algorithms always run in an offline mode, which means that the algorithms are given the entire dataset under consideration from the beginning, and are required to output maximum likelihood or MAP estimates of the parameters at hand. However, the offline mode is not always efficiently applicable, so some scholars have conducted research [14, 17, 18] on Online EM algorithms. Unlike offline EM, Online EM algorithms process their observations one-by-one, in a serial fashion.

The first online EM algorithm, incremental online EM (iEM) was proposed by R.M. Neal and G.E. Hinton [18]. The basic idea of the iEM is that it computes a maximum likelihood estimate for each just arrived sample, thus obtaining a parameter estimate s'. After that the difference between s' and s (the parameter estimate based on last observation) is used to adjust the next round of computation. A point worth emphasis is that, although iEm has a fast speed of convergence, it is still has jump-updating problems if there exists noise in the observations.

Cappé and Moulines [14, 19] proposed another online EM algorithm, step-wise online EM (sEM), which takes advantage of stochastic approximation theory. The sEM seeks the estimated value of the parameter in a successive approximation manner, which provides effective performance even under the impact of random noisy data. Algorithm 1 depicts the working process of sEM.

Algorithm 1: Stepwise Online EM(sEM)

$s_i \leftarrow initialization$;
$k \leftarrow 0$;
for *each example* $x^{(i)}, i = 1, \ldots, n$ *in random order* **do**
  E:
  $s'_i \leftarrow \sum_z P(z|x^{(i)}; \theta)\phi(x^{(i)}, z)$;
  $s_i \leftarrow (1 - \eta_k)s_i + \eta_k s'_i$;
  $k \leftarrow k + 1$;
  M:
  $\theta^{i+1} = \arg\max s_i(x)$;
end

The first part of the E step of sEM is similar to classical EM algorithms. It computes the posteriori probability of the hidden variable, s'_i according to the newly arrived sample, and it takes this value as the current estimate of the hidden variable. However, due to the assumed imprecision of parameter estimation for each sample, in the last part of the E step, the sEM updates the parameter as a weighted mean according to the formula $s_i \leftarrow (1 - \eta_k)s_i + \eta_k s'_i$, where $\eta_k = k^{-\alpha}$, where k is the number of updates made so far. In this way, sEM can decrease the effect of an excessively limited number of samples, as well as disturbances caused by noisy samples. Finally, sEM doesn't require solution of the problem of choosing the stepsize η_k, results from the stochastic approximation field prove that if the conditions, $\sum_{i=0}^{\infty} \eta_k = \infty$ and $\sum_{i=0}^{\infty} \eta_k^2 \leq \infty$, are satisfied, the algorithm can be guaranteed to converge to a local optimum. In practical use [20], we generally take $\eta_k = (k+2)^{-\alpha}$, where $0.5 < \alpha \leq 1$.

B. sEM VS ST-WTA

The version of EM-DeSTIN to be proposed here deploys sEM in place of ST-WTA within each DeSTIN node. In order to more fully motivate and understand this substitution, we present a few theoretical results regarding these methods.

ST-WTA updates cluster centers based on the following formula:

$$\begin{cases} j = \arg\min_i ||\mu_i^t - o^t|| \\ \mu_j^{t+1} = \mu_j^t - \alpha(\mu_j^t - o^t) \end{cases} \tag{3}$$

where o^t is the t-th ($t \in \{1, \ldots, m\}$) sample and μ_j^t is the location of the center j when sample o^t at the time t. The first

part of the above formula is used to determine which cluster center to be updated, and the second part describes how to update the center.

Theorem 1 *The ST-WTA is a first-order approximation.*

Proof: Let us define a target function $f(\mu_1, \ldots, \mu_n)$ as follows:

$$f(\mu_1, \ldots, \mu_n) = \sum_{i=1}^{m} \min_{j=1\ldots n} \|\mu_j - o^i\|^2$$

where $\mu_j (1 \leq j \leq n)$ represents the location of the j-th clustering center and o^i means the i-th observation. It sums distance between each observation and the closest clustering center. If we use a first order approximating method to minimize the function, for each μ_j $(1 \leq j \leq n)$, the first partial derivative is:

$$\frac{\partial f}{\partial \mu_j} = 2 \sum_{o^i \in \mu_j} (\mu_j - o^i)$$

where $o^i \in \mu_j$ indicates that μ_j is closest to o^i over all centers. Then μ_j is updated by

$$\mu_j' = \mu_j - \alpha \frac{\partial f}{\partial \mu_j} = \mu_j - \alpha \sum_{o^i \in \mu_j} (\mu_j - o^i) \tag{4}$$

