Appl. Comput. Harmon. Anal. ••• (••••) •••–•••

Contents lists available at ScienceDirect

Applied and Computational Harmonic Analysis

www.elsevier.com/locate/acha

Diffusion nets

Gal Mishne [a,*], Uri Shaham [b,c], Alexander Cloninger [d], Israel Cohen [e]

[a] *Department of Mathematics, Yale University, New Haven, CT 06520, USA*
[b] *Final research, Hertzliya 46730, Israel*
[c] *Center for Outcome Research, Yale University, New Haven, CT 06520, USA*
[d] *Department of Mathematics, University of California, San Diego, La Jolla, CA 92093, USA*
[e] *Faculty of Electrical Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel*

ARTICLE INFO

Article history:
Received 25 June 2015
Received in revised form 25 January 2017
Accepted 23 August 2017
Available online xxxx
Communicated by Charles K. Chui

Keywords:
Manifold learning
Diffusion maps
Deep learning
Autoencoder
Out-of-sample extension

ABSTRACT

Non-linear manifold learning enables high-dimensional data analysis, but requires out-of-sample-extension methods to process new data points. In this paper, we propose a manifold learning algorithm based on deep learning to create an encoder, which maps a high-dimensional dataset to its low-dimensional embedding, and a decoder, which takes the embedded data back to the high-dimensional space. Stacking the encoder and decoder together constructs an autoencoder, which we term a diffusion net, that performs out-of-sample-extension as well as outlier detection. We introduce new neural net constraints for the encoder, which preserve the local geometry of the points, and we prove rates of convergence for the encoder. Also, our approach is efficient in both computational complexity and memory requirements, as opposed to previous methods that require storage of all training points in both the high-dimensional and the low-dimensional spaces to calculate the out-of-sample-extension and the pre-image of new points.

1. Introduction

Real world data is often high dimensional, yet concentrates near a lower dimensional manifold, embedded in the high dimensional ambient space. In many applications, finding a low-dimensional representation of the data is necessary to efficiently handle it and the representation usually reveals meaningful structures within the data. In recent years, different manifold learning methods have been developed for high dimensional data analysis, which are based on the geometry of the data, i.e. preserving distances within local neighborhoods of the data. These include kernel PCA [1], ISOMAP [2], locally linear embedding (LLE) [3], Laplacian eigenmaps [4], Hessian eigenmaps [5] and Diffusion maps [6]. Some of these methods are spectral methods, based on the eigenvectors of adjacency or affinity matrices of graphs on the data. These methods are capable

* Corresponding author.
E-mail addresses: gal.mishne@yale.edu (G. Mishne), uri.shaham@yale.edu (U. Shaham), acloninger@ucsd.edu (A. Cloninger), icohen@ee.technion.ac.il (I. Cohen).

http://dx.doi.org/10.1016/j.acha.2017.08.007
1063-5203/

of capturing a smooth representation of the data and have been shown to be robust to noise and outliers. The ability to capture the underlying low-dimensional structure of data has made these methods appropriate for dimensionality reduction. Unlike classical dimensionality reduction methods, such as principal component analysis (PCA), these methods are nonlinear, which is essential as real-world data typically does not lie on a hyperplane. In addition, they preserve local structures in the data while disregarding distances between points that are far apart, which are typically meaningless in high-dimensional data. These approaches are very popular in machine learning, signal processing and data mining applications.

When the data set is very large, or when processing online sequential data, it is impractical to directly compute an embedding for the entire dataset. The computational complexity of calculating the affinity matrix and the eigen-decomposition of the matrix become infeasible in terms of memory and running-time. Since these non-linear techniques do not provide an explicit mapping from the data to the embedding, out-of-sample extension (OOSE) methods are used to extend the embedding to new data points [7–14]. In such cases, the low-dimensional embedding is constructed for a representative sample of the data and is then extended to all remaining, or new points. This is a common approach in image processing applications, for example, especially for high-resolution images [15,16].

Deep learning has gained popularity in the past years, achieving state-of-the-art results in machine learning, computer vision and speech processing applications, handling increasingly large datasets [17–19]. Deep neural nets are capable of learning increasingly abstract representations for the data [20], some of which are robust to small perturbations around training points [21,22]. However, these representations are built globally, without incorporating local geometry or density of the data. Jia et al. [23] recently introduced Laplacian Autoencoders, which impose locality preserving constraints via the weighted affinity matrix. Their goal is to improve pre-training of autoencoders in constructing neural networks. However, their formulation lacks both a parameter to balance between the reconstruction loss and the affinity regularizer, and an embedding to compare to, so there is no way to ensure that the final representation is anywhere near the true manifold eigenfunctions. Because of this, there are no theoretical guarantees on the convergence of the algorithm or on the usefulness of the representations.

In this paper, we propose a new approach to out-of-sample extension, applying a deep neural network to learn the mapping between the data and the embedding. We employ deep learning from a manifold learning perspective, by explicitly incorporating a manifold embedding of the data in the deep learning framework. We address three closely connected problems: OOSE of the embedding to new points, a pre-image solution [24, 25] that can include regularization, and outlier detection on test data. This third goal is important and often neglected in OOSE methods. If new data is an outlier and does not fit the model of the training data, the extension is ill-defined and its new representation will be insufficient.

To accomplish these three goals, we propose to train a neural network-based encoder and decoder, and combine them in an autoencoder. First, we train a multi-layer encoder to approximate the low-dimensional embedding on the data; specifically we use diffusion maps [6] for embedding the data. The encoder performs OOSE of the embedding. We then train a multi-layer decoder whose input is the diffusion map, to learn the inverse mapping between the embedding and the data. The decoder enables to recover the pre-image of the embedding, mapping new points in the diffusion space to the high-dimensional data space. Finally, by stacking the two networks, we obtain an autoencoder, termed the diffusion net. The diffusion net enables to perform outlier detection, indicating when the extension of a given point via the encoder is faulty due to its being an outlier that does not follow the model of the data. The diffusion net also performs denoising of the data, reconstructing a clean version of noisy data. Our approach is both computationally efficient and has low memory costs. Once the diffusion net has been trained, it is unnecessary to retain the training data and embeddings as required in other methods. Thus, harnessing the efficiency of deep learning networks enables to efficiently process large quantities of data.

This paper is organized as follows. Section 2 provides background on manifold learning and deep neural networks. In Sections 3 and 4 we propose a deep learning approach for manifold learning, enabling out-of-

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

sample extension, pre-image computation and outlier detection. A proof on bounding the convergence of a single layer encoder is presented in Section 5. We present experimental results in Section 6. Future research directions are discussed in Section 7.

2. Background and related work

2.1. Diffusion maps

Diffusion maps is a popular manifold learning technique, based on the construction of the graph Laplacian of the data set [9]. It has been used successfully in various signal processing, image processing and machine learning applications [10,16,26–32]. In this section, we briefly review the diffusion maps construction. For a detailed discussion on diffusion maps, see [6].

Given a high dimensional set $X = \{x_i\}_{i=1}^m$, a weighted graph is constructed with the data points as nodes and the weight of the edge connecting two nodes, $k(x_i, x_j)$, $x_i, x_j \in X$, as a measure of the similarity between the two data points. The affinity matrix $K[i,j] = k(x_i, x_j)$ is required to be symmetric and non-negative, where a common choice is a radial basis function (RBF) kernel

$$k(x_i, x_j) = \exp\left\{-\|x_i - x_j\|^2/\sigma^2\right\}, \tag{1}$$

where $\sigma > 0$ is a global scale parameter. A local scale can be set for each point as in [33]. In practice, K can be computed using only the nearest neighbors of every point and $K[i,j]$ is set to zero for x_j that are not among the nearest neighbors of x_i.

We apply a normalization of the data to obtain an approximation of the Laplace–Beltrami operator on the data, so the embedding will not rely on the distribution of the points [6,10]. The kernel is normalized by the degree of each point $D[i,i] = \sum_{j\in X} k(x_i, x_j)$, by

$$\widetilde{K} = D^{-1} K D^{-1}.$$

A random walk is then created on the normalized data set by:

$$P = \widetilde{D}^{-1}\widetilde{K}, \quad \widetilde{D}(i,i) = \sum_j \widetilde{K}[i,j]. \tag{2}$$

The row-stochastic matrix P satisfies $P[i,j] \geq 0$ and $\sum_{j\in X} P[i,j] = 1$ and therefore can be viewed as the transition matrix of a Markov chain on the data set X. The eigen-decomposition of P yields a sequence of biorthogonal left and right eigenvectors, ϕ_ℓ and ψ_ℓ respectively, and a sequence of eigenvalues: $1 = |\lambda_0| \geq |\lambda_1| \geq \ldots$. Then, t steps of the Markov chain can be calculated as

$$p_t(x_i, x_j) = \sum_{l\geq 0} \lambda_\ell^t \psi_\ell(x_i)\phi_\ell(x_j).$$

A diffusion distance $d_{\mathrm{DM}}(x_i, x_j; t)$ between two points $x_i, x_j \in X$ is defined by

$$d_{\mathrm{DM}}(x_i, x_j; t) = \sum_{x_k\in X} \frac{\left(p_t(x_i, x_k) - p_t(x_j, x_k)\right)^2}{\phi_0(x_k)} = \sum_{l\geq 1} \lambda_\ell^{2t}(\psi_\ell(x_i) - \psi_\ell(x_j))^2, \tag{3}$$

where ϕ_0 is the stationary probability distribution on the graph. This metric is robust to noise, since the distance between two points depends on all possible paths of length t between the points. Due to the spectrum decay of P, the diffusion distance can be approximated using only the first d eigenvectors. Equation (3)

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

implies that a mapping can be defined between the original space and the eigenvectors ψ_ℓ. Retaining only the first d eigenvectors, the mapping Ψ_t embeds the data set X into the Euclidean space $\mathbb{R}^d$, where the diffusion distance is equal to the Euclidean distance in this new embedding:

$$\Psi_t : x_i \to \left(\lambda_1^t \psi_1(x_i), \lambda_2^t \psi_2(x_i), ..., \lambda_d^t \psi_d(x_i)\right)^T. \tag{4}$$

Note that ψ_0 is not used in the embedding because it is a constant vector. In this paper we set $t = 1$, but our approach can be used for estimating the embedding for general t.

2.2. Out-of-sample function extension

Having calculated a diffusion map Ψ on the data X, various methods have been proposed for extending Ψ to new points. In simple examples, there are analytic mechanisms for creating harmonic extensions when the eigenfunctions of the Laplacian can be derived analytically. However, this is not applicable in the general case. The Nyström extension method is a popular method for general OOSE. Given a new point $x' \in \mathcal{M} \setminus X$, the eigenvector ψ_ℓ is extended to this point as:

$$\widehat{\psi}_\ell(x') = \frac{1}{\lambda_\ell} \sum_{j=1}^{m} p(x', x_j) \psi_\ell(x_j), \quad \ell = 1, \cdots, d. \tag{5}$$

Geometric Harmonics [9,10] is an OOSE method which improves upon the Nyström extension method. It treats both the numerical instability due to λ by extending only the eigenvectors with significant eigenvalues. In addition, it finds an appropriate scale for the kernel in the extension, dependent on the function that is being extended. Rabin and Coifman proposed a Laplacian pyramids-based OOSE method in [11]. The eigenvectors are extended in an iterative multi-scale scheme, where the number of scales is adapted to the complexity of each eigenvector separately. This approach was recently extended in [12] to implicitly incorporate cross validation in the training procedure and avoid over-fitting in the training. Aizenbud, Bermanis and Averbuch [14] introduced an extension method based on a generalized least squares solution for each new test point within its local neighborhood in the training set. This solution is shown to minimize the Mahalanobis distance between the embedding of the training points and the estimated embedding for the new point, in respect to a covariance matrix that incorporates geometric properties of the data and embedding.

The computational complexity of these methods typically depends on the number of training points, since these methods rely on the distances between a new test point and all training points, or its nearest neighbors in the training set (determined by a nearest neighbor search algorithm). Therefore, to perform OOSE, it is necessary to keep all the training points and their corresponding embeddings in memory. If the affinity matrix K in (2) is based on non-Euclidean local metrics, such as [31,34–37], it is not possible to use nearest neighbors search, since each training point is associated with its own local metric. This increases the complexity of the distance calculation and adds to the memory requirements of the OOSE. The method we propose has no such requirement. After training, our approach enables OOSE without retaining any of the training data or embeddings.

