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

Interpolating Deep Spatio-Temporal Inference Network Features for Image Classification

Yongfeng Zhang, Changjing Shang and Qiang Shen
Department of Computer Science
Institute of Mathematics, Physics and Computer Science
Aberystwyth University, UK
Email: {yoz1, cns, qqs}@aber.ac.uk

***Abstract*—This paper presents a novel approach for image classification, by integrating the concepts of deep machine learning and feature interpolation. In particular, a recently introduced learning architecture, the Deep Spatio-Temporal Inference Network (DeSTIN) [1] is employed to perform feature extraction for support vector machine (SVM) based image classification. Linear interpolation and Newton polynomial interpolation are each applied to support the classification. This approach converts feature sets of an originally low-dimensionality into those of a significantly higher dimensionality while gaining overall computational simplification. The work is tested against the popular MNIST dataset of handwritten digits [2]. Experimental results indicate that the proposed approach is highly promising.**

I. Introduction

Deep machine learning is an emerging framework for dealing with complex data in a hierarchical fashion which draws inspiration from biological sources [3], [4], [5], [6], [7]. The use of multiple levels of operations can greatly simplify the computational load of a learning architecture, provided that it can be successfully trained and optimised for a given application such as image classification. One fairly recent deep learning architecture worthy of noting is the Deep Spatio-Temporal Inference Network (DeSTIN) [1], [8], [9], which is to be employed in the present work.

The key challenge to successfully build an image classifier is to extract and use informative features from given images [10], [11]. The extracted features minimise the computational complexity to a certain extent. Unfortunately, the technique underlying DeSTIN itself introduces significant computation, which may well offset the potential benefit regarding the efficiency in the implementation of the entire feature extraction process. An alternative approach is to employ interpolation to produce a more informative feature set, which helps improve the representation of the underlying images to be classified with minimal computational overheads, thereby increasing the classification accuracy without sacrificing the efficiency. This paper presents a novel discriminating deep learning architecture that combines DeSTIN with interpolation, in an attempt to implement the above idea. This architecture, which is referred to DESTINI hereafter, leads to a highly scalable modelling system which is capable of effectively extracting image features efficiently.

To perform the actual feature-based image classification task, Support Vector Machines (SVMs) [12] are employed, by mapping input feature vectors onto the underlying image class labels. This is due to the recognition of their high generalisation performance in complex data sets [12], [13]. Such a classifier seeks to find the optimal separating hyperplane amongst different classes, by focusing on those training points (named support vectors) which are placed at the edge of the underlying feature vectors and whose removal would change the solution to be found [11]. The fact that SVM-based classifiers typically involve a fair amount of computation makes it even more significant to gain maximal efficiency from the feature extraction process.

The rest of this paper is organised as follows. Section II introduces the MNIST dataset of handwritten digits that will be used as the illustrative problem for the subsequent work presented herein. Section III outlines the background of DeSTIN. Section IV details the proposed DESTINI approach, including both linear interpolation and Newton interpolation mechanisms, supported with worked examples. Section V gives a brief overview of SVMs that are used to implement the image classifiers for experimental evaluation. Section VI shows the experimental results, supported by comparative studies, demonstrating that the framework is highly promising. The paper is concluded in Section VII, including a discussion about further research.

II. MNIST Dataset of Handwritten Digits

In this work, DESTINI is tested on a popular problem, the MNIST data set of handwritten digits, which is widely used for various machine learning algorithms. The dataset consists of 60,000 training images and 10,000 test images. Each hand-written digit was originally extracted from a larger set made available by NIST [2], before being size-normalised and centred in a fixed-size image (28×28 pixels). Each image is labelled by one of 10 classes corresponding to the numbers 0-9. Fig. 1. shows a tiny part of the MNIST database, and Fig. 2. presents a sample image of digit 4. The application problem for this research is to develop an image classifier that can detect and recognise different digital numbers from such a given hand-written figure.

III. Deep Spatio-Temporal Inference Network

A Deep Spatio-Temporal Inference Network (DeSTIN) [1] offers an emerging framework for dealing with complex high-dimensional data. Fig. 3. shows the generic architecture of such a network. It contains a hierarchy of layers whereby each layer consists of multiple instantiations of an identical cortical circuit or node. Each node is tasked with observing

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

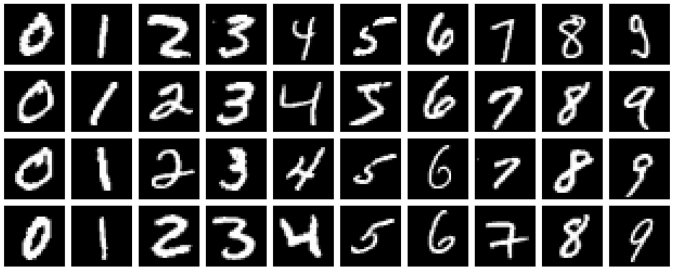

Fig. 1. Example images in MNIST database

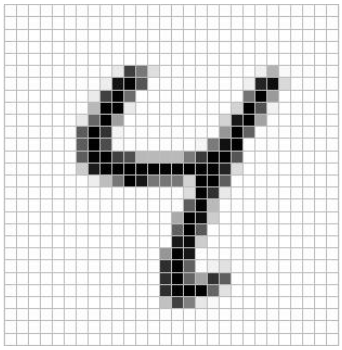

Fig. 2. Example image of digit 4

and learning the sequences of data that are presented to it. The lowest layer of the hierarchy processes the input data (that may be temporally varying) to the network, such as image pixels, and over time continuously constructs a belief state that attempts to characterise the data sequences received. The second layer, and all those above it receive as input the belief states of the nodes at their respective lower layers, and attempt to construct belief states that capture any interesting regularities in the data. The resulting outputs can be fed to a classifier, such as an SVM as extracted features for subsequent processing (e.g., classification).