Consider that observations are received one at a time, we can rewrite Eq.4 in an online form:

$$\mu_j' = \mu_j^{t_{k+1}} = \underbrace{\mu_j^{t_1} - \alpha(\mu_j^{t_1} - o^{t_1}) - \cdots - \alpha(\mu_j^{t_{k-1}} - o^{t_{k-1}})}_{\mu_j^{t_k}} - \alpha(\mu_j^{t_k} - o^{t_k}) \tag{5}$$

where k is the number of observations which belong to μ_j, o^{t_i} means that the observation is received at time t_i $(0 < t_1 < t_2 \cdots < t_k)$ and $\mu_j^{t_i}$ represents the center μ_j at time t_i. As a consequence, when an observation o^t is received, we update the closest center μ_j^t by

$$\mu_j^{t+1} = \mu_j^t - \alpha(\mu_j^t - o^t) \tag{6}$$

It can be seen that Eq.6 is equal to Eq.3, which indicates that the ST-WTA is working as a first order approximation. ■

It is easy to get the following theorem from the literature [19].

Theorem 2 *The sEM is a second order approximation.*

Proof: According to algorithm of the sEM (algorithm1), we find that for $n+1$-th sample O_{n+1}, the parameter θ_{n+1} is updated as follows:

$$\theta_{n+1} = \arg\max(Q_n(\theta_n) + \eta_{n+1}(E[\log\{f(X_{n+1}; \theta_n)\}|O_{n+1})] - Q_n(\theta_n)) \tag{7}$$

where $Q_n(\theta_n)$ is the posteriori probability of the hidden variable X_n, which is calculated according to probability density function $f(x; \theta_n)$, and η_n is the relaxation factor of nth sample. Through the first-order Taylor series expansion, the formula 7 is transformed into:

$$\theta_{n+1} = \theta_n + \gamma_{n+1} I^{-1}(\theta_n) \nabla_\theta \log\{f(X_{n+1}; \theta_n)\} + \gamma_{n+1}\rho_{n+1}$$

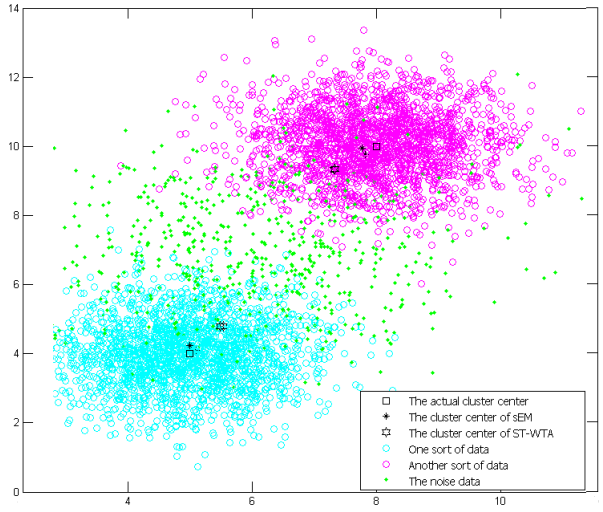

Fig. 2. The experimental result of sEM and ST-WTA

where ρ_{n+1} is the integral remainder term of the above Taylor series, and $\lim_{n\to\infty} \rho_n = 0$. $I(\theta_n)$ is the Fisher-information matrix. When the model is an exponential family model,$I = -\nabla_\theta^2 \log\{f(X_{n+1}; \theta)\}$. So $\theta_{n+1} = \theta_n - \frac{f'(\theta_n)}{f''(\theta_n)}$, which means sEM is a second approximation method. ■

The above theorems show that, in most cases, sEM will have better performance than ST-WTA. An intuitive way to summarize these theorems is to say that sEM uses a quadratic surface to fit the local surface of the current location, whereas ST-WTA just uses a flat surface to fit the local surface. So under ordinary circumstances, sEM should have better robustness and convergence speed than ST-WTA.

To validate the practical relevance of these theoretical notions, we conducted comparative experiments with sEM and ST-WTA in a noisy environment. The result (Fig. 2) verifies our theoretical conclusions.

C. EM-DeSTIN

In this section, we will describe the training and testing process of EM-DeSTIN in more detail. The overall structure of EM-DeSTIN is identical to that of DeSTIN (refer to Fig. 1). In the architecture of EM-DeSTIN, four nodes in a level are assembled into a single group, and the output of the group is associated with the input of a corresponding node in the upper level. There are several centroids in a node, and those centroids can be understood as "distribution of distributions" of the features contained in raw data in some sense. The primary difference between EM-DeSTIN and DeSTIN is the training process; Fig 3 is a brief explanation of the training process of EM-DeSTIN.