2.3. Deep neural networks

Artificial neural networks (ANNs) are networks composed of interconnected computational units termed *neurons*, which are typically organized in layers. A deep neural network is composed of multiple hidden layers, and is typically designed as a feed-forward network, in which there are no cycles or loops. In our framework, we use multi-layer perceptrons (MLP), which are a popular and important class of neural nets,

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

in which the layers are densely connected. The output of each layer is computed as an affine mapping of the previous layer followed by a non-linear function:

$$a^{(l+1)} = \sigma(W^{(l)}a^{(l)} + b^{(l)}),$$

where a is termed the activation, $W^{(l)}[i,j]$ denotes the weight associated with the connection between unit j in layer l and unit i in layer $l+1$, $b^{(l)}$ is a bias vector, $\sigma(\cdot)$ is a non-linear function applied element-wise, and $a^{(1)} = X$. We denote the number of hidden units in layer l by s_l and the overall number of layers in a network, including the input and output, by L. In our experiments, we used a sigmoid non-linearity in the activation: $\sigma(z) = \frac{1}{1+e^{-z}}$. Other choices include $\sigma(z) = \tanh(z)$ and rectified linear units: $\sigma(z) = \mathrm{ReLU}(z) = \max\{0, z\}$.

Deep nets have been successfully applied to various tasks such as regression for learning a function over a given dataset, classification, feature learning, etc., achieving state-of-the-art results. The task the network performs is determined by the output layer and the cost function minimized over the network. In supervised learning, the goal is to predict a function or labels on the input data. The cost function of a network consists of a loss function, and a weight regularization term is usually added in order to prevent over-fitting. Given a multi-layer network, we denote the weights and biases of all the layers by $\theta = \{W^{(l)}, b^{(l)}\}_l$. For regression of a multi-dimensional function, $y \in \mathbb{R}^d$, as in our application, the squared error can be used for the loss term:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m} \|o(x_i;\theta) - y_i\|^2 + \frac{\mu}{2}\sum_{l=1}^{L-1} \|W^{(l)}\|_F^2, \tag{6}$$

where $o(x_i) \in \mathbb{R}^d$ is the output of the net for input x_i, $\|\cdot\|_F^2$ is the square of the Frobenius norm, μ is a cost parameter. We use an l_2 penalty on the weights, which is a very popular choice for regularization. Neural nets are typically trained using variants of stochastic gradient descent (SGD) for calculating the weight and biases in the network that minimize the cost function. The gradient of the loss function relative to the weights is computed efficiently using backpropagation [38], starting from the output layer backwards to the input.

2.3.1. Autoencoders

Deep learning can also be used in an unsupervised manner, such as in training autoencoders. An autoencoder is composed of an encoder function $f(\cdot)$ and a decoder function $g(\cdot)$, where the dimension of f is typically smaller than the dimension of the input data. The reconstruction of an input x_i is given by stacking the decoder on the encoder: $r(x_i) = g(f(x_i))$, and the autoencoder is trained to minimize a reconstruction error loss $L(x, r(x))$ over the training points, i.e. trying to approximate the identity function. This is performed by setting the number of output units of the decoder to be the dimension of the input data, and fine-tuning a loss function between the output and the input. The output of the encoder, i.e. the middle layer of the autoencoder, can be seen as a low-dimensional representation of the data. Autoencoders were popularized by Hinton [18]. Basic autoencoders consist of a single hidden layer [21,22], but deep autoencoders have also been proposed for classification, denoising, image retrieval, speech enhancement and more [18,21].

3. Diffusion net

3.1. Problem setup

We assume the data lies on a smooth, compact, d-dimensional Riemannian manifold $\mathcal{M}$ embedded in a high-dimensional space $\mathbb{R}^n$, $(d < n)$. Calculating the diffusion map $\Psi : X \to \mathbb{R}^d$ (4) for a given training

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

set $X = \{x_i\}_{i=1}^m$ maps the high-dimensional space to Euclidean space $\mathbb{R}^d$, revealing the low-dimensional structure of the data. We address three related problems in this setting: (a) Out-of-sample extension, (b) pre-image solution and (c) outlier detection.

Given new test points $X' = \{x'\} \subseteq \mathcal{M} \setminus X$, the purpose of an out-of-sample extension method is to extend Ψ to the new points. For the given training points, the extension should be as close as possible to the true embedding: $\|\widehat{\Psi}(x_i) - \Psi(x_i)\|_2 < \epsilon$, for small ϵ. In addition, the extension $\widehat{\Psi}(x')$ should preserve the properties of the original embedding, such as preserving local structures in the data.

The second problem we address is calculating the pre-image [24,25]. Given a point φ in the diffusion space, the pre-image of φ is a data-point x for which $\Psi(x) = \varphi$. Note that the problem is technically ill-posed, as such a point x might not exist. However, we can still solve for an approximate pre-image by solving

$$\hat{x} = \arg\min_x \|\Psi(x) - \varphi\|^2. \tag{7}$$

The pre-image problem has been shown to be closely related to the OOSE problem [25]. Calculating the pre-image is essential to being able to pull back calculations made in the embedding space to the data space.

Finally, we aim to provide a new outlier detection measure, to detects outliers in newly arrived data. In OOSE algorithms, the ability to provide a good embedding for new points depends on how well they follow the model of the training data, i.e. their distance from the training data. For example in kernel-based extension methods such as Nyström, the scale of the Gaussian kernel limits how close points need to be to the training data. Denote by $x^* = \arg\min_{x_i \in X} \|x' - x_i\|$ the nearest-neighbor of a test point x' within the training set X. If the distance between the two is very large, $\|x' - x^*\| \gg \sigma$, then $\widehat{\Psi}(x') = \Psi(x^*)$, i.e. the OOSE reverts to 1-nearest neighbor prediction. Numerically, if $\|x' - x^*\|/\sigma \to \infty$, then the affinity to the training set evaluates to zero, and the OOSE is the origin. In related works (e.g. [8–11]), there is typically an implicit assumption that the test data follows the distribution of the training data. However, the embedding obtained for outliers is uninformative and can lead to mistakes in the task the embedding is used for, i.e. classification, signal processing, etc. Therefore, indicating if a point is an outlier within an OOSE framework is important to properly processing the data. We define x' as an outlier with respect to X, the dataset for which the diffusion map was calculated, if $\hat{\Psi}(x') \neq \Psi(x')$, where $\Psi(x')$ is the embedding we would obtain if $x' \in X$ when calculating the embedding.

3.2. Our approach

We propose a solution to these problems based on deep neural nets, training a neural network-based encoder and decoder, and combining them in a deep multi-layer autoencoder. Thus, instead of training a general autoencoder in which the middle layer can be seen as providing a low-dimensional representation of the data, we incorporate the given embedding on the data in the training procedure. The encoder and the decoder are both MLPs composed of L layers, whose output layer is a regression layer. In our approach, the encoder learns a mapping from the manifold to the diffusion space, by minimizing the squared loss between the output of the encoder and $\Psi(x)$. In addition, we impose a new constraint to preserve properties of the embedding. The decoder learns the inverse mapping from the low-dimensional diffusion space back to the high-dimensional space of the data, solving the pre-image problem. By stacking the encoder and decoder we obtain an autoencoder, whose inner-most layer computes the diffusion embedding of the data. The autoencoder can be used for both outlier detection and denoising. Our framework is presented in Algorithms 1, 2 and 3.

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

Algorithm 1 Out-of-sample extension.

Encoder (Section 3.3)
Training phase:
Input Training set X, diffusion embedding Ψ, number of layers for encoder L.
1: Initialize the weights $\theta^e = \{W^{(l)}, b^{(l)}\}_l$ of the encoder and set $a^{(1)} = X$, $y = \Psi$.
2: Optional: pre-train every hidden layer as an autoencoder. (Section 4.1)
3: Optimize the parameters of the network by fine-tuning cost (9) with back-propagation. (Section 4.2)
Test phase:
Input test data X'
4: Calculate the out-of-sample-extension to new points as the output of the encoder $\widehat{\Psi}(x') = o^e(x')$.

Algorithm 2 Pre-image.

Decoder (Section 3.4)
Training phase:
Input Training set X, diffusion embedding Ψ, number of layers for decoder L.
1: Initialize the weights $\theta^d = \{W'^{(l)}, b'^{(l)}\}_l$ of the decoder and set $a^{(1)} = \Psi$, $y = X$.
2: Optional: pre-train every hidden layer as an autoencoder.
3: Optimize the parameters of the network by fine-tuning cost (10) with back-propagation.
Test phase:
Input test points in the embedding space $\{\varphi\}$
4: Calculate the pre-image of the test points as the output of the decoder $\widehat{x} = o^d(\varphi)$.

Algorithm 3 Autoencoder.

Autoencoder (Section 3.5)
Training phase:
1: Stack decoder trained as in Algorithm 2 on top of encoder trained as in Algorithm 1 and obtain autoencoder with $2L-1$ layers.
2: Calculate average reconstruction error ϵ.
Test phase:
Input test data X'
3: Calculate reconstruction of the point as the output of the autoencoder $\widehat{x'} = r(x')$.
4: Calculate an outlier detection score as threshold on the reconstruction cost $\|r(x') - x'\| > C\epsilon$.

3.3. Encoder

The encoder learns the mapping between the high-dimensional data space and the embedding space. The encoder is designed as an MLP, minimizing the l_2 loss between the diffusion map Ψ and the output of the net, i.e. $o^e(x_i) = \Psi(i)$ in (6), where o^e denotes the output matrix of the encoder for all training points, $o^e \in \mathbb{R}^{d\times m}$. To facilitate the output of the encoder approximating the diffusion map, we add a new constraint to the training objective of the encoder. Since the coordinates of the diffusion map are eigenvectors of the random walk matrix on the data, P in (2), we add a new cost term on the output of the encoder being an eigenvector of this matrix. The additional term, termed the eigenvector (EV) constraint, is

$$J^{\mathrm{EV}} = \frac{\eta}{2m}\sum_{j=1}^{d}\|(P - \lambda_j I_{m\times m})(o_j^e)^T\|^2, \tag{8}$$

where o_j^e is the j-th row of the output matrix, λ_j is the j-th eigenvalue of the affinity matrix, and η is an optimization cost parameter.

This new term imposes smooth locality on the training points, and maintains the local geometry of the points in the encoder function. In contrast to general constraints in deep learning, where each data point is processed independently, this constraint "mixes" output values of different data points together, via the random-walk matrix. The EV constraint enforces that an output value for a given data point can be reconstructed as a weighted average of the outputs of the local neighborhood of the data point. The locality of the neighborhood depends on the bandwidth of the kernel used to define the affinity matrix (1). This constraint serves to minimize the loss error on the output, such that the output is not a general regression of the desired function, but an eigenvector of the matrix, thus maintaining the geometric properties of the embedding.

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

Although we use diffusion maps for manifold learning, other kernel methods may be used and imposed with the EV constraint. This only requires replacing the matrix P with the matrix that is decomposed in other approaches, for example the normalized affinity matrix $D^{-1/2}KD^{-1/2}$ as in spectral clustering [39] or the unnormalized graph Laplacian matrix $L = D - K$ as in Laplacian Eigenmaps [4].

Adding the EV constraint to the cost function of the encoder, the total cost is:

$$J^e(\theta^e) = \frac{1}{2m}\sum_{i=1}^{m}\|o^e(x_i) - \Psi(x_i)\|^2 + \frac{\mu}{2}\sum_{l=1}^{L-1}\|W^{(l)}\|_F^2 + \frac{\eta}{2m}\sum_{j=1}^{d}\|(P-\lambda_j I)(o_j^e)^T\|^2, \tag{9}$$

where $\theta^e = \{W^{(l)}, b^{(l)}\}_l$ denotes the set of the weights and biases of all the layers of the encoder. To incorporate this constraint in the gradient calculation in back-propagation, the gradient of this constraint in regards to the output layer is:

$$\nabla J^{\mathrm{EV}} = \frac{\eta}{m}\sum_{j=1}^{d} o_j^e \left(P^T - \lambda_j I\right)\left(P - \lambda_j I\right).$$

For new data points, the encoder performs out-of-sample extension, based on the mapping learned from the training points. This extension is computationally efficient as it relies on a few affine transforms and non-linear element-wise functions. In addition, once trained, it does not depend at all on the training data, making it efficient also in terms of memory.

