7. Supplementary Materials

7.1. BASiS Decoder

In Sec. 5 we illustrate the extensive uses of spectral embedding and how to use BASiS to learn it. As has been extensively researched, the orthogonal basis defined by the spectral decomposition of the graph-Laplacian can preserve significant features of the data (*e.g.* DM's representation, as shown in Fig. 5) and contribute to the performance of diverse learning problems (for example clustering, as illustrated in Sec. 5.2).

In this section we examine another use of the spectral representation in an Encoder-Decoder system. We train 3 concatenated networks: Encoder, BASiS and Decoder, over 2 digits of MNIST: 0 and 1 (The architectures and training parameters are detailed in the Tab. 2 and Tab. 3). The Encoder and the Decoder are trained to minimize the reconstruction error (that is, the output of the Decoder should be close in Euclidean sense to the input image). BASiS is trained as described in Algorithm 1. In addition, Algorithm 2 is used to deal with changes in the input features to BASiS, cased by the Encoder update. The spectral embedding dimension set to 9. The 3 models were trained for 2500 iterations. During the inference, a test image is passed through the 3 networks.

Next, we extract the spectral embedding (that is, the output of BASiS for the given test image) and examine the response of the Decoder to multiplication of each one of the first 5 dimensions by one of the constants $\{-2,-1.5,-1,-0.5,0,0.5,1,1.5,2\}$. The output of the Decoder for all the mentioned options is shown in Fig. 9. It can be noticed that different features of the resulting image correspond to different dimensions of the spectral embedding. For example, changes in the first dimension affect, as expected based on spectral clustering theory, on the label of the image, where for non-spectral embedding (i.e., standard Encoder-Decoder system) there is no dimension that is directly responsible on this feature. Furthermore, when examining the changes in the output image as a result of changing the constant by which a given dimension is multiplied, it can be noticed that the change is carried out smoothly. This distinction is consistent with the result shown in the toy examples of Fig. 1, where the model smoothly divides the space.

7.2. Alignment Transformation Calculation

In this section we detail about the solution of the least squares problem of finding the alignment transformation. For simplicity, let us assume $K=2$. In this case, the optimal transformation for the optimization problem of Eq. (13), for a single anchor i is the one that sustains

$$\begin{pmatrix}\varphi_i^{a,ref}(1)\\ \varphi_i^{a,ref}(2)\end{pmatrix} = \underbrace{\begin{pmatrix} t_1 & t_2 & t_3\\ t_4 & t_5 & t_6\end{pmatrix}}_{T}\begin{pmatrix}\varphi_i^{a}(1)\\ \varphi_i^{a}(2)\\ 1\end{pmatrix}. \tag{22}$$

Equivalently , one can write

$$\begin{pmatrix}\varphi_i^{a,ref}(1)\\ \varphi_i^{a,ref}(2)\end{pmatrix} =$$

$$\begin{pmatrix}\varphi_i^{a}(1) & \varphi_i^{a}(2) & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & \varphi_i^{a}(1) & \varphi_i^{a}(2) & 1\end{pmatrix}\underbrace{\begin{pmatrix}t_1\\ t_2\\ t_3\\ t_4\\ t_5\\ t_6\end{pmatrix}}_{t},$$

where t is the column representation of T. Note that each anchor provides two constraints. Since, the transformation has 6 degrees of freedom (DOF), at least 3 anchors are needed. However, as mentioned in Sec. 4, it is recommended to use more constraints. Given l anchors the affine transformation can be calculated by solving the following equation system

$$\min_{t}||Mt-h||_2^2, \tag{23}$$

where,

$$M = \begin{pmatrix} . & . & . & . & . & .\\ . & . & . & . & . & .\\ \varphi_i^{a}(1) & \varphi_i^{a}(2) & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & \varphi_i^{a}(1) & \varphi_i^{a}(2) & 1\\ . & . & . & . & . & .\\ . & . & . & . & . & .\end{pmatrix} \in \mathbb{R}^{2l\times 6}$$

$$h = \begin{pmatrix} .\\ .\\ \varphi_i^{a,ref}(1)\\ \varphi_i^{a,ref}(2)\\ .\\ .\end{pmatrix} \in \mathbb{R}^{2l}.$$

Now it is possible to find the vector t, and the transformation T, by a simple least squares problem that takes into account all the constraints. Note that expanding the spectral dimension to $K>2$ is immediate and only affects the sizes of the matrices.