For completion the theoretical foundation of DeSTIN is outlined below; further details can be found in [1], [8], [9]. Let the current observation be denoted by o, the current sequence by s, the new (updated) sequence by $s^{'}$, and the parent node sequence by $s^{''}$. Also, let b denote the belief state of s, $b^{'}$ denote a new (updated) belief state of $s^{'}$, and c denote the belief state of $s^{''}$, such that

$$b^{'} = P_r(s^{'}|o,b,c) = \frac{P_r(s^{'},o,b,c)}{P_r(o,b,c)} \tag{1}$$

or alternatively

$$b^{'} = \frac{P_r(o|s^{'},b,c)P_r(s^{'}|b,c)P_r(b,c)}{P_r(o|b,c)P_r(b,c)} \tag{2}$$

Under the assumption that observations depend only on the underlying true state, or $P_r(o|s^{'},b,c) = P_r(o|s^{'})$, the above can be further simplified such that

$$b^{'} = \frac{P_r(o|s^{'})P_r(s^{'}|b,c)}{P_r(o|b,c)} \tag{3}$$

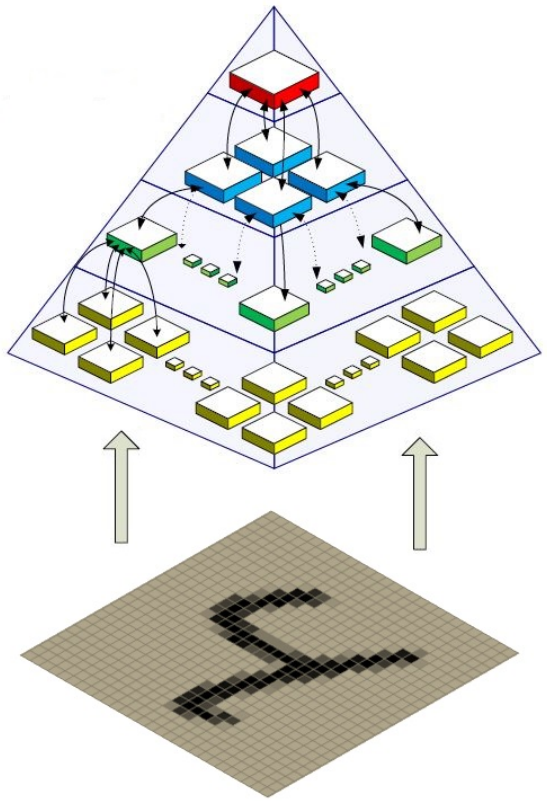

Fig. 3. Topological architecture of DeSTIN

where $P_r(s^{'}|b,c) = \sum_{s\in S} P_r(s^{'}|s,c)b$, yielding the belief update rule

$$b^{'} = \frac{P_r(o|s^{'})\sum_{s\in S} P_r(s^{'}|s,c)b}{\sum_{s^{''}\in S} P_r(o|s^{''})\sum_{s\in S} P_r(s^{''}|s,c)b} \tag{4}$$

where S denotes the sequence set (i.e., the so-called belief dimension) and the denominator term is a normalisation factor.

One interpretation of Eqn. (4) is that the belief state inherently captures both spatial and temporal information. Note that there are two core constructs in it to be learned: $P_r(o|s^{'})$ and $P_r(s^{'}|s,c)$. The former can be learned via online clustering while the latter is learned by trial and error through adjusting the parameters with each transition from s to $s^{'}$ given c.

To compute $P_r(o|s^{'})$, an online clustering algorithm is utilised with a finite set of centroids that each represent the possible state s of a certain node. The cardinality of the centroid set is a predefined parameter, which is preferably larger than necessary to provide each node with sufficient representational capacity in modelling the observed images. Let n be the cardinality of the centroid set defined for the top layer, then the output of DeSTIN can be represented by a vector of features, say, $V_{original} = (O_1, \cdots, O_n)^T$ that jointly represent the current input of the network.

IV. DESTINI

For the present application, the input to a DeSTIN is an image, represented as a high-dimensional signal:

$$M_{n\times m}=\begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0m} \\ a_{10} & a_{11} & \cdots & a_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{nm} \end{pmatrix} \tag{5}$$

After propagating through a DeSTIN, the original image is converted to a vector of features, which as indicated previously, can be represented as

$$V_{original}=(O_1,\cdots,O_n)^T \tag{6}$$

To perform effective image classification such a vector is expected to have a significant dimensionality. However, this means that the number of layers and/or the number of nodes required in each layer is also significant and hence will incur considerable computation. Thus, a trade off between subsequent classification effectiveness and computational effort in converting the original image into the feature vector is needed. This observation leads to the present development, by applying simple interpolation methods to create additional elements in an artificially expanded feature vector. That is, interpolation can be applied to $V_{original}$ in order to generate additional vector elements:

$$V_{additional}=(l_1,\cdots,l_m)^T \tag{7}$$

resulting in a final vector of features that jointly represent the original image:

$$V_D=(O_1,\cdots,O_n,l_1,\cdots,l_m)^T \tag{8}$$

A. Interpolation methods

1) Linear interpolation: The simplest possible interpolation is to linearly set $l_k=i_k\ (1\le k\le m)$ with i_k being local sums $O_j+O_{j+1}, 1\le j\le n-1$, which is represented as $O_{j+(j+1)}$. On top of these local sums, the global sum $sum=\sum_{i=1}^{n}O_i$ can also be obtained in a straightforward manner. In so doing, the dimensionality of the additional feature vector can be increased to a number that is up to n. Obviously, such linear addition-based interpolation involves little computation.