Note that for a high value of η, the EV constraint forces the output to the origin, which is a trivial solution to minimizing this constraint. To drive the solution away from the origin, an "orthogonality" constraint can be added, such that $o^e(o^e)^T = \Psi\Psi^T$. For symmetric affinity matrices as in spectral clustering or Laplacian Eigenmaps this becomes $o^e(o^e)^T = I_d$, where I_d is a $d \times d$ identity matrix. In our simulations, adding this constraint to the encoder cost function did not have a meaningful impact, and it increased the training time while requiring fine-tuning of an additional cost parameter with respect to the other terms. Other constraints could be used as an alternative to the EV constraint we introduce, such as the Rayliegh quotient [23], which will be explored in future work.

3.4. Decoder

The decoder has the same architecture as the encoder, only in reverse. The input layer is $a^{(1)} = \Psi$, with size $s_1 = d$, and $y = X$ with size $s_L = n$. There are no new constraints in the decoder cost, and it consists of a loss term and weight regularization term as in (6):

$$J^d(\theta^d) = \frac{1}{2m}\sum_{i=1}^{m}\|o^d(i) - x_i\|^2 + \frac{\mu}{2}\sum_{l=1}^{L-1}\|W'^{(l)}\|_F^2, \tag{10}$$

where $\theta^d = \{W'^{(l)}, b'^{(l)}\}_l$ denotes the set of the weights and biases of all the layers of the decoder, and $o^d(i) \in \mathbb{R}^n$ is the output for input $\Psi(x_i)$. Note that the decoder weights $\{W'^{(l)}\}_l$ are not tied to those of the encoder $\{W^{(l)}\}_l$, therefore the number of units in the hidden layers can be different than for those in the encoder. Enforcing tied weights in the autoencoder will be explored in future work.

The decoder solves the pre-image problem and has several interesting applications. As the decoder learns the non-linear inverse mapping between the data and the diffusion space, it enables to output points in the data space given new points in the diffusion space. This enables data visualization, performed by covering the diffusion space with new points, and inputting these points to the decoder. This also enables to perform data augmentation for increasing the training set in machine learning applications. Another benefit of the

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

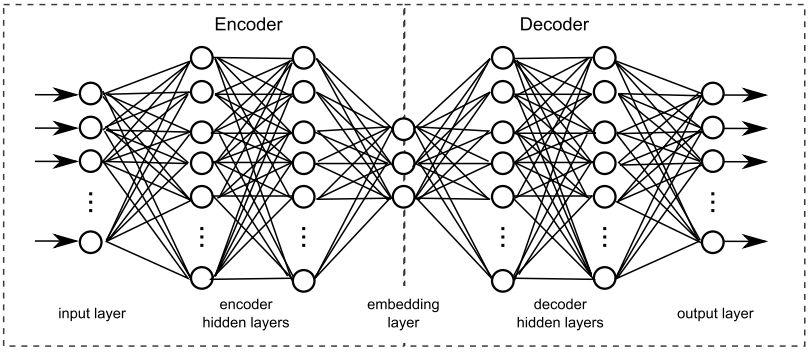

Fig. 1. Autoencoder. Left: encoder with two hidden layers, Right: decoder with two hidden layers. The output of the encoder is a prediction of the diffusion map. The output of the decoder is a reconstruction of the data.

decoder is the ability to perform calculations in the embedding space and then pull back to the data-space, for example, calculating centroids in a clustering application, or interpolation of points in the embedding space. In our current formulation we do not impose constraints on the decoder, however, in future work we will explore including additional constraints on recovering the data such as a harmonic constraint. This will enable recovering interesting surfaces, such as "minimal surfaces", from the embedding.

3.5. Autoencoder

Having trained the encoder and the decoder, the two networks are stacked together to obtain an autoencoder. This architecture is displayed in Fig. 1. The network is composed from two stacks, one for the encoder and one for the decoder. In most of our experiments each stack has 2 hidden layers, and in some we use simpler stacks of a single hidden layer.

One application of the autoencoder is to use the autoencoder reconstruction error to detect outliers in the data. Denoting the output of the autoencoder by r, the training data provides an average reconstruction error:

$$\epsilon = \frac{1}{m}\sum_{i=1}^{m} \|x_i - r(x_i)\|^2. \tag{11}$$

This enables setting a threshold on new test points to determine whether they fit the model of the data learned by the autoencoder. If the reconstruction error for a new point x' is greater than this threshold, the point is considered an outlier and its OOSE can be disregarded. This condition is given by

$$\|r(x') - x'\|^2 > C\epsilon,$$

where C is a constant determined by the user and the training data. Performing outlier detection within the framework of OOSE was previously addressed in [14], where a Mahalanobis distance in the embedding space was used for anomaly detection. Our approach differs in that our measure depends on the reconstruction of the point in the data space and not in the embedding space. This approach is similar to anomaly detection via dictionary learning, where a high reconstruction error of a test point indicates that the learned dictionary does not represent the point and it is therefore an outlier [40].

A second application of the autoencoder is denoising. When applied to noisy data, the diffusion map recovers a smooth manifold. Noise in the data, which relates to the relaxation time in the diffusion process, is typically manifested in eigenvectors relating to small eigenvalues [26]. These are usually disregarded when choosing the number of dimensions to retain in the embedding. Thus, noisy points which relate to the same original clean point are embedded at identical coordinates, and this mapping is learned by the encoder. The

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

decoder, which learns the inverse mapping from the embedding to the data, can only recover a smooth version of the data. Variations in the recovered data are such that they are along the low-dimensional manifold, whereas variations due to noise, that are not "seen" by the manifold, are suppressed. Thus, inputting new noisy points into the autoencoder, outputs a clean version of the points.

From a deep learning perspective, the embedding in the middle layer of the autoencoder serves as a regularization that promotes denoising, as the contractive penalty is used for regularization in contractive autoencoders [22]. From a manifold learning perspective, the same mechanism we propose here can also be applied to perform denoising with other paired techniques of out-of-sample extension and pre-image computation. This is also true for the outlier detection scheme. However, our approach provides a single general framework with which to perform these tasks.

Vincent et al. [21] provide an interpretation of their denoising autoencoder based on the manifold assumption, however they do not explicitly incorporate the manifold in their training procedure. In addition, they are intentionally corrupting the data by adding noise to the input and then training the network to recover the clean data. We do not assume the data is clean, but that the embedding provides a smooth representation of the data.

4. Practical considerations

This section provides implementation details regarding the optimization of the diffusion net and the computational complexity of our approach.

4.1. Pre-training

In general neural nets, a greedy layer-wise pre-training procedure was first developed by Hinton, Osindero and Teh [17] for Restricted Boltzman Machines, and later extended to autoencoders [19]. This procedure was shown to improve the optimization of the neural network. In this approach, each layer is first trained in an unsupervised manner as an autoencoder whose input is the output of the previous layer. This enables to "initialize" the parameters $\theta = \{W^{(l)}, b^{(l)}\}_l$ of the network before performing backpropagation, termed fine-tuning, over the complete network for the supervised learning task. It was shown empirically that this procedure yields improved results compared to initializing with random weights which can get stuck in poor local minimum solutions.

Recently, deep networks trained using very large quantities of labeled data have achieved successful results, with pre-training having very little impact, if at all. It appears that employing large amounts of data in the training cancels the advantage gained by pre-training [20]. However, in our setting, the amount of the data used to train the network is not large, therefore pre-training is beneficial.

We pre-train every hidden layer as a denoising autoencoder [21] with a sparsity term. This term encourages the average activation of every hidden unit to be small so that the hidden representation of the data is sparse [41]. The loss function is given by

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m} \|\hat{\widetilde{x}}_i - x_i\|^2 + \frac{\mu}{2}\sum_{l=1}^{2} \|W^{(l)}\|_F^2 + \beta \sum_{j=1}^{s_1} \mathrm{KL}(\rho \| \hat{\rho}_j), \tag{12}$$

where $\theta = W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}$. The input point is x_i, $\widetilde{x}$ is x corrupted by setting a random subset of entries to zero and $\hat{\widetilde{x}}_i$ is the reconstruction of x_i by the autoencoder. The third term is the sparsity loss where s_1 is the number of hidden units, ρ is a sparsity parameter, typically a small value close to zero, $\hat{\rho}_j$ is the average activation of hidden unit j and $\mathrm{KL}(\rho\|\hat{\rho}_j)$ is the Kullback–Leibler divergence between the probability mass functions of Bernoulli random variables with parameters ρ and $\hat{\rho}_j$. This constraint imposes that each hidden units "responds" only to specific input patterns in the data.

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

4.2. Optimization

The cost function in neural networks is highly non-convex. However, convex optimization methods are used, as typically a local minimum yields good results. We perform training with Limited memory BFGS (L-BFGS) [41,42] with line search procedure, a quasi-Newton method implemented in Matlab's *minFunc* package, and the gradients are computed using standard back-propagation [38]. L-BFGS is a suitable approach when the dataset is not too large, and even out-performs SGD, which is a more popular approach in deep learning, in certain settings [41].

The weights and biases in all layers are initialized with random values from a normal distribution with zero mean and small variance. The weight regularization parameter μ was set to be small, i.e. order of 10^{-7} or 10^{-10}. In the experimental results in Section 6 we examine the performance of the new constraint (8) for different values of η.

4.3. Mini-batch training

Mini-batch training is typically used with large datasets in deep learning and has been shown to improve performance. In this method, a random subset of samples is chosen for each iteration (or number of iterations) and the gradients are computed and averaged only for this small subset. Since the constraint we added (8) entails multiplying the output of the encoder by the normalized random walk matrix, it is essentially "mixing" between the outputs of different training points. This is problematic in mini-batch training, since a point's local neighborhood which most influences the eigenvector constraint will not necessarily be included in the mini-batch. If extracting only a subset of points, a new matrix needs to be calculated for the mini-batch by extracting the relevant rows and columns, and renormalizing the new matrix. In this case, this constraint is reduced to the output being only an approximation of the eigenvector of the full random walk matrix. However, since the size of training data used in manifold learning is usually not large due to computational complexity constraints, it is unnecessary to train using mini-batches. In this our setting differs from typical large-scale applications in deep learning such as Imagenet, CIFAR-100, etc.

4.4. Computational complexity

Training is an iterative sequential process, performed offline, as in other OOSE methods [9,11,12]. The training time to convergence depends on the cost parameters, initialization and size of the training set. To overcome the sensitivity of the optimization to the random initialization of the network parameters, we employ greedy pre-training (Sec. 4.1) to improve the convergence of the network. In the following we report the computational and memory cost of processing new points. Once the encoder has been trained, out-of-sample extension on new data is calculated with complexity $\mathcal{O}(\sum_{l=1}^{L-1} s_l s_{l+1})$ where $s_1 = n$ and $s_L = d$. In our experiments, we used an encoder with two hidden layers so the complexity was of order $\mathcal{O}(ns_2 + s_2 s_3 + s_3 d)$. This is opposed to other OOSE methods in which the complexity depends on m, the number of training points, where typically, $m \gg n > d$. For example, the complexity of OOSE using a naïve implementation of Nyström is $\mathcal{O}((d+n)m)$, as it requires calculating the distance to all training points. This cost can be reduced using an accurate or approximate k-nearest-neighbor search [43,44], however their efficiency depends on the dimensionality and distribution of the data. In terms of memory requirements, these algorithms still rely on retaining all the training points and may even incur increased memory requirements for efficient search [44].

An advantage of our method is that there is no need to retain the training points and embedding, once the network has been trained. Only the weight matrices and bias vectors of all the layers of the network are necessary. Thus, our approach requires memory on the order of $\mathcal{O}(\sum_{l=1}^{L-1} s_l s_{l+1})$. Other methods, on the other hand, require retaining all training points and embeddings. For Nyström and Geometric Harmonics [9]

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

this results in memory on order of $\mathcal{O}((d+n)m)$. The memory cost of the PCA-based approach in [14] is higher, requiring an additional $\mathcal{O}(md^2)$ to save covariance matrices for the embeddings of the training points. If using a non-Euclidean metric as in [34,36], retaining the covariance matrices of the training points leads to an additional memory cost on the order of $\mathcal{O}(mn^2)$. Thus, our method has a large advantage in applications and systems in which the memory and run-time are limited.