The following steps detail the process of training one level in EM-DeSTIN.

Step 1: Some initializations need to be performed before learning begins. This includes determining the number of levels according to input data, determining how

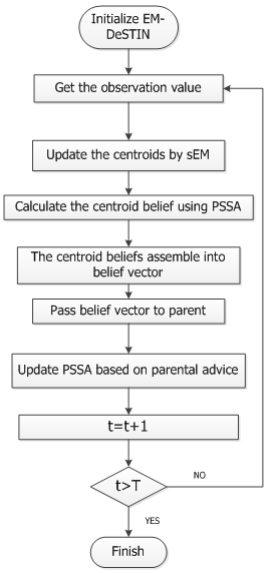

Fig. 3. The Training Process of EM-DeSTIN

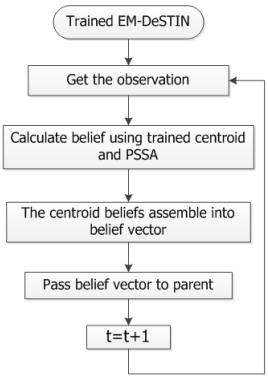

Fig. 4. The Testing Process of EM-DeSTIN

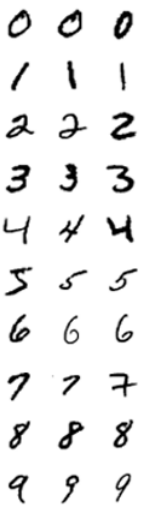

Fig. 5. The sample of dataset

Fig. 6. The sample of noise dataset

many centroids will be included in a node, setting an initial value for each centroid, and initializing the PSSA table.

Step 2: Each node receives a belief vector sent from the nodes in the lower level and advice sent from the parent node.

Step 3: Each centroid in a node can be regarded as a distribution of belief values regarding the contents of the spatiotemporal region referred to by the node. Concretely, the value of a centroid is a vector that has the same dimension as the belief vector a node has received. For all nodes in a level, EM-DeSTIN uses the sEM algorithm to update those centroids.

Step 4: EM-DeSTIN will calculate the belief value for every centroid according to the updated results obtained from the above step, the updated PSSA table, and the "advice" sent from the parent node. Recall that the belief value of a node is a vector which includes all the values of the beliefs belonging only to itself; and that the advice passed from the parent node denotes the ordinal of a centroid in the upper level, which is closest to the input belief value.

Step 5: EM-DeSTIN will combine the belief values of the four nodes in a group into a vector, and output the vector to the corresponding node in the upper level.

Step 6: The PSSA table will be updated according to the results obtained from Step 4 and the parent node's advice.

Fig. 4 describes the testing process of EM-DeSTIN. It resembles the training process. The main difference is that the belief value centroids and PSSA table are not updated during the testing phase.

IV. Experiment

We have implemented EM-DeSTIN as a modification of the standard DeSTIN codebase, and uploaded the code[4] to the open source version of DeSTIN, which is hosted in the Github repository of the OpenCog AI project. Using this implementation, we conducted a three-part program of experiments. In the first part we compared the speed of convergence of EM-DeSTIN and DeSTIN. In the second part, EM-DeSTIN was compared with DeSTIN and other deep networks, including DBN and Stacked Auto-Encoders, in a noise-free situation. In the third part, we tested the performance of these deep networks under noisy environments. Our experiments were run on a HP Z800 machine (96G memory) with Ubuntu 12.04 LTS.

Taking our cue from prior DeSTIN publications, we took the MNIST handwritten Digit database[5] as a benchmark. The images in the MNIST were normalized and fixed-size (28×28 pixels). We randomly selected 5000 pictures from the MNIST as training samples and 10,000 pictures as testing samples. For the training samples, there were 500 pictures with each picture represented by a single digit, such as a 9 or 3. Since there are ten digits, there were 5,000 pictures total in the training samples.

In order to carry out noise tests, two kinds of common noise were added to the training samples: Gaussian noise and Salt-and-Pepper noise. For the Gaussian noise, the mean was 0 and the variance was set to 0.01, 0.06 and 0.1 respectively. For the Salt-and-Pepper noise, the noise density was set to 0.01, 0.1 and 0.5 respectively. Fig. 5 is the original pictures in the MNIST dataset and Fig. 6 shows the pictures after adding Gaussian noise and Salt-and-Pepper noise.