Consider that if $m=2$, which means that additional vectors of each containing two elements, $V_{additional}=(i_1,i_2)^T$ can be generated from the original vector produced by DeSTIN. In particular, i_1 may be set to be one element of the local sum set $\{O_{1+2},O_{2+3},\cdots,O_{(n-1)+n}\}$ or the global sum $sum=\sum_{i=1}^{n}O_i$. So there are n options, after choosing the value of i_1, the size of the optional set is reduced to $n-1$. For i_2, there are $n-1$ options. So if $m=2$, there are C_n^2 different types of possible combination of the elements contained within the optional set to construct an additional feature vector.

In general, if $m=j,\ j\in\{2,3,...,n\}$, then j-dimensional vectors can be generated, with $i_k,(1\le k\le m)$ representing the local sum $O_{1+2},O_{2+3},\cdots,O_{(j-1)+j},\cdots,O_{(n-1)+n}$ or the global sum $sum=\sum_{i=1}^{n}O_i$. There are C_n^j different options of possible combination to form such additional vectors. Thus, if necessary, a total of $C_n^2+C_n^3+\cdots+C_n^{n-1}+C_n^n$ possible combinations can be generated when $V_{original}$ is an n-dimensional vector.

2) Newton interpolation: In addition to linear interpolation (that is computationally least complex), another possible means to create artificial features is through Newton interpolation. The basic idea is that given the values of $V_{original}=(O_1,\cdots,O_n)^T$, a set of $k+1$ data points $P=\{(x_0,y_0),(x_1,y_1),\ldots,(x_k,y_k)\},(k\ge n)$ are created, such that the values of $\{x_0,x_1,\ldots,x_k\}$ are set to consecutively integer values, and the elements of $V_{original}$ are assigned as the values to a subset of $\{y_0,y_1,\ldots,y_k\}$, with the other $(k+1-n)$ missing values of this set to be generated via interpolation. To distinguish linearly interpolated features from those produced by the Newton method, the latter are denoted by $I_j\ (1\le j\le k+1-n=m)$ below.

Formally, the Newton interpolation polynomial is of the form:

$$N(x)=\sum_{j=0}^{k}a_jn_j\left(x\right) \tag{9}$$

with the so-called Newton basis polynomials defined by

$$n_j\left(x\right)=\prod_{i=0}^{j-1}\left(x-x_i\right) \tag{10}$$

and the coefficients by

$$a_j=[y_0,\ldots,y_j] \tag{11}$$

where $[y_0,\ldots,y_j]$ denotes divided differences. Given $k+1$ data points $(x_0,y_0),(x_1,y_1),\ldots,(x_k,y_k)$, the divided differences are defined as:

$$[y_v]=y_v, v\in\{0,\ldots,k\} \tag{12}$$

$$[y_v,\ldots,y_{v+j}]=\frac{[y_{v+1},\ldots,y_{v+j}]-[y_v,\ldots,y_{v+j-1}]}{x_{v+j}-x_v} \tag{13}$$

where $v\in\{0,\ldots,k-j\}, j\in\{1,\ldots,k\}$.

Now, consider the specific case of $m=1$, where additional 1-dimensional vectors $V_{additional}=(I_1)^T$ can be generated from the original vectors produced by DeSTIN. In this case, the dimensionality of data points is $n+1$. First, $\{x_0,x_1,\ldots,x_k\}$ are set to integer values consecutively; without losing generality, these can be represented as $\{x_0=1,x_1=2,\ldots,x_{k-1}=n,x_k=n+1\}$. Then, O_1 is set to becoming the value of $y_i,i\in\{0,1,2,...,k\}$. So, there are $n+1$ options for such an assignment. After choosing the value of y_i, the size of the optional set is reduced to n. For O_2, there are n options, and so on. Thus, if $m=1$, there are C_{n+1}^1 different kinds of possible combination of artificial features to form an additional feature vector.

In general, similar to the linear interpolation case, if $m=j,\ j\in\{2,3,...,n+1\}$, then j-dimensional vectors can be generated, with $I_k\ (1\le k\le m)$ representing the Newton interpolated feature value. There are C_{n+1}^j different options of possible combination to construct the additional vector. Together, if required, a total of $C_{n+1}^1+C_{n+1}^2+\cdots+C_{n+1}^n+C_{n+1}^{n+1}$ possible combinations can be generated when $V_{original}$ is an n-dimensional vector.

B. The DESTINI algorithm

A given image for classification is first processed by a DeSTIN of a low complexity. The resulting low-dimensional feature vector of the (relatively simple) DeSTIN is then translated into a feature set of a higher dimensionality and hence of potentially more discriminating power, through interpolation that leads to Eqn. 8. Reflecting the above-introduced straightforward interpolation mechanisms, l_j ($j \in \{1, 2, ..., m\}$) in V_D stands for i_j in linear interpolation and for I_j in Newton interpolation. The feature vectors so produced that consist of the original DeSTIN outputs and interpolated additional artificially created features are regarded as the returns of the DESTINI system, thereby computationally integrating DeSTIN and feature interpolation. From this, any of the generated V_D can be fed to an SVM (which acts as the image classifier, see below) for the purpose of robust image classification. This integrated use of DeSTIN and interpolation significantly increases the feature vector dimensionality without incurring much additional computation.

The working of DESTINI can be summarised as shown in Algorithm 1. Given a set of training data, the algorithm starts by assuming that the (initial) brief states b are set to default values (line 1). As the main body of the algorithm, it then runs an iteration loop (lines 2-6), in which an image is assigned to a matrix $M_{n\times m}$ (line 2), and then for each layer, an update is carried out to refine the brief state (line 3). Having updated all the layers, set the original features $V_{original}$ to be the output of DeSTIN (line), and interpolate it to build V_D (line 5). Repeat the loop until no image remaining to be processed (line 6). What is returned by this algorithm is the V_D constructed from a set of original features $M_{n\times m}$ (line 7).