5. Bounding the out-of-sample-extension error

In this section, we provide a theoretical bound on the error rate for approximating eigenfunctions of the Laplacian using a single layer network of sigmoid hidden units. The full derivation is given in Appendix A. Suppose $\mathcal{M}$ is a smooth compact d-dimensional submanifold of $\mathbb{R}^n$. Assume $\mathcal{M}$ is equipped with a metric $\rho : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^+$ which is locally bi-Lipschitz with respect to the Euclidean metric, meaning $\exists \delta, \epsilon$ such that

$$(1-\epsilon)\|x-y\|_2 \leq \rho(x,y) \leq (1+\epsilon)\|x-y\|_2, \text{ for } \|x-y\| < \delta.$$

Theorem 5.1. *Let $\mathcal{M}$ be a smooth compact submanifold of $\mathbb{R}^n$ equipped with a metric which is locally bi-Lipschitz with respect to the Euclidean metric. Let B_r be a Euclidean ball of radius r such that $\mathcal{M} \subset B_r$. Let ψ be an eigenfunction of the Laplacian of $\mathcal{M}$ with eigenvalue λ, and let $f : \mathbb{R}^n \to \mathbb{R}$ be an extension of the eigenfunction to the ambient dimension via*

$$f(x) = \exp(-\lambda \|x - P_{\mathcal{M}}x\|_2^2)\psi(P_{\mathcal{M}}x), \text{ where } P_{\mathcal{M}}x = \arg\min_{y \in \mathcal{M}} \|x-y\|_2. \tag{13}$$

Then there is a linear combination f_K of K sigmoidal units such that

$$\left(\int (f(x) - f_K(x))^2 dx\right)^{\frac{1}{2}} \leq \frac{C}{\sqrt{K}}. \tag{14}$$

Corollary 5.2. *Under the same conditions as Theorem 5.1, let $\psi_1, ..., \psi_d$ be eigenfunctions of the Laplacian and f_i be the extension of ψ_i. Let $f(x) = (f_1(x), ..., f_d(x))$. Then there exists a single hidden layer network with Kd sigmoidal units and output $o(x) \in \mathbb{R}^d$ such that*

$$\left(\int (f(x) - o(x))^2 dx\right)^{\frac{1}{2}} \leq \frac{C}{\sqrt{K}}. \tag{15}$$

While our theorem addresses the true Laplacian, our implementation considers the empirical Laplacian, which converges to the true Laplacian given infinite training data uniformly sampled from the manifold. In addition, Theorem 5.1 guarantees existence of a solution, yet since the optimization problem is non-convex, it does not guarantee convergence. We are bounding the best in-class approximation error for a fixed size of parameters (network size), however results on whether gradient descent achieves this ideal are still lacking. There are recent results showing that, for deep feed-forward linear models, every local minimum is also a global minimum [45]. While this no longer applies when using non-linearities, a similar cost function landscape is conjectured to apply, thus implying a local minimum may attain similar approximation error to the best-in-class network.

6. Experimental results

In this section, we present experimental results for several toy problems and real image data. We demonstrate the performance of the encoder and decoder separately and then joining them in an autoencoder.

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

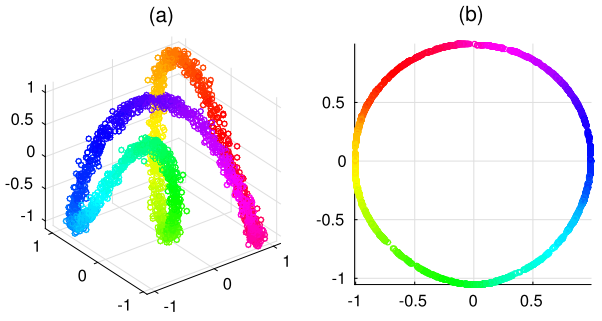

Fig. 2. (a) 3D closed curve. (b) First two coordinates of the diffusion map. Points colored by θ_i.

We evaluate the effect of the eigenvector constraint and demonstrate how it improves the performance of our system, especially in noisy scenarios. Finally, the autoencoder is successfully used in outlier detection in images. This demonstrates that our solution not only performs OOSE, but also means by which to verify that new data agrees with the model inferred from the training data.

6.1. Encoder

Our first example is a closed 3-D curve parametrized by:

$$x_i = (\cos(\theta_i), \sin(2\theta_i), \sin(3\theta_i))^T, \quad \theta_i \in (0, 2\pi). \tag{16}$$

We examine the effect of adding the eigenvector constraint (8) to the encoder cost function (9) for this toy problem. We address both the effect of the architecture of the network, i.e. number of layers and units, and the effect of noise in the data. We add white Gaussian noise ν_i with standard deviation (std.) $\sigma_\nu = 0.05$ to the data:

$$\tilde{x}_i = x_i + \nu_i, \quad \nu \sim \mathcal{N}(0, \sigma_\nu^2 I_{3\times 3})$$

and sample 2000 points from this noisy curve. The 2D diffusion map of these points is a smooth circle (Fig. 2).

To evaluate the performance of the encoder, we calculate a leave-one-out cross validation error as in [7]. We first calculated the embedding Ψ for $m = 2000$ training points. Then, for every point $\tilde{x}_i$, we calculated an embedding $\widetilde{\Psi}$ using the remaining $\tilde{m} = 1999$ points, and calculated the mapping between the two embeddings. A leave-one-out evaluation minimizes the inherent error due to mapping the training manifold embedding to the manifold calculated for the training and test set together, which grows with the number of added test points.

In our experiments, the data is closed curve, so that its continuous equivalent is the heat equation with Neumann boundary condition. Therefore, continuous solutions to this PDE are the trigonometric functions $\sin(\cdot), \cos(\cdot)$. For our discrete 3D data, the eigenvectors approximate these continuous periodic eigenfunctions. Specifically, the first two eigenvectors belong to the same eigenvalue and form a 2D circle in the embedding space. Therefore, two such embeddings calculated for the same data are similar up to a rotation. We calculated the rotation between both embeddings Ψ and $\widetilde{\Psi}$ using the 1999 shared points. This was calculated by [32]

$$S[i,j] = \sum_k \widetilde{\Psi}_i^T(k) * \Psi_j(k), \quad k \in \{1, ..., \tilde{m}\}\ i,j = 1,2. \tag{17}$$

The SVD of S is $U\Lambda V^T$. Then the rotation from $\widetilde{\Psi}$ to Ψ is given by

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

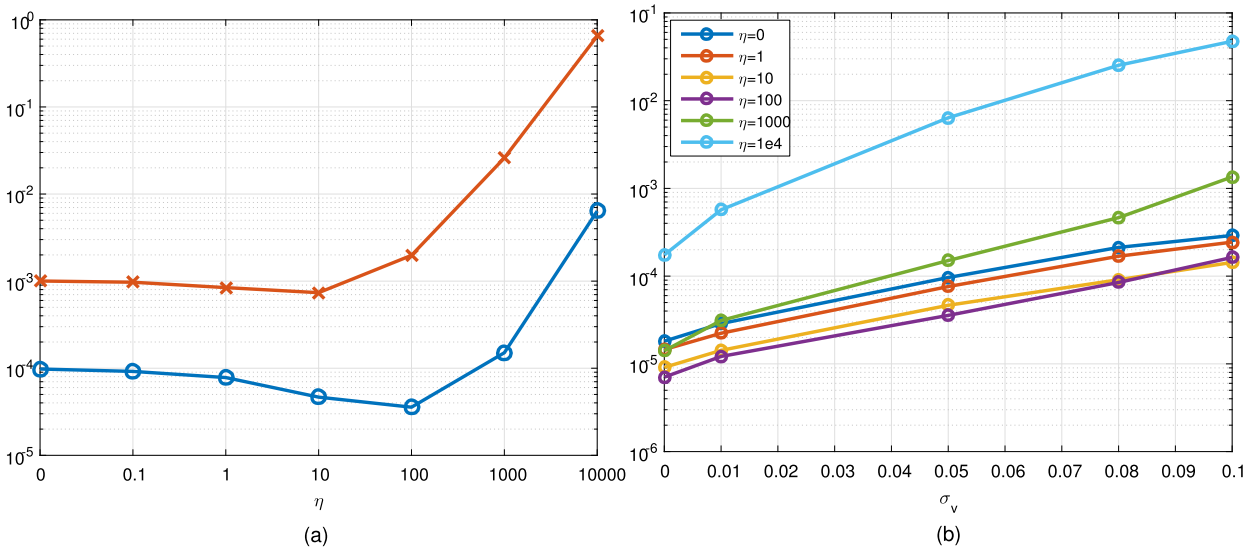

Fig. 3. Leave-one-out cross validation MSE of the encoder. (a) Comparing 1 hidden layer (red 'x') vs. 2 hidden layers (blue circle) for varying η values for fixed $\sigma_\nu = 0.05$. (b) Comparing various η values for increasing noise std. σ_ν.

$$R = VU^T. \tag{18}$$

We trained the encoder on the 1999 points and their corresponding embedding $\widetilde{\Psi}$, and extend the embedding to the excluded point $\tilde{x}_i$, denoted $\widetilde{\Psi}(i) = o^e(\tilde{x}_i)$. We then calculated the MSE error between the original embedding of $\tilde{x}_i$ and its rotated extension:

$$\|\Psi(i) - R\widetilde{\Psi}(i)\|^2.$$

We performed this for each point $\tilde{x}_i$ separately and then averaged the leave-one-out MSE error over all points.

Fig. 3(a) plots the MSE for various values of η for an encoder with 1 hidden layer and with 2 hidden layers. In Fig. 4, we plot several results from this experiment. The top row displays the output of the encoder with one hidden layer and the bottom row with two hidden layers. Column (a) is the result without the added constraint ($\eta = 0$). Columns (b)–(d) are for $\eta = 10,\ 100,\ 10000$ respectively.

A first conclusion from these simulations is that adding depth to the network improves the results. This follows known observations in deep learning, that functions that can be compactly represented by networks with L layers, can require an exponential number of units in a network with $L-1$ layers [19]. Using one hidden layer, the reconstruction of the embedding is noisy and the added constraint is not enough. Using two hidden layers, the results are significantly improved and the eigenvector constraint serves to smooth out the noise in the reconstruction. In terms of the MSE, adding a layer decreases the error by a factor of 10. When the cost parameter is too large as in Fig. 4(d), this constraint collapses the output of the encoder to the origin, which is the trivial solution to minimizing this constraint. The optimal results were achieved for $\eta = 10, 100$. Not imposing the EV constraint ($\eta = 0$) yielded an error that was twice as high and results in a noisy approximation.

We note that similar results regarding the advantages of depth were observed in training an *autoencoder* for this data. Adding a layer to the autoencoder decreases the reconstruction MSE by a factor close to 10. Thus, using 2 layers in the network improves both encoder and autoencoder errors.