In these experiments, due to the size of the pictures in the MNIST dataset, EM-DeSTIN was set to have 4 layers. Each layer contained 8*8, 4*4, 2*2 and 1 nodes and every node in the different layers included 16, 8, 4 and 1 centroids separately. DeSTIN also had a similar setup.

For testing the convergence speed of EM-DeSTIN and DeSTIN, we recorded the changes of the belief value of the top

4. https://github.com/minjiang/EM-DeSTIN
5. http://yann.lecun.com/exdb/mnist/

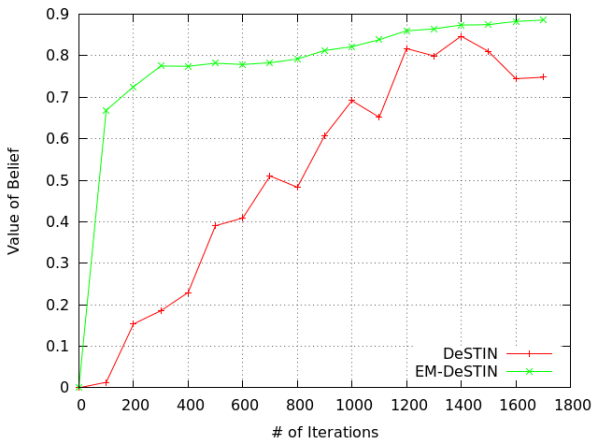

Fig. 7. The speed of convergence: EM-DeSTIN VS DeSTIN

layers when the networks were trained to recognize the digit "0". Fig. 7 describes the result. It is clear that EM-DeSTIN has a smoother converging curved line than DeSTIN, and it reduces the time to reach steady state.

We also tested the robustness of EM-DeSTIN and DeSTIN. We used noisy samples to train the networks. When the networks were stable, we restored the images by using the centroids in the top layers of the two network. The right part of Fig.8 and Fig.9 indicates restored photos from EM-DeSTIN under Gaussian noise and Salt-and-Pepper noise. The left part of the same figure is a collection of restored photos from DeSTIN under the same noisy setting. The experiment shows that we can get a more detailed photo from EM-DeSTIN, and that EM-DeSTIN has better performance.

In the second part of our experiments, we compared the classification accuracy rate of EM-DeSTIN with that of other deep networks. In our DeSTIN classification experiments, we took the outputs of the top two layers of EM-DeSTIN and DeSTIN as input features to train two SVMs, and used the trained SVMs to classify the same testing samples separately. For our testing with other deep networks, we chose three different approaches, Deep Belief Nets (DBN), Convolutional Neural Nets (CNN) and Stacked Autoencoders (SAE) to be equal. The source code of these three other approaches was obtained from DeepLearnToolbox [6].

Fig. 10 shows the changes of the accuracy rates over different iterations. According to the figure, we can see that EM-DeSTIN and CNN both have high accuracy (over 90%) with 2000 iterations.

In the third part of our experiments, we carried out comparison tests under noisy environments. In this experiment, we randomly selected 500 pictures of the digit "7" and added the six different types of noise we mentioned above to the pictures separately. Next, we used the noisy pictures to train DeSTIN and EM-DeSTIN, and then took their outputs to train SVMs. After the trainings were finished, we fed some pictures of the same digit but without noise into the trained SVMs to carry out recognition test. Table I describes the results, which shows that EM-DeSTIN maintained a high accuracy rate in different situations.

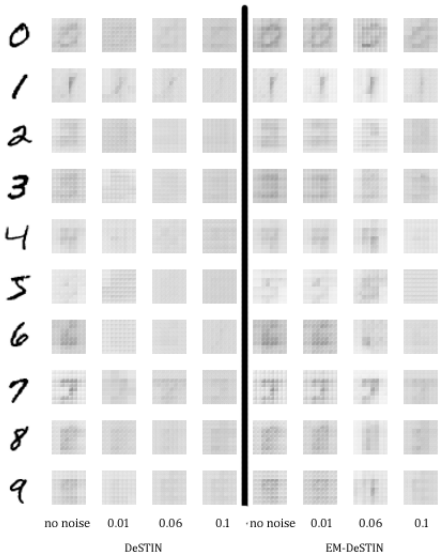

Fig. 8. The figure indicates feature restored photo of DeSTIN and EM-DeSTIN under of the Gaussian noise.