Algorithm 1: The DESTINI Algorithm

(1) Initialise b, with all unclassified images to buffer
(2) Load an image to $M_{n\times m}$
(3) For each layer of DeSTIN, update b to $b^{'}$
(4) Set $V_{original}$ to the output of DeSTIN $(O_1, \cdots, O_n)^T$
(5) Interpolate $V_{original}$ to derive $V_D = (O_1, \cdots, O_n, l_1, \cdots, l_m)^T$
(6) If every image has been processed go to the next, else go to (2)
(7) Return $V_D = (O_1, \cdots, O_n, l_1, \cdots, l_m)^T$

The time complexity of DESTINI is mainly determined by two aspects: the time complexity of generating $V_{original} = (O_1, \cdots, O_n)^T$ and that of computing $V_{additional} = (i_1, \cdots, i_m)^T, (2 \leq m \leq n)$. Consider that the core calculation for $V_{original}$ is the application of the Euclidean distance metric, so the time complexity of computing $V_{original}$ is $O(n^2)$. If $V_{additional}$ is generated by linear interpolation, the time complexity for $V_{additional}$ is $O(m)$. Together the time complexity of producing $V_D = (O_1, \cdots, O_n, i_1, \cdots, i_m)^T, (2 \leq m \leq n)$, is $O(n^2)+O(m)$. If however, $V_{additional}$ is generated by Newton interpolation, the time complexity for obtaining $V_{additional}$ is $O(m^2)$. Together the time complexity of producing V_D is $O(n^2) + O(m^2)$. Yet, to generate an original feature vector of the same dimensionality, which is $(m + n)$, the time complexity of DeSTIN is $O((m + n)^2)$. Because $2 \leq m \leq n$, given the same dimensionality, it is clearly more efficient for DESTINI to generate V_D.

C. Generic worked examples

To illustrate the basic idea of DESTINI, consider the case where the output of DeSTIN is a 3-dimensional vector:

$$V_{original} = (O_1, O_2, O_3)^T \tag{14}$$

Thus, the following additional interpolated vectors

$$V_{additional} = (i_1, \cdots, i_m)^T, 2 \leq m \leq 3 \tag{15}$$

can be generated by linear interpolation, plus the global feature $sum = \sum_{i=1}^{3} O_i$.

If $m = 2$, which means additional vectors of 2 dimensions $V_{additional} = (i_1, i_2)^T$ can be generated from the original vector. So $i_k, k \in \{1, 2\}$, represents O_{1+2}, O_{2+3} or $sum = \sum_{i=1}^{3} O_i$, and there are C_3^2 =3 different kinds of combination:

$$(i_1 = O_{1+2}, i_2 = O_{2+3})^T \tag{16}$$

$$(i_1 = O_{1+2}, i_2 = sum)^T \tag{17}$$

$$(i_1 = O_{2+3}, i_2 = sum)^T \tag{18}$$

Consequently, the combined V_D may be either of the following three:

$$(O_1, O_2, O_3, O_{1+2}, O_{2+3})^T \tag{19}$$

$$(O_1, O_2, O_3, O_{1+2}, sum)^T \tag{20}$$

$$(O_1, O_2, O_3, O_{2+3}, sum)^T \tag{21}$$

If $m = 3$, then a 3-dimensional additional feature vector consisting of the following can be produced: $i_1 = O_{1+2}$, $i_2 = O_{2+3}$, $i_3 = sum$. Thus, V_D represents

$$V_D = (O_1, O_2, O_3, O_{1+2}, O_{2+3}, sum)^T \tag{22}$$

Together, out of the original 3-dimensional feature vector, $C_3^2 + C_3^3 = 4$ combinations can be created to act as the artificially generated additional features.

Now, consider an example using Newton interpolation with the same 3-dimensional original feature vector. In this case, the following additional interpolated vectors can be generated:

$$V_{additional} = (I_1, \cdots, I_m)^T, 1 \leq m \leq 4 \tag{23}$$

If $m = 1$, which means that additional vectors of 1 dimension $V_{additional} = (I_1)^T$ are required to be generated. Thus, the set of data points is $P = \{(1, y_0), (2, y_1), (3, y_2), (4, y_3)\}$, which may be either of:

$$\{(1, I_1), (2, O_1), (3, O_2), (4, O_3)\} \tag{24}$$

$$\{(1, O_1), (2, I_1), (3, O_2), (4, O_3)\} \tag{25}$$

$$\{(1, O_1), (2, O_2), (3, I_1), (4, O_3)\} \tag{26}$$

$$\{(1, O_1), (2, O_2), (3, O_3), (4, I_1)\} \tag{27}$$

There are therefore C_4^1 = 4 different kinds of combination.

If $m = 2$, additional vectors of 2 dimensions $V_{additional} = (I_1, I_2)^T$ can be generated. The set of artificially created data

points is $P = \{(1, y_0), (2, y_1), (3, y_2), (4, y_3), (5, y_4)\}$, which may be either of:

$$\{(1, I_1), (2, O_1), (3, I_2), (4, O_2), (5, O_3)\} \quad (28)$$

$$\{(1, I_1), (2, O_1), (3, O_2), (4, I_2), (5, O_3)\} \quad (29)$$

$$\{(1, I_1), (2, O_1), (3, O_2), (4, O_3), (5, I_2)\} \quad (30)$$

$$\{(1, O_1), (2, I_1), (3, O_2), (4, I_2), (5, O_3)\} \quad (31)$$

$$\{(1, O_1), (2, I_1), (3, O_2), (4, O_3), (5, I_2)\} \quad (32)$$

$$\{(1, O_1), (2, O_2), (3, I_1), (4, O_3), (5, I_2)\} \quad (33)$$

forming the C_4^2 = 6 different kinds of combination.