To examine the effect of noise on the approximation, we ran simulations with varying values of σ_ν, and present the results in Fig. 3(b). The encoder was trained with 2 hidden layers, with 20 hidden units in each layer. We calculate the leave-one-out MSE, comparing several encoders with different value of η in the EV

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

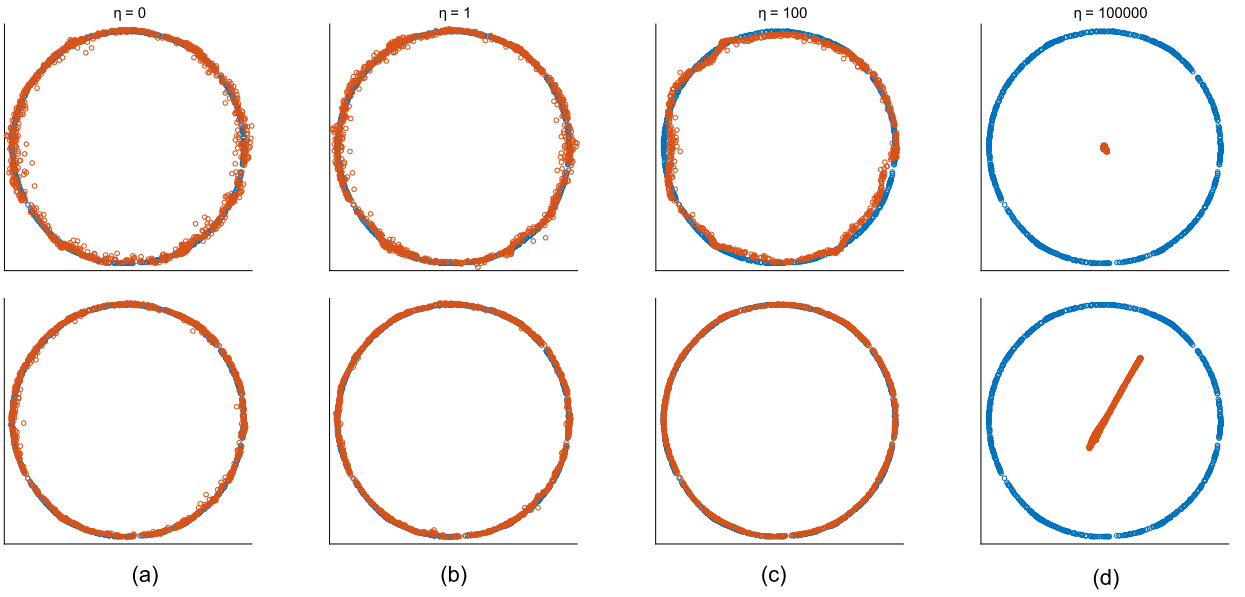

Fig. 4. Top row: Encoder with 1 hidden layer, Bottom row: Encoder with 2 hidden layers. Blue plot is original diffusion map, red plot is output of the encoder. The columns examine the effect of the eigenvector constraint for increasing values of η. (a) $\eta = 0$, (b) $\eta = 10$, (c) $\eta = 100$, (d) $\eta = 10^5$. If η is too large, the eigenvector constraint dominates the cost function and the output collapses to the trivial solution that all points equal zero. For a one hidden layer encoder, the eigenvector constraint is not enough to smooth out the noise and obtain a good embedding. For a two layer net, increasing η within a reasonable range smooths the noise and yields an improved output compared to not including this constraint. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

constraint. The best results are obtained by $\eta = 100$ up to $\sigma_\nu = 0.1$, where $\eta = 10$ has the lowest error. Even for no noise ($\sigma_\nu = 0$), including the EV constraint with proper η is meaningful for estimating the diffusion embedding ($\eta = 0$ has the second highest error for all σ_ν values). If is η is too high, the solution collapses to zero, resulting in a high error. For low σ_ν, $\eta = 1000$ performs well so there is some trade-off between the SNR and how high η can be.

To examine the dependence of the mean error on the number of units in the single layer encoder, we trained the network with a varying number of hidden units $s_1 \in \{5, ..., 60\}$, for $\eta = 100$, given clean data, i.e. $\sigma_\nu = 0$. Fig. 5 shows that the empirical mean error e is bounded by $\frac{C}{s_1}$, so that empirically we are achieving a tighter bound than the bound guaranteed in Corollary 5.2.

In Fig. 6(a) we compare the encoder to competing OOSE methods: Nyström, Geometric Harmonics (GH) [9] and Auto-adaptive Laplacian Pyramids (AALP) [12] (an improved version of [11]), for varying noise. We also examine how the number of hidden units affects the performance of the encoder. We compare three DN encoders with two hidden layers, which were trained using $\eta = 100$:

- DN-20-10: $s_1 = 20$ and $s_2 = 10$ hidden units
- DN-20-20: $s_1 = 20$ and $s_2 = 20$ hidden units
- DN-40-20: $s_1 = 40$ and $s_2 = 20$ hidden units

The DN-40-20 encoder outperforms all competing methods for all noise std. For no noise ($\sigma_\nu = 0$) AALP has the second lowest error, but there is a decrease in performance when noise is introduced. Geometric Harmonics does better than Nyström for low noise, but also shows a significant decrease in performance as the noise grows. Nyström is least dependent on noise in this scenario but does poorly for low noise. Finally, the results demonstrate that using more hidden units improves the OOSE as the encoder is able to learn a more accurate mapping.

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

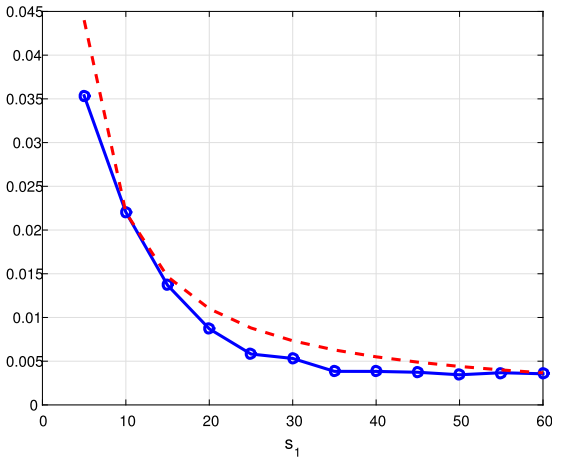

Fig. 5. Embedding estimation error vs. number of hidden units s_1 in 1-layer encoder (solid line). The empirical error is bounded by $\frac{C}{s_1}$ (dashed line), which is a tighter bound than the theoretical bound $\frac{C}{\sqrt{s_1}}$ guaranteed in Corollary 5.2.

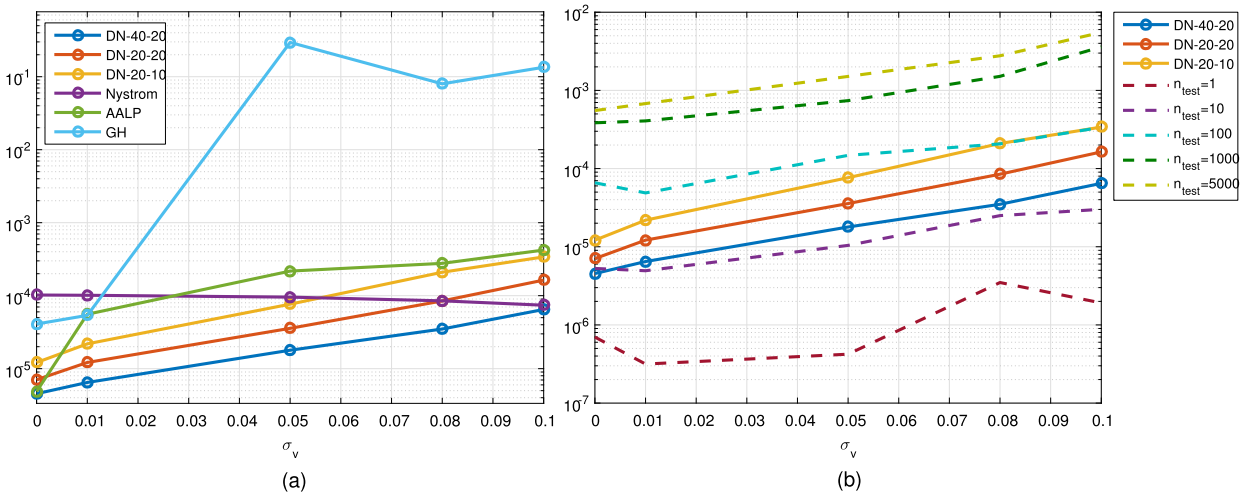

Fig. 6. (a) Out-of-sample extension MSE vs. σ_ν: Comparing three DN encoders to competing methods. (b) Out-of-sample extension error vs. variation in the in-sample embeddings. Dashed plot is $\epsilon_{n_{\text{test}}}$, solid line is DN encoder MSE error.

We also compare the OOSE error to the inherent mapping error between embeddings, due to increasing the number of points for which the embedding is calculated, similar to [7]. The purpose of an out-of-sample extension algorithm in the manifold learning setting is to provide an extension of the embedding to new points, such that this extension on the new points is close to the embedding of these points if the embedding was calculated over all the points. Due to the discrete nature of the data, there is no "true" embedding; rather the value of the eigenvectors for a given training point depends on the other points in the set and the scale used in the affinity matrix. We calculate the mapping error between two embeddings as follows.

We sample $m = 2000$ training points along the curve and calculated the diffusion embedding Ψ for these points. We then added n_{test} points to the training points, and calculated an embedding $\widetilde{\Psi}$ for all $\tilde{m} = m + n_{\text{test}}$ points. We calculated the rotation between both embeddings using the shared points, i.e. the m training points, following (17)–(18), and calculated their average mapping error:

$$\epsilon_{n_{\text{test}}} = \frac{1}{m} \sum \|\Psi(k) - R\widetilde{\Psi}(k)\|^2, \quad k \in \{1, ..., m\}.$$

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

We calculated this error over 10 realization of the data and for 5 σ_ν values. This error is the variation in the in-sample embedding due to increasing the number of points for which the embedding is calculated. Fig. 6(b) compares this error to the leave-one-out OOSE error of the DN encoders, trained using $m = 2000$ points and $\eta = 100$. For $n_{\text{test}} > 100$, the out-of-sample extension MSE of our encoders is lower than the variation in the in-sample embeddings. Since, OOSE will typically be performed for n_{test} much higher than 100, this demonstrates that our method provides a good out-of-sample extension.

Regarding the parameters used in the encoder simulations, the number of units in the hidden layers in the simulations was $s_l = 20$, unless stated otherwise. The cost parameter of the weight regularization term was set as $\mu = 10^{-10}$. For the sparsity constraint in pre-training the auto-encoder (12), we tried various values of β for $\rho = 0.1$ and they do not affect the errors in any significant manner. Therefore, we set $\beta = 1$.

6.2. Decoder

The decoder solves the pre-image problem. It can be used to extend data from the diffusion embedding space to the data space, thus creating new points that lie on the manifold. This enables a better visualization of the data belonging to the manifold, in the data space.

Our first example is based on the clean version of the 3D curve, given in (16). We draw 2000 random samples from this curve. The diffusion map is calculated with $\sigma = 0.1$ for the scale in the kernel (1). The first two diffusion coordinates are a circle, as in the previous example. We train a decoder using these 2000 samples and their embedding. We then cover the diffusion space with points enclosed within a circle extended beyond the radius of the embedding and use the decoder to predict the data in the 3D data space.

Fig. 7(a) displays the points in the embedding space colored by angle (top) and radius (bottom). Fig. 7(b) presents two different views of points predicted by the DN decoder in the 3D data space, where the colors correspond to the color of the points in the diffusion space. We can see the points are restricted to a 2D surface enclosed within the 3D curve. Note that the embedding of the origin is handled smoothly, with no singularities in the data space. Also note that points in the diffusion space which are located beyond the original circle (points colored in red in the bottom plots), are decoded in the data space along the boundary of the surface. Thus, the range of extension is limited and follows the geometry of the original data. In comparison, Fig. 7(c) and (d) present a pre-image calculated by weighted averaging using an affinity kernel with a fine and coarse scale, respectively. For a fine scale, the range of extension is limited close to the 3D curve, with points with a distance much larger than the scale collapsing to the origin. For a large scale, the points along the original embedding are no longer mapped along the 3D curve, but within it, resulting in a high data reconstruction error.

The next example is of more complex data, whose embedding is also a 2D circle. Given an image of a noisy periodic function (std. of the noise is 25), shown in Fig. 8(a), we extract all overlapping patches sized 5×5 pixels, and construct a random walk on the patches, to obtain the diffusion map Ψ. The first two coordinates of the diffusion embedding are a circle shown as the blue points in Fig. 8(b). Training the decoder with two hidden layers for the diffusion map and the patches, yields a function that "produces" image patches. The input dimension is $s_1 = 2$ and the output dimension is $s_L = 25$. The decoder enables visualization of the data. Fig. 8(b) displays the patches obtained by inserting several test points from the diffusion space into the decoder, where the position of the patches in the circle corresponds to the location of the points entered into the decoder. This example shows that the radius in the diffusion embedding represents the amplitude of the periodic function in the image patches. At the origin of the diffusion space, we get a smooth patch. Note the patches are clean as opposed to the training patches.