7.3. Technical Details

7.3.1 Data Sets

In Sec. 5.2 we compare between BASiS and competing models over 4 well-known datasets. Examples of which

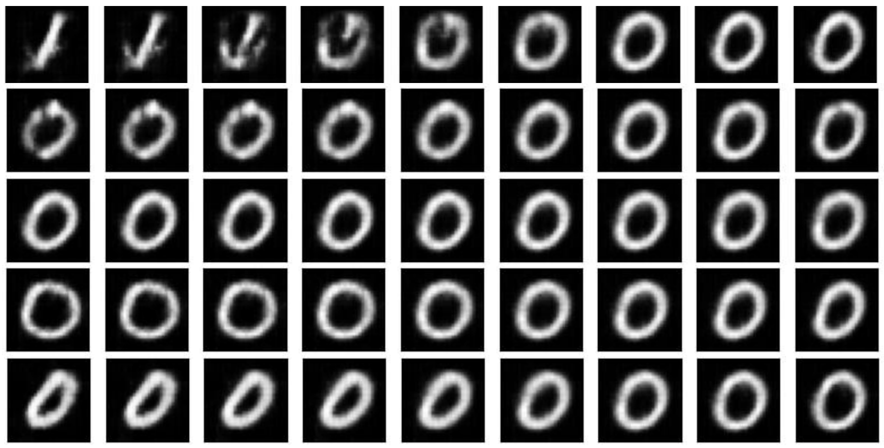

Figure 9. **The Decoder response for changes in the spectral embedding.** The reconstructed images obtained after changes in the first 5 dimensions of the spectral representation of a test image. Each line corresponds to one of the 5 dimensions, each column represents a multiplication of the original value by one of the constants $\{-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2\}$ respectively.

can be found in Fig. 10. In this section we describe those datasets.

MNIST. The Modified National Institute of Standards and Technology database includes 70,000 gray-scale images of handwritten digits, labeled from 0 to 9. The original size of the images is 28×28. The dataset is divided to 60,000 training images and 10,000 test images.

Fashion-MNIST. This dataset includes gray-scale images divided into categories of fashion products (*e.g*. T-shirt, Sandal, Bag etc.). It is identical to the original MNIST in the training and test sets size and in the image dimensions.

SVHN. The Street View House Numbers dataset contains real-world RGB images of dimension $3 \times 32 \times 32$. This dataset includes $73,257$ training images and $26,032$ test images, labeled from 1 to 10.

CIFAR-10. The Canadian Institute For Advanced Research dataset includes RGB images of size $3 \times 32 \times 32$ labeled as animals (*e.g*. cat, dog) and vehicle (*e.g*. airplane, truck). This dataset includes $50,000$ training images and $10,000$ test images.

Siamese Network Representation. Siamese Networks are DNNs trained to find a low-dimensional representation of the data, with Contrastive loss, Eq. (21). In the training process, the data is divided into pairs from the same class (labeled as positive), and pairs from different classes (labeled as negative). In the obtained representation, instances from the same class are expected to be close to each other, in the sense of Euclidean distance, where instances from different classes far from each other. Therefore, this representation is useful when working with the affinity matrix, Eq. (1), which is based on Euclidean distance between the graph nodes.