If $m = 3$, additional feature vectors of 3 dimensions $V_{additional} = (I_1, I_2, I_3)^T$ can be created. The set of data points is $P = \{(1, y_0), (2, y_1), (3, y_2), (4, y_3), (5, y_4), (6, y_5)\}$, which may be either of the C_4^3 =4 different kinds of combination:

$$\{(1, I_1), (2, O_1), (3, I_2), (4, O_2), (5, I_3), (6, O_3)\} \quad (34)$$

$$\{(1, I_1), (2, O_1), (3, I_2), (4, O_2), (5, O_3), (6, I_3)\} \quad (35)$$

$$\{(1, I_1), (2, O_1), (3, O_2), (4, I_2), (5, O_3), (6, I_3)\} \quad (36)$$

$$\{(1, O_1), (2, I_1), (3, O_2), (4, I_2), (5, O_3), (6, I_3)\} \quad (37)$$

If $m = 4$, which means that a 4-dimensional additional vector $V_{additional} = (I_1, I_2, I_3, I_4)^T$ can be generated, with the set of artificial data points being $P = \{(1, y_0), (2, y_1), (3, y_2), (4, y_3), (5, y_4), (6, y_5), (7, y_6)\}$:

$$\{(1, I_1), (2, O_1), (3, I_2), (4, O_2), (5, I_3), (6, O_3), (7, I_4)\} \quad (38)$$

Altogether, $C_4^1 + C_4^2 + C_4^3 + C_4^4 = 15$ combinations can be produced from the original 3-dimensional feature vector.

V. Support Vector Machines

As indicated previously, Support Vector Machines (SVMs) [12] are herein used to implement the task of image classification. SVMs work by mapping input feature vectors onto the underlying image class labels. Such a classifier seeks to find the optimal separating hyperplane amongst different classes, by focusing on those training points (named support vectors) which are placed at the edge of the underlying feature vectors and whose removal will change the solution to be found.

More formally, SVMs construct a hyperplane in a space of a dimensionality higher than that of the original, which is then used for classification (or for other tasks such as regression and prediction). The underlying intuition is that by mapping the original data space onto a much higher-dimensional space, the class separation between data points will become easier in that space. SVMs use a specific mapping such that the cross products of data points in the larger space are defined in terms of a kernel function [14] which is selected to suit the given problem. In so doing, the cross products may be computed in terms of the variables in the original space, thereby minimising computational effort. In particular, a hyperplane in the higher dimensional space is defined as the set of points whose inner product with any vector in that space is constant. A good hyperplane is learned over a training process such that the resulting hyperplane has the largest distance to the nearest training data points of any given class. This is in order to increase the discriminating power of the trained classifier.

In the following, the Radial Basis function (RBF) kernel is adopted to implement the SVM-based classifiers, and the sequential minimal optimisation algorithm of [15] is used to train the SVMs. Note that in implementation, as the default settings of SVMs are taken from the LIBSVM [16], no hyper parameters of individual SVMs are further tuned in order to give equal footings in results comparison. Detailed SVM learning mechanism is beyond the scope of this paper, but can be found in the literature (e.g., [12], [17]).

VI. Experimental results

The topology of the underlying DeSTIN chosen to perform this experimental investigation consists of 4 layers, with the first layer hosting 8×8 nodes, each receiving a non-overlapping 4×4 patch of a given image that is vectorised into a 16 element input. At each subsequent layer, there are one quarter of the number of the nodes within the preceding layer, such that layer two hosts 4×4 nodes, layer three 2×2 nodes, and layer four just one node as depicted in Fig. 3.

The task of the experiments is to classify images into one of the 10 classes corresponding to the digits 0-9. As indicated previously, the MNIST database [2] which contains 60,000 training images and 10,000 testing images is used to facilitate the experimental comparison below. The dimensionalities of the belief states for layers one, two and three are set to 25, 16 and 12, respectively. In order to give the top layer sufficient representational capacity, the dimensionality of its centroid set is ranged from 3 to 6. Thus, the output of the DeSTIN can be represented as $V_{original} = (O_1, \cdots, O_n)^T, 3 \leq n \leq 6$. Note that as the layer index increases, the information compression rate increases, as reflected by the corresponding reduced dimensionality of the belief space.

A. Use of DESTINI features with linear interpolation

TABLE I. Accuracy using additional interpolated features based on a 3-dimensional original vector

Feature list	Accuracy
O_1, O_2, O_3	86.73%
$O_1, O_2, O_3, O_{1+2}, O_{2+3}$	87.96%
$O_1, O_2, O_3, O_{1+2}, sum$	88.61%
$O_1, O_2, O_3, O_{2+3}, sum$	88.80%
$O_1, O_2, O_3, O_{1+2}, O_{2+3}, sum$	88.92%

This experimentation is to investigate the effect of utilising linear interpolation to enrich the representation of features extracted by the given DeSTIN. It first focuses on the use of 3-dimensional DeSTIN outputs in order to minimise the underlying computational complexity required: $V_{original} = (O_1, O_2, O_3)^T$. Additional vectors of 2 dimensions $V_{additional} = (i_1, i_2)^T$ are generated from the original vectors, and there are 3 different kinds of possible combination: $V_{additional} = (i_1 = O_{1+2}, i_2 = O_{2+3})^T$, $V_{additional} = (i_1 = O_{1+2}, i_2 = sum)^T$, and $V_{additional} = (i_1 = O_{2+3}, i_2 = sum)^T$, where $sum = \sum_{i=1}^{3} O_i$. A further interpolated vector of 3 dimensions $V_{additional} = (i_1 = O_{1+2}, i_2 = O_{2+3}, i_3 =$