Fig. 8(c) shows how this can be used for image manipulation. A rotation of π and scale by 0.45 is applied to the diffusion map as

$$\widetilde{\Psi} = 0.45 \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \Psi.$$

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

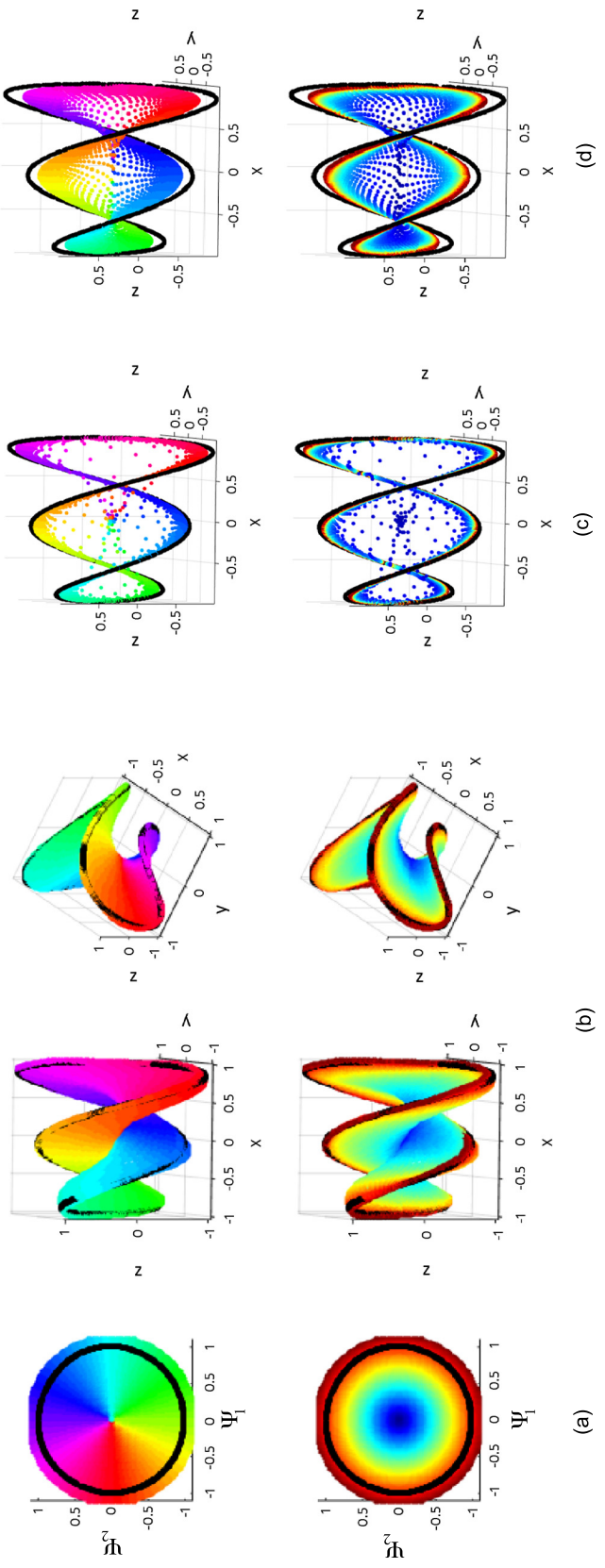

Fig. 7. (a) Diffusion space is covered in points within a circle. The black points are the diffusion coordinates for the training data and are the input points to the decoder. (b) Two different views of the output of the decoder, where the black points are the original training datapoints. (c) Output of weighted average for fine scale. (d) Output of weighted average for coarse scale. Top row: points colored by angle in the diffusion space. Bottom row: points colored by radius.

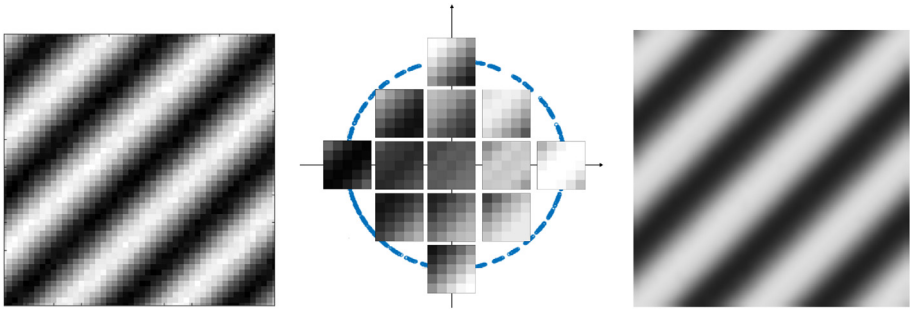

Fig. 8. (a) Image of noisy periodic function. (b) Decoding image patches. Blue points are the diffusion map of the training image patches. Various points along and within the circle are inputted to the decoder. We display the obtained patches at the locations in the diffusion space that were used an input to the decoder. This demonstrates that the radius of the embedding corresponds to the amplitude of the periodic function in the image space. (b) Image manipulation: the training diffusion map is rotated by 180 degrees and the values multiplied by 0.45. Inputting these points into the decoder and reconstructing the image from output, shows the period of the image has indeed shifted and the amplitude decreased. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The values of $\widetilde{\Psi}$ are inputted into the decoder. Then the output values are used to reconstruct the image where each value is assigned to its original pixel location. The resulting *clean* image is indeed a shift of the original image in Fig. 8(c) and the amplitude has been decreased.

6.3. Autoencoder

We now turn to the Diffusion Net autoencoder, which is trained following Algorithm 3. We compare the autoencoder to a Nyström-based OOSE and pre-image, extending the data to the manifold and pulling back to the data space. We begin with synthetic data, examining the effect of the ambient dimension of the data and the number of points used to train the network. The high-dimensional data is defined by pairs of sine and cosine function, whose period increases by 2π for increasing dimension:

$$\begin{aligned} x_i[2k-1] &= \cos(2k\pi\theta_i) \\ x_i[2k] &= \sin(2k\pi\theta_i) \end{aligned} \tag{19}$$

where $\theta_i \in (0,1)$, and we run simulations for increasing k. As the dimension of the data increases, the complexity of the data increases as well, becoming more and more oscillatory. We add i.i.d. Gaussian noise with std σ_ν to each high-dimensional point, obtaining a training set $X_{\text{train}} = \{\tilde{x}_i\}_{i=1}^{N_{\text{train}}}$. The diffusion embedding of this data is a 2D circle. The reconstruction of the data is tested on a test set $X' = \{\tilde{x}'_j\}_{j=1}^{m}$, where $m = 5000$ points.

In Fig. 9, we present the denoising capabilities of the autoencoder, compared to the Nyström-based reconstruction of the data. The DN autoencoder was trained with 2 hidden layers in the encoder, and 2 hidden layers in the decoder, with 20 hidden units in each layer. Following the experimental results of the encoder, we set $\eta = 100$ in the EV constraint. We compare performance for three different noise levels: $\sigma_\nu = \{0, 0.1, 0.2\}$, and we average the results of 10 realizations of the data for each simulation. The reconstruction MSE is given by

$$\epsilon = \frac{1}{m} \frac{\sum_{j=1}^{m} \|x'_j - r(\tilde{x}'_j)\|^2}{\|X'\|_F},$$

where we divide by the Frobenius norm of the set X' to normalize the effect of increasing the dimension of the data. Note that we are comparing the output of the autoencoder to the *clean* data x'_j, although the

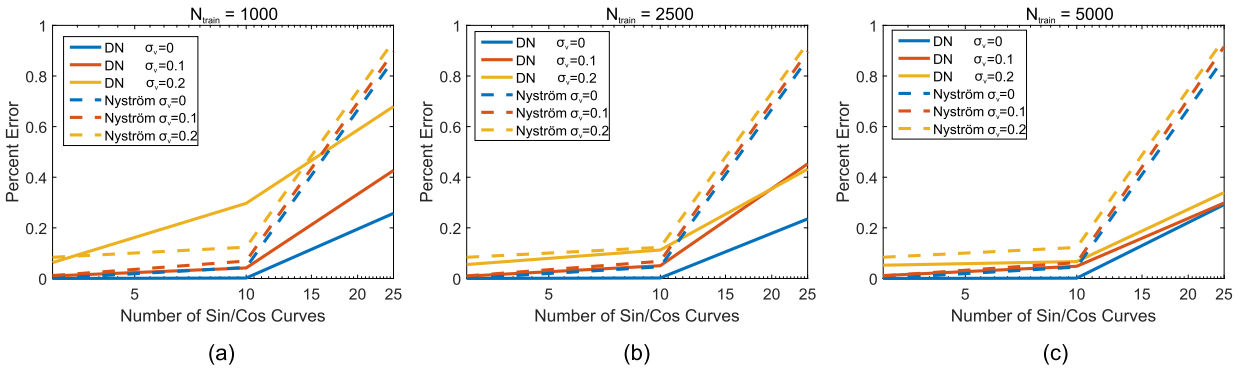

Fig. 9. Reconstruction MSE of the autoencoder for data with increasing noise, where the network is trained using $N_{\text{train}} = 1000$ (a), 2500 (b), 5000 (c) points.

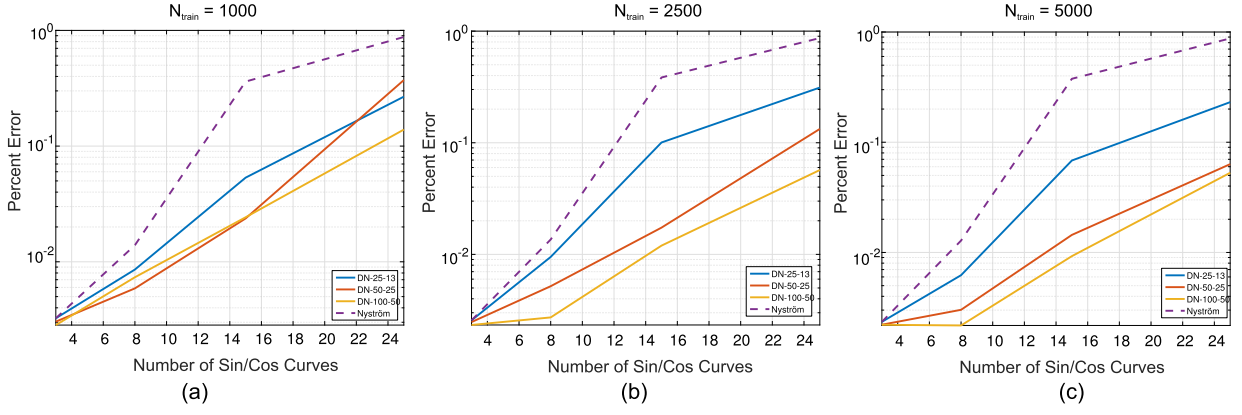

Fig. 10. Reconstruction MSE of the autoencoder for increasing complexity of the network (number of hidden units) for data with fixed noise $\sigma_\nu = 0.05$. The network is trained using $N_{\text{train}} = 1000$ (a), 2500 (b), 5000 (c) points.

autoencoder is trained using the noisy data $\{\tilde{x}_i\}_i$. Aside from a configuration of low number of training points ($N_{\text{train}} = 1000$) and high noise std ($\sigma_\nu = 0.2$), the autoencoder demonstrates better denoising for all noise std, compared to the Nyström-based reconstruction. Increasing the number of training points improves the denoising of the diffusion net, as expected with a neural network, whereas Nyström is mostly unaffected. Note that increasing the number of training points does not increasing the memory and computational requirements of the Diffusion net, as opposed to Nyström.

In Fig. 10, we examine the effect of the number of units in the hidden layers (the complexity of the network) on the denoising of the data. We compare three configurations of the network, denoted DN-100-50, DN-50-25 and DN-25-13, where $s_1 = \{100, 50, 25\}$ and $s_2 = \frac{s_1}{2}$ in the encoder, and vice-versa for the decoder, i.e. $s_2 = \{100, 50, 25\}$ and $s_1 = \frac{s_2}{2}$ in the decoder. We compare the autoencoder to the Nyström-based approach for fixed noise $\sigma_\nu = 0.05$, and increasing number of training points. The Diffusion net again out-performs Nyström. For a low number of training points ($N_{\text{train}} = 1000$) and a high number of hidden units (DN-50-25), the error of the network increases, indicating an over-fit of the training data with respect to the number of optimized parameters in the network. For DN-50-25 and DN-100-50, increasing the number of training points significantly improves the MSE. The performance of DN-25-13 is less affected, as perhaps the network has too few parameters to gain from the additional training data. As before, Nyström is mostly unaffected by the increase of number of training points. In both simulations, the performance of Nyström greatly degrades with the increasing complexity of the data (the ambient dimensionality).