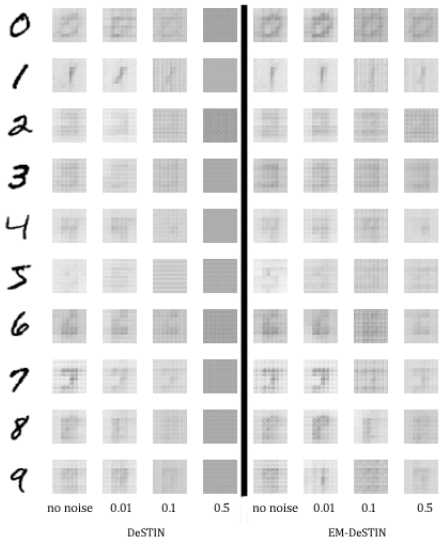

Fig. 9. The figure indicates feature restored photo of DeSTIN and EM-DeSTIN under of the Salt-and-Pepper noise.

The above results show, in terms of accuracy rate, CNN is the winner of this particular contest. However, there are two points requiring some explanation. At first, in our experiment, the training time of EM-DeSTIN is much shorter than CNN. For example, one iteration of training EM-DeSTIN takes 1.4s but CNN spends 20s. Secondly, EM-DeSTIN constitutes a very different sort of general pattern recognition framework than CNN, as discussed in [9, 10, 16]. The experiments given here were not designed to explore the full generality of DeSTIN's applicability or capability, but merely to investigate the impact of substituting k-means with EM within DeSTIN. These results quite clearly suggest that such a substitution is a good idea.

6. https://github.com/rasmusbergpalm/DeepLearnToolbox/

TABLE I. SHOW THE ROBUST TEST OF EM-DESTIN ,UNIFORM DESTIN ,DBN,CNN AND AUTOENCODER(ITERATION 2000)

	Gauss			Salt-Pepper		
	0.01	**0.06**	**0.1**	**0.01**	**0.1**	**0.5**
DBN	0.808	0.635	0.575	0.884	0.727	0.239
CNN	0.972	0.963	0.962	0.979	0.971	0.923
SAE	0.878	0.846	0.833	0.904	0.856	0.778
DeSTIN	0.848	0.716	0.551	0.907	0.851	0.234
EM-DeSTIN	0.961	0.936	0.843	0.950	0.915	0.742

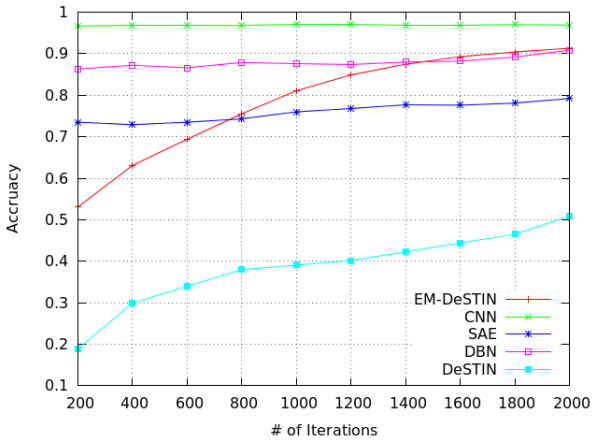

Fig. 10. Accuracy rate: EM-DeSTIN VS other Deep Networks

V. CONCLUSIONS

We have explored a modification of a deep learning architecture, DeSTIN, consisting of a replacement of DeSTIN's internal ST-WTA clustering algorithm with a more sophisticated online EM algorithm. First we performed a theoretical analysis, indicating that online EM is likely to provide superior performance due to providing a second order approximation, whereas ST-WTA is first order. Then we conducted experiments on image classification and restoration, whose results validated the hypothesis suggested by the theoretical analysis: EM-DeSTIN, incorporating online EM, provides superior performance. Comparisons between EM-DeSTIN and the other Deep Neural Networks, including DBN, CNN and Stacked Auto-Encoders, also confirmed the effectiveness of our approach.

The DeSTIN architecture is a work in progress, and many other avenues for improvement remain. However, the replacement of ST-WTA with online EM seems a solid step forward. In future studies, we will pay more attentions on neural-symbolic integration [21]. On the one hand, we will study how to encode a probabilistic or multi-dimensional modal logic program into a deep neural network [22, 23]. On the other hand, we will investigate how to implement an eye-hand coordination [24, 25] or path planning system [26] by using EM-DeSTIN.