TABLE II. ACCURACY USING LINEAR INTERPOLATION BASED ON DIFFERENT DIMENSIONAL ORIGINAL VECTORS

Feature list	Combination	Exception	Best	Worst	Average	Original
$O_1, O_2, O_3, i_1, \ldots, i_m (m = 2, 3)$	4	0	88.92%	87.96%	88.57%	86.73%
$O_1, O_2, O_3, O_4, i_1, \ldots, i_m (m = 2, 3, 4)$	11	0	97.56%	97.03%	97.29%	96.52%
$O_1, O_2, O_3, O_4, O_5, i_1, \ldots, i_m (m = 2, 3, 4, 5)$	26	1	98.56%	98.20%	98.41%	98.22%
$O_1, O_2, O_3, O_4, O_5, O_6, i_1, \ldots, i_m (m = 2, 3, 4, 5, 6)$	57	1	98.73%	98.22%	98.50%	98.25%

$sum)^T$ can also be generated. Table I lists the correct classification rates produced by the resulting SVM classifiers, respectively using different vectors composed by the union of the original features and certain interpolated features. Clearly, the classification accuracy of using the DESTINI features is greater than 86.73%, the accuracy obtained using the original features alone.

Experimentation has also been carried out for cases where more than three original DeSTIN features are used. Table II lists the correct classification rates based on the DeSTIN outputs of a different dimensionality. It presents six performance indicators to show the accuracy. When the output of the underlying DeSTIN is a 4-dimensional vector, the best classification accuracy of using the DESTINI features is 97.56%, while the worst is 97.03%, with the average accuracy being 97.29%, which is significantly higher than that of using four original features (96.52%). Also, the classification accuracy of using the DESTINI features is generally higher than that (98.22%) of using 5-dimensional original DeSTIN features, with only one exception, where the $V_{additional} = (i_1 = O_{3+4}, i_2 = O_{4+5})^T$. Even on this occasion, the accuracy is 98.20%, a mere 0.02% worse off. The classification accuracy of using the DESTINI features is obviously better than that (98.25%) of using the original 6 dimensional features, again with only one exception, where the $V_{additional} = (i_1 = O_{4+5}, i_2 = O_{5+6})^T$ involving a tiny difference in value (0.03%).

Overall, it can be seen from the above results that the classification rates using the DESTINI features are higher than those achievable using the original DeSTIN features alone. This shows that the classification accuracy is improved with only very minor extra computation overheads, without the need of directly generating a larger number of DeSTIN features which would otherwise incur substantially more computation.

B. Use of DESTINI features with Newton interpolation

The second experimentation reported here deals with the use of applying Newton interpolation to the DeSTIN outputs. The first subset of experiments involves a 4-dimensional original feature vector $V_{original} = (O_1, O_2, O_3, O_4)^T$. Additional vectors of one dimension $V_{additional} = (I_1)^T$ are artificially created using the original feature vector. There are 5 different kinds of such artificial vector. Additional vectors of 2 dimensions $V_{additional} = (I_1, I_2)^T$ are also generated and in this case, there are $C^n_{n+2} = C^4_6$ possible combinations with the original. Of course, many further interpolated vectors can be generated which are shown in Table III. The best classification accuracy of using the DESTINI features is 98.65%, and the average accuracy is 98.03%. There is only one exception, where the use of $V_D = (O_1, I_1, O_2, I_2, O_3, I_3, O_4)^T$ fails to beat the use of just the original DeSTIN features, though the difference between 96.43% and 96.52% is rather small.

The second subset of experiments investigates the correct classification rates produced by the SVM classifiers through the use of different dimensional DeSTIN outputs. The results are listed in Table IV. When the output of the given DeSTIN is a 3-dimensional vector, the best classification accuracy using the DESTINI features is 95.62%, while the average reaches 90.96%. There are 3 exceptions (out of 15 combinations), which are $V_D = (O_1, I_1, O_2, O_3)^T$, $V_D = (O_1, O_2, I_1, O_3)^T$, and $V_D = (O_1, I_1, O_2, I_2, O_3)^T$ where the use of the DESTINI features leads to a lower accuracy than that (86.73%) obtainable by the use of the original DeSTIN features (with the corresponding accuracy rates being 86.47%, 85.97% and 85.1%).

The performance of using the DESTINI features can be further improved when the dimensionality of the original DeSTIN feature vectors is slightly increased. For example, with 5-dimensional DeSTIN feature vectors, the accuracy rate is generally higher than that (98.22%) of using just the original, with only two exceptions over 63 cases (where $V_D = (O_1, O_2, I_1, O_3, I_2, O_4, O_5)^T$ and $V_D = (O_1, O_2, I_1, O_3, I_2, O_4, I_3, O_5)^T$) that lead to slightly lower rates (98.09% and 98.02%). However, the average accuracy is 98.68%. Also, the classification accuracy of using the DESTINI features is obviously better than that (98.25%) when using 6-dimensional feature vectors, with only one exception (of $V_D = (O_1, O_2, O_3, I_1, O_4, O_5, O_6)^T$). Even on this exceptional occasion, the accuracy is 98.14%, a mere degradation in performance (0.11%). Overall, it is clear that the classification rates attainable by the use of the DESTINI features are significantly higher than those of using the original features alone.

C. Comparison between linear and Newton interpolations

Clearly, both interpolation methods work well with DeSTIN. The employment of linear interpolation helps improve the accuracy without causing much computation, while Newton interpolation performs even better with a little extra computation. To support a more direct comparison between the two versions of DESTINI, Figs. 4, 5, 6 and 7 summarise the accuracy rates gained using Newton interpolation and those using linear interpolation, for cases where the original feature set utilised is of a dimensionality of 3, 4, 5 and 6, respectively.