Finally, we test the autoencoder on recovering digits in the MNIST [46] dataset. The dimensionality of the data is now very high $n = 784$ as compared to the synthetic data we previously addressed. We calculate

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

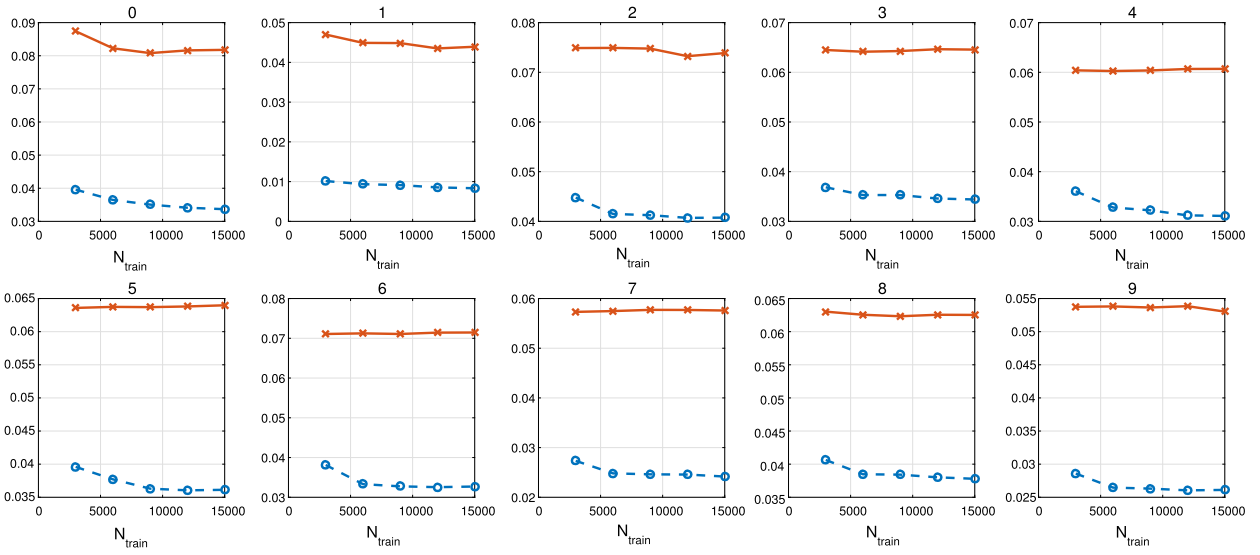

Fig. 11. Reconstruction MSE per digit for MNIST for varying N_{train} training points. Blue 'o' plot is DN autoencoder, Red 'x' plot is Nyström.

the embedding and train the network for N_{train} points, and compare the reconstruction of $m = 60000$ test points by the DN autoencoder to Nyström. In Fig. 11 we present the reconstruction MSE for each of the 10 digits vs. N_{train}. In all cases, the DN autoencoder out-performs the Nyström-based recovery. In addition, aside for digits 0,1 and 2, as before, Nyström is unaffected by the number of training points, whereas the performance of the autoencoder improves for all digits with additional training data.

6.4. Outlier detection

We now apply the autoencoder to real image data and demonstrate that the autoencoder performs outlier detection. As stated in Sec. 3, test data will not necessarily follow the model of the data used to calculate the embedding. OOSE applied to new data that significantly differs from the training data will assign embedding values that do not distinguish it from the data. Therefore, it is important to be able to determine when the embedding is unreliable.

Figs. 12(a) and 13(a) display two images of patterned semiconductor wafers acquired by a scanning electron microscope, sized 200×200 pixels. Both wafer images have a defect near the center of the image. For each image separately, we randomly extract 2500 patches from the image, sized 8×8 pixels. This training set is used to calculate a diffusion map, reducing the data from dimensionality $n = 64$ to $d = 2$ dimensions. The training set is used to train an autoencoder, as in Algorithm 3, and the average training reconstruction error ϵ (11) is calculated. We then input all overlapping image patches from the image into the autoencoder and calculate the reconstruction error of each patch. Figs. 12(b) and 13(b) display $\|r(x') - x'\|/\epsilon$ for all pixels in the image, revealing that this approach easily separates the defects from the background.

This is a result of the diffusion map capturing the main structure of the data, i.e. the pattern of the wafer, as represented in the training data. Patches which differ from the training set, as in the case of the defects, are not represented in the diffusion map. Thus, when applying the autoencoder, these patches are not properly reconstructed by the mappings learned from the data space to the diffusion space and back. This result obtained by the autoencoder indicates that, for these patches, the embedding provided by the encoder does not properly represent them.

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

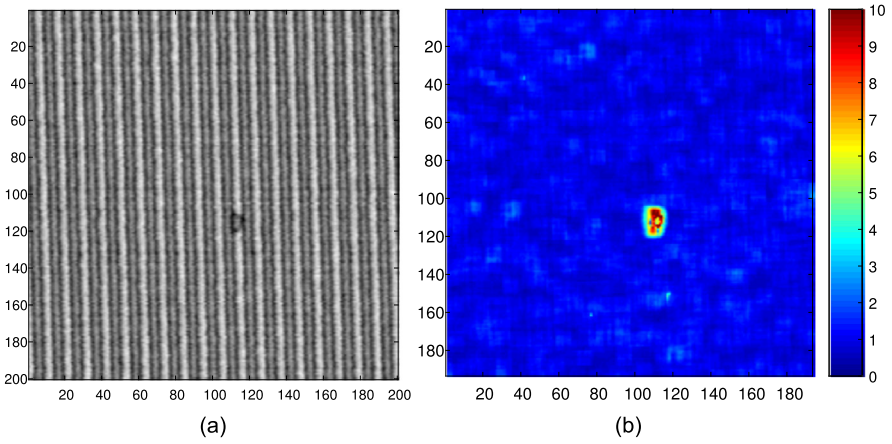

Fig. 12. (a) SEM image of a semiconductor wafer with defect near the middle of the image. (b) Reconstruction error of the image relative to the average training error: $\|r(x') - x'\|^2/\epsilon$. This reveals the wafer defect. Color version of figure is available online.

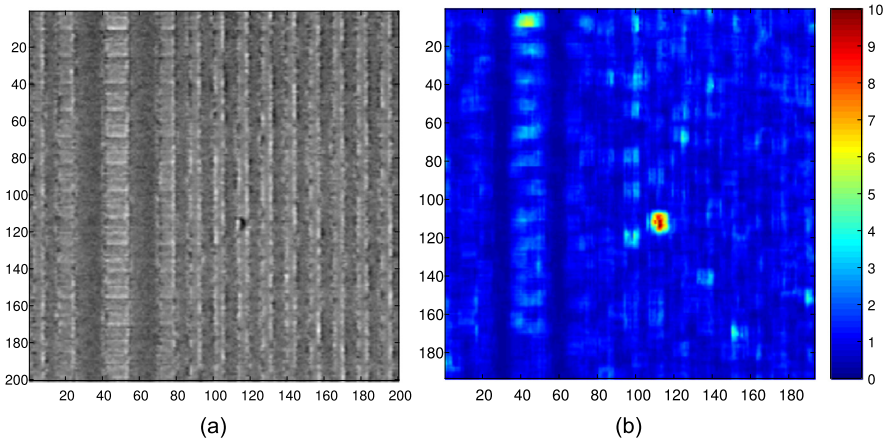

Fig. 13. (a) SEM image of a semiconductor wafer with defect near the middle of the image. (b) Reconstruction error of the image relative to the average training error: $\|r(x') - x'\|^2/\epsilon$. This reveals the wafer defect. Color version of figure is available online.

7. Conclusions and future work

We have presented a new framework employing deep learning for manifold learning. We proposed designing an encoder and decoder that learn the mapping between a given high-dimensional dataset and low-dimensional embedding, and vice-versa. To this end, we proposed a new constraint in training the encoder, which preserves the locality of the points in the embedding. We demonstrated empirically that this constraint improves the approximation of the embedding. Our encoder enables very efficient out-of-sample extension of the non-linear embeddings to new points, with low memory costs. The decoder provides a solution to the pre-image problem, enabling data visualization and augmentation. Finally, stacking the two networks together as a deep autoencoder enables both denoising and outlier detection of the data, as seen via the embedding. Calculating the reconstruction error of the autoencoder for new points allows to evaluate whether the OOSE provided by the encoder properly represents these new points. We presented experimental results in noisy scenarios for simulated and real data, demonstrating the properties of the proposed architecture and out-performing competing methods.

Our main focus has been on the encoder for performing out-of-sample extension for data whose distribution follows the distribution of the training data. However, in different applications, such as sequential signal processing, the nature of the data can change over time. In manifold learning, the embedding is usu-

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

ally calculated once for training points, and does not adapt over time for new points, as opposed to online dictionary learning in sparse representations, for example. This could lead to the embedding not providing a good representation of the data as it evolves, and requires re-calculating the embedding again and again. In future work, we propose to develop a method based on online fine-tuning of the autoencoder that will adapt the embedding to new points which do not fit the model of the training data. Instead of performing "regular" fine-tuning of the autoencoder, constraints can be added that will maintain the middle layer as an approximation of the embedding, as we proposed with the encoder in this work. In this case, we will fine-tune with both the test and training data, where the training data regularizes the autoencoder so that its middle layer remains an approximation of the embedding for the training points. By fine-tuning the network so that it reconstructs the new test data, the middle layer should recover a new embedding for the test data. This adaptive approach will be explored in future work.

A second direction is to further explore the decoder and how including different regularizations affects the solution of the pre-image problem. Including a harmonic constraint for example should enable recovering a minimal surface as the example shown in Sec. 6.2. The error rate we provided on the encoder does not apply to the decoder as it requires the function that is being approximated to be band-limited, which does not hold for the decoder in a general case. In future work on the decoder, we intend to provide a theoretical analysis of the decoder, and to expand our theoretical results to multi-layer nets. Computing the pre-image is important in different applications in which interpolating the data by averaging in the high-dimensional data space is meaningless, such as the possibility of performing image texture synthesis. We will analyze datasets in which the high-dimensional data is more complicated and examine how this affects the required complexity of the decoder architecture.

A third research direction is to examine improving deep learning applications. Our network enables to determine the number of nodes needed to learn the geometry of the data and can be used to infer the maximal number of nodes needed to model the complexity of the system for an unconstrained neural net. In addition, we intend to explore whether incorporating our encoder into a deep network will improve deep neural networks. This is motivated by previous works that have shown that implicitly incorporating the manifold assumption in the construction of deep networks improves classification results. Therefore, we expect that explicitly including the embedding in the network via the encoder should be beneficial.

Acknowledgments

This research was supported by the Israel Science Foundation (grant no. 576/16). Alexander Cloninger is supported by NSF Award No. DMS-1402254. The authors thank Ronald Coifman, Ronen Talmon and Roy Lederman for helpful discussions and suggestions. The authors also thank the anonymous reviewers for their constructive comments and useful suggestions.

Appendix A. Proof of Theorem 5.1

Proof. The proof of the theorem requires two key results in the literature, relying on theorems by Barron [47] and Coifman and Lafon [9].

Theorem Appendix A.1. *(Theorem 1 from [47]) Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function with bounded first moment of its Fourier transform*

$$\int_{\mathbb{R}^n} |\widehat{f}(\omega)||\omega| d\omega \leq C_f.$$

Let $B_r \subset \mathbb{R}^n$ be a Euclidean ball around zero with radius r, which we assume contains our data. Then for every $n \in \mathbb{N}$ there exists a linear combination f_K of K sigmoidal units such that

$$\left(\int (f(x) - f_K(x))^2 dx\right)^{\frac{1}{2}} \leq \frac{C_1}{\sqrt{K}}, \tag{A.1}$$

where $C_1 = 2\pi C_f \mu(B_r)$*, and* μ *is the Lebesgue measure on* $\mathbb{R}^n$*.*

Now it suffices to show that the extension function f of ψ from (13) has a bounded first moment of its Fourier transform. To show this, we rely on a second result.