VI. ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (No.61003014, 61273338 and 61203336).The authors are very grateful to William Gunther Lauritzen and the anonymous reviewers for their constructive comments that have helped significantly in revising this work.

REFERENCES

[1] T. S. Lee, D. Mumford, R. Romero, and V. A. Lamme, "The role of the primary visual cortex in higher level vision," *Vision research*, vol. 38, no. 15, pp. 2429–2454, 1998.

[2] D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral cortex*, vol. 1, no. 1, pp. 1–47, 1991.

[3] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Large vocabulary continuous speech recognition with context-dependent dbn-hmms," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4688–4691.

[4] V. Nair and G. E. Hinton, "3d object recognition with deep belief nets," in *Advances in Neural Information Processing Systems*, 2009, pp. 1339–1347.

[5] T. Deselaers, S. Hasan, O. Bender, and H. Ney, "A deep learning approach to machine transliteration," in *Proceedings of the Fourth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2009, pp. 233–241.

[6] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[7] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[9] I. Arel, D. Rose, and R. Coop, "Destin: A scalable deep learning architecture with application to high-dimensional robust pattern recognition," in *Proc. AAAI Workshop on Biologically Inspired Cognitive Architectures*, 2009, pp. 1150–1157.

[10] I. Arel, D. Rose, and T. Karnowski, "A deep learning architecture comprising homogeneous cortical circuits for scalable spatiotemporal pattern inference," in *NIPS 2009 Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[11] J. Hawkins and D. George, "Hierarchical temporal memory: Concepts, theory and terminology," *Whitepaper, Numenta Inc*, 2006.

[12] D. C. Rose, I. Arel, T. P. Karnowski, and V. C. Paquit, "Applying deep-layered clustering to mammography image analytics," in *Biomedical Sciences and Engineering Conference (BSEC), 2010*, 2010, pp. 1–4.

[13] Y. Zhang, C. Shang, and Q. Shen, "Interpolating destin features for image classification," in *Computational Intelligence (UKCI), 2013 13th UK Workshop on*. IEEE,

2013, pp. 292–298.
[14] O. Cappé and E. Moulines, "Online expectation-maximization algorithm for latent data models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 71, no. 3, pp. 593–613, 2009.
[15] S. R. Young and I. Arel, "Recurrent online clustering as a spatio-temporal feature extractor in destin," *arXiv preprint arXiv:1301.3385*, 2013.
[16] B. Goertzel, "Modifying the destin perception architecture to enable representationally transparent deep learning," 2012.
[17] D. Li, L. Xu, and E. Goodman, "On-line em variants for multivariate normal mixture model in background learning and moving foreground detection," *Journal of Mathematical Imaging and Vision*, pp. 1–20, 2012.
[18] R. M. Neal and G. E. Hinton, *A view of the EM algorithm that justifies incremental, sparse, and other variants*. Springer, 1998, pp. 355–368.
[19] O. Cappé, "Online expectation-maximisation," *Mixtures: Estimation and Applications*, pp. 1–53, 2011.
[20] P. Liang and D. Klein, "Online em for unsupervised models," in *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*. Association for Computational Linguistics, 2009, pp. 611–619.
[21] M. Jiang, C. Zhou, and S. Chen, "Embodied concept formation and reasoning via neural-symbolic integration," *Neurocomputing*, vol. 74, no. 1, pp. 113–120, 2010.
[22] M. Jiang, Y. Yu, F. Chao, M. Shi, and C. Zhou, "A connectionist model for 2-dimensional modal logic," in *Computational Intelligence for Human-like Intelligence (CIHLI), 2013 IEEE Symposium on*. IEEE, 2013, pp. 54–59.
[23] M. Jiang, J. Xu, and F. Liu, "Uncertain formal concept based on 3-valued lukasiewicz logic," in *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*. IEEE, 2010, pp. 1–4.
[24] F. Chao, Z. Wang, C. Shang, Q. Meng, M. Jiang, C. Zhou, and Q. Shen, "A developmental approach to robotic pointing via human-robot interaction," *Information Sciences*, 2014.
[25] F. Chao, L. Hu, M. Shi, and M. Jiang, "Robotic 3d reaching through a development-driven double neural network architecture," in *Knowledge Engineering and Management*. Springer, 2011, pp. 179–184.
[26] M. Jiang, Y. Yu, X. Liu, F. Zhang, and Q. Hong, "Fuzzy neural network based dynamic path planning," in *Machine Learning and Cybernetics (ICMLC), 2012 International Conference on*, vol. 1. IEEE, 2012, pp. 326–330.