The general trends across all cases are rather similar. Although the worst result obtained using Newton interpolation is worse than that of using linear interpolation, it is known from the previous discussions that only for few situations, using either linear or Newton interpolation, does DESTINI show such under-performed behaviour. For a great majority of cases, Newton interpolation-based DESTINI systems outperform linear interpolation-based ones. In particular, consider the case of having a 3-dimensional original feature vector as an example. The best performance achieved by the use of linear

TABLE III. ACCURACY USING NEWTON INTERPOLATED FEATURES BASED ON A 4-DIMENSIONAL ORIGINAL VECTOR

Feature list	Accuracy	Feature list	Accuracy
I_1, O_1, O_2, O_3, O_4	98.39%	$I_1, O_1, I_2, O_2, I_3, O_3, O_4$	98.10%
O_1, I_1, O_2, O_3, O_4	97.18%	$I_1, O_1, I_2, O_2, O_3, I_3, O_4$	98.28%
O_1, O_2, I_1, O_3, O_4	96.65%	$I_1, O_1, I_2, O_2, O_3, O_4, I_3$	98.58%
O_1, O_2, O_3, I_1, O_4	97.34%	$I_1, O_1, O_2, I_2, O_3, I_3, O_4$	98.06%
O_1, O_2, O_3, O_4, I_1	98.37%	$I_1, O_1, O_2, I_2, O_3, O_4, I_3$	98.58%
$I_1, O_1, I_2, O_2, O_3, O_4$	98.62%	$I_1, O_1, O_2, O_3, I_2, O_4, I_3$	98.60%
$I_1, O_1, O_2, I_2, O_3, O_4$	98.16%	$O_1, I_1, O_2, I_2, O_3, I_3, O_4$	96.43%
$I_1, O_1, O_2, O_3, I_2, O_4$	98.21%	$O_1, I_1, O_2, I_2, O_3, O_4, I_3$	98.01%
$I_1, O_1, O_2, O_3, O_4, I_2$	98.45%	$O_1, I_1, O_2, O_3, I_2, O_4, I_3$	98.59%
$O_1, I_1, O_2, I_1, O_3, O_4$	96.86%	$O_1, O_2, I_1, O_3, I_2, O_4, I_3$	98.18%
$O_1, I_1, O_2, O_3, I_2, O_4$	97.24%	$I_1, O_1, I_2, O_2, I_3, O_3, I_4, O_4$	97.92%
$O_1, I_1, O_2, O_3, O_4, I_2$	98.59%	$I_1, O_1, I_2, O_2, I_3, O_3, O_4, I_4$	98.65%
$O_1, O_2, I_1, O_3, I_2, O_4$	96.93%	$I_1, O_1, I_2, O_2, O_3, I_3, O_4, I_4$	98.65%
$O_1, O_2, I_1, O_3, O_4, I_2$	98.28%	$I_1, O_1, O_2, I_2, O_3, I_3, O_4, I_4$	98.45%
$O_1, O_2, O_3, I_1, O_4, I_2$	98.49%	$O_1, I_1, O_2, I_2, O_3, I_3, O_4, I_4$	97.68%
$I_1, O_1, I_2, O_2, I_3, O_3, I_4, O_4, I_5$	98.43%	O_1, O_2, O_3, O_4	96.52%

TABLE IV. ACCURACY USING NEWTON INTERPOLATION BASED ON DIFFERENT DIMENSIONAL ORIGINAL VECTORS

Feature list	Combination	Exception	Best	Worst	Average	Original
$O_1, O_2, O_3, I_1, \ldots, I_m (m = 1, 2, 3, 4)$	15	3	95.62%	85.10%	90.96%	86.73%
$O_1, O_2, O_3, O_4, I_1, \ldots, I_m (m = 1, 2, 3, 4, 5)$	31	1	98.65%	96.43%	98.03%	96.52%
$O_1, O_2, O_3, O_4, O_5, I_1, \ldots, I_m (m = 1, 2, 3, 4, 5, 6)$	63	2	98.98%	98.02%	98.68%	98.22%
$O_1, O_2, O_3, O_4, O_5, O_6, I_1, \ldots, I_m (m = 1, 2, 3, 4, 5, 6, 7)$	127	1	98.92%	98.14%	98.65%	98.25%

interpolation is 88.92% whilst its counterpart using Newton interpolation is 95.62%. Also, the average accuracy by using Newton interpolation is 90.96% which is also higher than 88.57% that is achievable using linear interpolation. Of course, as stated earlier, Newton interpolation does incur slightly more computation. Nevertheless, the cost is still substantially less than that required to generate and use vectors of original features of a dimensionality which is equal to that of the artificially expanded feature sets.

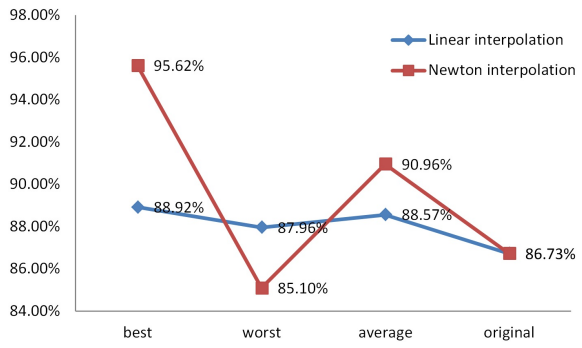

Fig. 4. Accuracy based on a 3-dimensional original vector

VII. CONCLUSION

This paper has presented a novel deep learning architecture which draws upon the fundamentals of DeSTIN, supported by feature interpolation. The resulting feature extraction mechanism is well-suited for image classification which is implemented using popular SVMs. This work has been tested using the MNIST dataset. Systematic experimental results demonstrate that the approach developed in this research is capable of efficiently extracting features suitable for input to SVM-based classifiers, generally delivering significantly improved performance in terms of classification accuracy.