Theorem Appendix A.2. *(Proposition 11 from [9]) Let* $\mathcal{M}$ *be a submanifold in* $\mathbb{R}^n$ *and* $\psi : \Gamma \to \mathbb{R}$ *be an eigenfunction of its Laplacian with eigenvalue* λ*. Let* $\delta > 0$ *be an approximation level. Let* f *be an extension function as in* (13)*. Then there exists a band limited function* $b : \mathbb{R}^n \to \mathbb{R}$ *with band* $C_{\delta,\psi}\lambda$ *such that*

$$\frac{\int_{\mathbb{R}^n} (f(x) - b(x))^2 dx}{\int_{\mathbb{R}^n} (f(x))^2 dx} < \delta.$$

We need several more intermediate claims before addressing the result.

Claim Appendix A.3. *The function* f *from* (13) *is in* $L^2(\mathbb{R}^n)$*.*

Proof of Claim Appendix A.3. Since ρ is locally bi-Lipschitz,

$$\begin{aligned}
\int_{\mathbb{R}^n} f(x)^2 dx &= \int_{\mathbb{R}^n} e^{-2\lambda \|x - P_{\mathcal{M}} x\|_2^2} \psi(P_{\mathcal{M}} x)^2 dx \\
&\leq \int_{\mathcal{M}} \left(\int_{\mathbb{R}^{n-d}} e^{-2\lambda \|x - x'\|_2^2} dx \right) \psi(x')^2 dx' \\
&= C_{n-d,\lambda} \int_{\mathcal{M}} \psi(x')^2 dx' \\
&\leq \frac{C_{n-d,\lambda}}{1-\epsilon} \int_{\mathcal{M}} \psi(z)^2 d\rho(z) \\
&= \frac{C_{n-d,\lambda}}{1-\epsilon} \qquad \square
\end{aligned}$$

Claim Appendix A.4. *Let* b *be as in Theorem Appendix A.2, and* $\widehat{b}$ *be its Fourier Transform. Then* $b \in L^2(\mathbb{R}^n)$ *and*

$$\int |\widehat{b}(\omega)||\omega| d\omega < \infty.$$

Proof of Claim Appendix A.4. Clearly $b \in L^2(\mathbb{R}^n)$ since

$$\|b\|_{L^2} \leq \|f\|_{L^2} + \|b - f\|_{L^2}.$$

Because b is band limited with band $C_{\delta,\psi}\lambda$, meaning $\operatorname{supp}(\widehat{b})$ is contained inside a ball of radius $C_{\delta,\psi}\lambda$, then

$$\int |\widehat{b}(\omega)||\omega| d\omega = \int_{B_{C_{\delta,\psi}\lambda}} |\widehat{b}(\omega)||\omega| d\omega$$

$$
\begin{aligned}
&\leq \left(\int (\widehat{b}(\omega))^2 d\omega\right)^{\frac{1}{2}} \left(\int_{B_{C_{\delta,\psi}\lambda}} |\omega|^2 d\omega\right)^{\frac{1}{2}} \\
&= \left(\int (b(x))^2 dx\right)^{\frac{1}{2}} \left(\int_{B_{C_{\delta,\psi}\lambda}} |\omega|^2 d\omega\right)^{\frac{1}{2}} \\
&\leq \left(\|f\|_{L^2} + \sqrt{\delta}\|f\|_{L^2}\right) \left(\int_{B_{C_{\delta,\psi}\lambda}} |\omega|^2 d\omega\right)^{\frac{1}{2}} \\
&= \left(\|f\|_{L^2} + \sqrt{\delta}\|f\|_{L^2}\right) \left(\frac{n}{n+2}(C_{\delta,\psi}\lambda)^2 \mu(B_{C_{\delta,\psi}\lambda})\right)^{\frac{1}{2}} \\
&= \left(1+\sqrt{\delta}\right) \sqrt{\frac{n\mu(B_1)}{n+2}} \left(C_{\delta,\psi}\lambda\right)^{\frac{n+2}{2}} \|f\|_2,
\end{aligned}
$$

where $\mu(B_1)$ is the volume of a ball of radius 1 in $\mathbb{R}^n$. □

Now we prove Theorem 5.1. By setting $\delta = \frac{1}{K}$ in Theorem Appendix A.2, we combine the above results to show

$$
\begin{aligned}
\left(\int (f(x) - f_K(x))^2 dx\right)^{\frac{1}{2}} &\leq \left(\int (f(x) - b(x))^2 dx\right)^{\frac{1}{2}} + \left(\int (b(x) - f_K(x))^2 dx\right)^{\frac{1}{2}} \\
&\leq \frac{C}{\sqrt{K}},
\end{aligned}
$$

where

$$
\begin{aligned}
C &= \|f\|_2 + C_1 \\
&= \|f\|_2 + 2\pi\mu(B_r) C_b \\
&\leq \left(1 + 2\pi\mu(B_r)\left(1+\sqrt{\delta}\right)\sqrt{\frac{n\mu(B_1)}{n+2}}\left(C_{\delta,\psi}\lambda\right)^{\frac{n+2}{2}}\right)\|f\|_2 \\
&\leq \left(1 + 2\pi\mu(B_r)\left(1+\sqrt{\delta}\right)\sqrt{\frac{n\mu(B_1)}{n+2}}\left(C_{\delta,\psi}\lambda\right)^{\frac{n+2}{2}}\right)\sqrt{\frac{C_{n-d,\lambda}}{1-\epsilon}}.
\end{aligned}
$$

□

References

[1] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[2] J.B. Tenenbaum, V. de Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (Dec. 2000) 2319–2323.
[3] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
[4] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (6) (2003) 1373–1396.
[5] D.L. Donoho, C. Grimes, Hessian eigenmaps: new locally linear embedding techniques for high-dimensional data, Proc. Natl. Acad. Sci. USA 100 (2003) 5591–5596.
[6] R.R. Coifman, S. Lafon, Diffusion maps, Appl. Comput. Harmon. Anal. 21 (1) (July 2006) 5–30.
[7] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N.L. Roux, M. Ouimet, Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering, in: S. Thrun, L. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems 16, MIT Press, 2004, pp. 177–184.

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

[8] C. Fowlkes, S. Belongie, F. Chung, J. Malik, Spectral grouping using the Nyström method, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2) (Jan. 2004) 214–225.
[9] R.R. Coifman, S. Lafon, Geometric harmonics: a novel tool for multiscale out-of-sample extension of empirical functions, Appl. Comput. Harmon. Anal. 21 (2006) 31–52.
[10] S. Lafon, Y. Keller, R.R. Coifman, Data fusion and multicue data matching by diffusion maps, IEEE Trans. Pattern Anal. Mach. Intell. 28 (11) (Nov. 2006) 1784–1797.
[11] N. Rabin, R.R. Coifman, Heterogeneous datasets representation and learning using diffusion maps and Laplacian pyramids, in: Proc. 12th SIAM International Conference on Data Mining, 2012.
[12] A. Fernández-Pascual, N. Rabin, J.R. Dorronsoro, Auto-adaptative Laplacian pyramids for high-dimensional data analysis, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2016, April 2016.
[13] S. Mousazadeh, I. Cohen, Out-of-sample extension of band-limited functions on homogeneous manifolds using diffusion maps, Signal Process. 108 (2015) 521–529.
[14] Y. Aizenbud, A. Bermanis, A. Averbuch, PCA-based out-of-sample extension for dimensionality reduction, eprint arXiv:1511.00831, 2015.
[15] J. He, L. Zhang, Q. Wang, Z. Li, Using diffusion geometric coordinates for hyperspectral imagery representation, IEEE Geosci. Remote Sens. Lett. 6 (4) (Oct. 2009) 767–771.
[16] Z. Farbman, R. Fattal, D. Lischinski, Diffusion maps for edge-aware image editing, ACM Trans. Graph. 29 (6) (Dec. 2010) 145:1–145:10.
[17] G. Hinton, S. Osindero, Y. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (July 2006) 1527–1554.
[18] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[19] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (Jan. 2009) 1–127 [Online]. Available http://dx.doi.org/10.1561/2200000006.
[20] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (Aug. 2013) 1798–1828.
[21] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, ICML-08, ACM, 2008, pp. 1096–1103.
[22] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on Machine Learning, ICML-11, 2011, pp. 833–840.
[23] K. Jia, L. Sun, S. Gao, Z. Song, B.E. Shi, Laplacian auto-encoders: an explicit learning of nonlinear data manifold, Neurocomputing 160 (2015) 250–260.
[24] J.T.-Y. Kwok, I.W.-H. Tsang, The pre-image problem in kernel methods, IEEE Trans. Neural Netw. 15 (6) (2004) 1517–1525.
[25] P. Arias, G. Randall, G. Sapiro, Connecting the out-of-sample and pre-image problems in kernel methods, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR'07, IEEE, 2007, pp. 1–8.
[26] A. Singer, Y. Shkolnisky, B. Nadler, Diffusion interpretation of nonlocal neighborhood filters for signal denoising, SIAM J. Imaging Sci. 2 (1) (Jan. 2009) 118–139.
[27] R. Talmon, I. Cohen, S. Gannot, Single-channel transient interference suppression with diffusion maps, IEEE/ACM Trans. Audio Speech Lang. Process. 21 (1) (Apr. 2012) 130–142.
[28] G. David, A. Averbuch, Hierarchical data organization, clustering and denoising via localized diffusion folders, Appl. Comput. Harmon. Anal. 33 (1) (2012) 1–23.
[29] S. Gepshtein, Y. Keller, Image completion by diffusion maps and spectral relaxation, IEEE Trans. Image Process. 22 (8) (2013) 2839–2846.
[30] G. Mishne, I. Cohen, Multiscale anomaly detection using diffusion maps, IEEE J. Sel. Top. Signal Process. 7 (Feb. 2013) 111–123.
[31] A. Haddad, D. Kushnir, R.R. Coifman, Texture separation via a reference set, Appl. Comput. Harmon. Anal. 36 (2) (Mar. 2014) 335–347.
[32] R.R. Coifman, M.J. Hirn, Diffusion maps for changing data, Appl. Comput. Harmon. Anal. 36 (1) (2014) 79–107.
[33] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, NIPS 17 (2005) 1601–1608.
[34] A. Singer, R.R. Coifman, Non-linear independent component analysis with diffusion maps, Appl. Comput. Harmon. Anal. 25 (2) (2008) 226–239.
[35] R. Talmon, I. Cohen, S. Gannot, R.R. Coifman, Supervised graph-based processing for sequential transient interference suppression, IEEE/ACM Trans. Audio Speech Lang. Process. 20 (9) (2012) 2528–2538.
[36] R. Talmon, R.R. Coifman, Empirical intrinsic geometry for nonlinear modeling and time series filtering, Proc. Natl. Acad. Sci. 110 (31) (2013) 12 535–12 540.
[37] G. Mishne, R. Talmon, I. Cohen, Graph-based supervised automatic target detection, IEEE Trans. Geosci. Remote Sens. 53 (5) (May 2015) 2738–2754.
[38] R. Rojas, Neural Networks: A Systematic Introduction, Springer Science & Business Media, 1996.
[39] Y. Weiss, Segmentation using eigenvectors: a unifying view, in: The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 2, IEEE, 1999, pp. 975–982.
[40] B. Zhao, L. Fei-Fei, E. Xing, Online detection of unusual events in videos via dynamic sparse coding, in: Proc. Computer Vision and Pattern Recognition, CVPR, 2011, IEEE, 2011, pp. 3313–3320.
[41] Q.V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, A.Y. Ng, On optimization methods for deep learning, in: Proceedings of the 28th International Conference on Machine Learning, ICML-11, 2011, pp. 265–272.
[42] S.J. Wright, J. Nocedal, Numerical Optimization, Vol. 2, Springer, New York, 1999.

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007

[43] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.
[44] P.W. Jones, A. Osipov, V. Rokhlin, Randomized approximate nearest neighbors algorithm, Proc. Natl. Acad. Sci. (PNAS) 108 (38) (Sep. 2011) 15 679–15 686.
[45] K. Kawaguchi, Deep learning without poor local minima, in: D.D. Lee, M. Sugiyama, U.V. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems 29, Curran Associates, Inc., 2016, pp. 586–594.
[46] Y. LeCun, C. Cortes, C.J. Burges, The Mnist Database of Handwritten Digits, 1998.
[47] A. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Trans. Inform. Theory 39 (3) (May 1993) 930–945.

Please cite this article in press as: G. Mishne et al., Diffusion nets, Appl. Comput. Harmon. Anal. (2017), http://dx.doi.org/10.1016/j.acha.2017.08.007