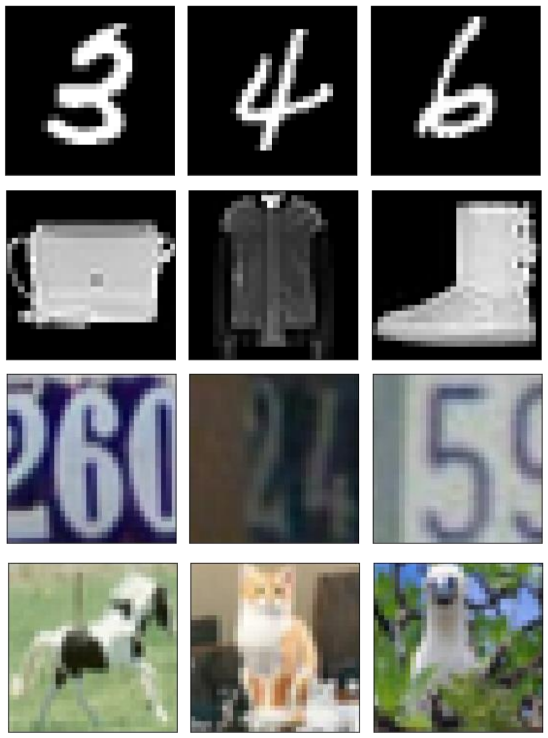

Figure 10. **Datasets Samples**. Images from the different datasets: MNIST (1st row), F-MNIST (2nd row), SVHN (3rd row), CIFAR-10 (4th row).

Spectral Modules	Siamese Net (gray-scale)	Siamese Net (RGB)	Encoder	Decoder
Linear (size=256)	Conv2d (o=20)	Conv2d (o=32)	Conv2d (o=8)	Linear (size=128)
ReLU	MaxPool	ReLU	ReLU	ReLU
Linear (size=512)	Conv2d (o=50)	Conv2d (o=64)	Conv2d (o=16)	Linear (size=288)
ReLU	MaxPool	ReLU	BatchNorm	Unflatten (size=(32,3,3))
Linear (size=512)	Flatten	MaxPool	ReLU	ConvTranspose2d (o=16)
ReLU	Linear (size=512)	Conv2d (o=32)	Conv2d (o=32)	BatchNorm
Linear (size=256)	ReLU	ReLU	Flatten	ReLU
ReLU	Linear (size=16)	Conv2d (o=8)	Linear (size=128)	ConvTranspose2d (o=8)
Linear (size=K)		ReLU	ReLU	BatchNorm
ReLU		MaxPool	Linear (size=16)	ReLU
		Flatten		ConvTranspose2d (o=1)
		Linear (size=256)		Sigmoid
		ReLU		
		Linear (size=16)		
		Sigmoid		

Table 2. **Architectures**. The architectures of the DNNs for the spectral modules (BASiS and the competing methods) and all the other mentioned networks. K is the dimension of the spectral embedding, o stand for the number of output channel in Conv2d and ConvTranspose2d layers.

	Spectral Modules	Siamese Net (gray-scale)	Siamese Net (RGB)	Encoder	Decoder
Batch size	512	128	256	512	512
Learning rate	10^{-4}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
Optimizer	Adam	Adam	Adam	Adam	Adam
Epochs	-	500	1000	-	-

Table 3. **Additional training parameters** The training parameters of the different DNNs. The Spectral Modules, the Encoder and the Decoder are not trained with fixed-epochs but with samples from the entire training set drawn in each iteration.

7.3.2 Networks Implementation

In this section we provide the implementation details of the networks we have worked with. The architectures for all the networks are detailed in Tab. 2. Additional parameters of the training are summarized in Tab. 3.

Spectral Modules. To fairly compare between BASiS and the other spectral models mentioned in Sec. 3, we used the same architecture with the same parameters for all the methods. The architecture is based on a simple fully connected network with ReLU activation between the layers. The output is of dimension K which is the number of eigenvectors the DNN learns. As suggested in [24], in order to obtain a good generalization, the minibatches for the spectral modules are randomly sampled from the entire training set, and not by using fixed epochs. Working with fixed epochs in the context of spectral models means learning the eigenvectors of submatrices of the graph-Laplacian. In order to avoid this and to learn a more generalized model, the spectral models are trained for 1000 iterations, with m instances being sampled from the entire training set, in each iteration.