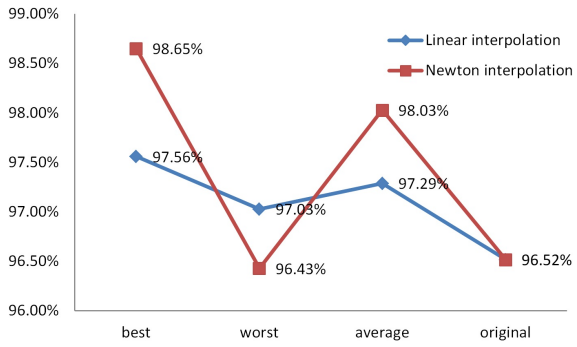

Fig. 5. Accuracy based on a 4-dimensional original vector

While promising, the work also gives rise to a number of open issues. First of all, it is interesting to investigate whether either of the two simple interpolation mechanisms will work when different original feature extraction methods are used. Also, a combined application of both linear and Newton interpolation may help further enrich the feature space.

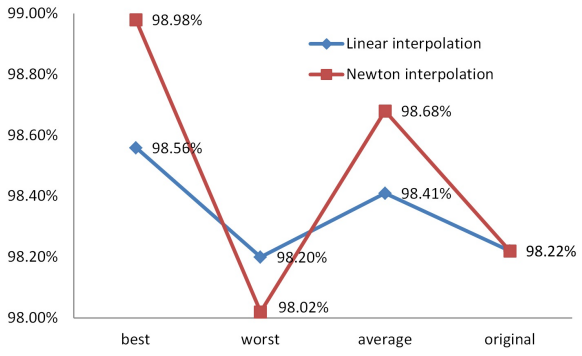

Fig. 6. Accuracy based on a 5-dimensional original vector

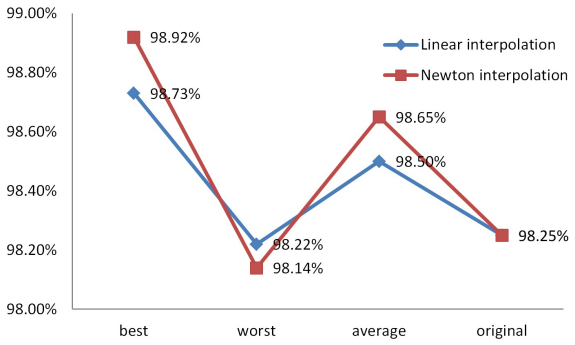

Fig. 7. Accuracy based on a 6-dimensional original vector

In addition, it is very interesting to apply the work to more complex application domains (e.g., to Mars images which vary significantly in terms of intensity, scale and rotation, and are blurred with measurement and transmission noise [11]). Furthermore, the implemented system performs interpolation on given features; it may be worth exploring the implication of interpolating classification rules themselves too, using advanced interpolation techniques such as those reported in [18], [19]. Finally, it is worth exploring whether imposing a certain selection of interpolated features may further reduce the overall computation cost for classification [20], [21].

REFERENCES

[1] 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.

[2] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," 1998.

[3] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[4] J.-C. Chappelier and A. Grumbach, "Rst: a connectionist architecture to deal with spatiotemporal relationships," *Neural computation*, vol. 10, no. 4, pp. 883–902, 1998.

[5] R. J. Douglas and K. A. Martin, "Neuronal circuits of the neocortex," *Annu. Rev. Neurosci.*, vol. 27, pp. 419–451, 2004.

[6] 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.

[7] A. Rockel, R. W. Hiorns, and T. P. Powell, "The basic uniformity in structure of the neocortex," *Brain*, vol. 103, no. 2, pp. 221–244, 1980.

[8] 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.

[9] S. Young, I. Arel, T. P. Karnowski, and D. Rose, "A fast and stable incremental clustering algorithm," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*. IEEE, 2010, pp. 204–209.

[10] K. Huang and S. Aviyente, "Wavelet feature selection for image classification," *Image Processing, IEEE Transactions on*, vol. 17, no. 9, pp. 1709–1720, 2008.

[11] C. Shang and D. Barnes, "Fuzzy-rough feature selection aided support vector machines for mars image classification," *Computer Vision and Image Understanding*, vol. 117, no. 3, pp. 202–213, 2013.

[12] V. N. Vapnik, "Statistical learning theory," 1998.

[13] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *Neural Networks, IEEE Transactions on*, vol. 10, no. 5, pp. 1055–1064, 1999.

[14] J. C. Platt, *12 Fast Training of Support Vector Machines using Sequential Minimal Optimization*. MIT Press, 1999.

[15] ——, "Sequential minimal optimization: A fast algorithm for training support vector machines," in *ADVANCES IN KERNEL METHODS-SUPPORT VECTOR LEARNING*, 1998.

[16] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[17] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[18] Z. Huang and Q. Shen, "Fuzzy interpolation and extrapolation: A practical approach," *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 1, pp. 13–28, 2008.

[19] L. Yang and Q. Shen, "Adaptive fuzzy interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 6, pp. 1107–1126, 2011.

[20] R. Diao and Q. Shen, "Feature selection with harmony search," *IEEE Trans. Syst., Man, Cybern. B*, vol. 42, no. 6, pp. 1509–1523, 2012.

[21] R. Jensen and Q. Shen, "New approaches to fuzzy-rough feature selection," *Fuzzy Systems, IEEE Transactions on*, vol. 17, no. 4, pp. 824–838, 2009.