Siamese Network. The Siamese networks are implemented as CNNs. We use two different architectures, one for gray-scale images (MNIST and Fashion-MNIST) and one for RGB images (SVHN and CIFAR-10). The new lower-dimension, for all the dataset, set to 16. Note that when we refer to batch size in the context of Siamese networks we mean pairs of images (some of which are labeled as positive and some as negative). For the gray-scale model the kernel size of the convolution layer set to $k = 5$, the stride parameter is $s = 1$ and the padding parameter is $p = 0$. For the RGB model the convolution layer parameters are $k = 3, s = 1, p = 1$. The Max-Pooling parameters for both models are $k = 2, s = 2, p = 0$. Note that we use these networks for the spectral clustering experiment of Sec. 5.2 and also as the features model in Sec. 5.4.

Encoder-Decoder. The Encoder and Decoder of Sec. 7.1 are implemented as CNNs. The parameters of the convolution layer of the Encoder and the transpose-convolution of the Decoder are $k=3, s=2, p=0$. Since BASiS model is trained between the Encoder and the Decoder, those models are also trained with iterations and not with pre-defined epochs.

7.3.3 Additional Implementation and Analysis Details

In all experiments, the affinity matrix W is defined by the number of nearest neighbors to each node. In order to maintain symmetry, we update the matrix such that $W \leftarrow (W+W^T)$. The soft-threshold parameter σ set, for MNIST and Fashion-MNIST dataset, by the distance from the 7th nearest neighbor to each node. For SVHN and CIFAR-10 we used fixed value of 1000.

Diffusion Net. The hyper parameters of the loss function, Eq. (7), set to $\mu = 10^{-8}$ and $\eta = 100$. As mentioned in Sec. 5.2, DN's training process is not scalable. Therefore, we randomly sample a batch-sized subset of the training set, and train the network based on this subset analytically calculated spectral embedding. We chose this setting, on the one hand, to fairly compare DN with the other models, which deal with a batch-size subgraph in each iteration, and on the other hand, to illustrate the problematic nature of DN, which is limited in its ability to deal with large and complex datasets. We note that in order to get good performance with this model it is necessary to define the graph using a sufficiently large environment for each node. As shown in Fig. 4, when the environment of each node is small, the performance of the model is highly dependent on the sampled training-batch and therefore a very large standard deviation is obtained.

SpectralNet1. Follow the instruction of the authors, at each iteration we do an orthogonalization step and gradient step. At the orthogonalization step we draw batch and update the last layer of the network by performing QR decomposition, via the Cholesky decomposition, over the inputs to this layer. At the gradient step, we draw a new batch and update all the network weights, except those of the last layer, based on the gradient of the loss function, Eq. (8). Note that in Sec. 5.2 we calculate the spectral embedding of the normalized- Laplacian L_N and therefore we normalize y_i, y_j of Eq. (8) with the corresponding node's degree as mentioned in Sec. 3.

SpectralNet2. In this model, the authors wish to waive the orthogonality step of SpecNet1. From the experiments we performed it seems that the orthogonality of the DNN's output Y is not sufficient. Orthogonality is achieved only after performing the post processing step over Y using the matrix O of Eq. (10). In addition, to get consistent performance at inference, for all the measures we examined, it is necessary to perform the post processing over all the test instances. That is, the solution of Eq. (10) should be performed over Y which includes the DNN's output for the entire test set. Otherwise, an inconsistency is obtained between different batches of the test images. This conclusion greatly limits the model when it is required to work with large test set.

BASiS. In all the experiments, the RANSAC algorithm is used for the calculation of the alignment transformation. The algorithm is preformed for 100 iterations. In each iteration, the transformation is calculated based on 20 anchors, draw randomly from all the given l anchors. For the transformation computed at each iteration, we calculate the reconstruction error based on all the given anchors and count the number of inliers for a tolerance set to 0.1. During the running of the algorithm we keep the iteration where the amount of inliers is the largest. The final transformation is calculated based on those inliers.

We note that the parameters in 5.2 (*e.g.* batch size, number of neighbors per node, number of iterations etc.) were chosen such that all the models could be stably trained. Since BASiS is fully-supervised and based on a simple MSE loss, we achieve good and stable performance also when changing those parameters.