Functional Autoencoders for Dimensionality Reduction in Multivariate Data

Alaa Hasaan Jalob[1] and Lekaa Ali Mohamed[2]
[1]*Department of Accounting, College of Management and Economics, Al-Karkh University of Science, 10001 Baghdad, Iraq*
[2]*Department of Statistics, College of Management and Economics, University of Baghdad, 10071 Baghdad, Iraq*
alaahasaan@kus.edu.iq,lekaa.a@coadec.uobaghdad.edu.iq

Keywords: Functional Data Analysis, Multivariate Functional Data, Functional autoencoder, Representation learning, Dimensionality Reduction.

Abstract: This paper addresses the dimensionality reduction of multivariable function data, where observations are presented as labeled correlated curves on a shared time network, typically sampled on discrete and fuzzy networks. Given the complexity of the interactions between variables and time, we train a small latent space of functions using a functional autoencoder (FAE) to handle novel and unprecedented multivariable smooth trajectories. The experiment assumes a tightly controlled simulation design to test the method under realistic sampling conditions, where the sample size (n), time network resolution (Time), number of Fourier basis functions (φ), and noise level are varied. Several complementary performance metrics are used to determine the quality of dimensionality reduction: the mean integrated error (IMSE), variance conservation ratio (VPR), and Spearman's correlation coefficient (ρ) to determine the overall geometry, and T(k) reliability to determine the local geometry of the reduced space. The results illustrate the trade-off between reconstructing and preserving the structure under different conditions, providing useful information about when functional autoencoders can be relied upon to be useful in multivariable functional analysis, preserving local structure, and reducing dimensions.

1 INTRODUCTION

Multivariable functional data analysis (MFDA) has become an important method of processing large-scale complex data in which data are collected as a function at different positions or locations in a continuum as a vector of correlated functions [1]. In order to gain valuable information on such high-dimensional constructs, the dimensionality reduction (DR) is necessary to maintain the underlying functional structure without deleting noise and slicing errors [2], [3]. Historically, this field has been dominated by classical methods, including functional principal component analysis (FPCA) and the various multivariate versions, as these methods can be interpreted as quasi-linear spaces [4]. MFDA, however, suggests that these linear models tend to fail in the cases where the data have multidimensional, non-linear interdependencies over time and channels [5], [6]. In order to address these constraints, the principle of representation learning has been generalized to functional space with functional autoencoders (FAEs) [7]. It has been demonstrated that FAEs can learn latent representations in a highly efficient way by minimizing the loss of reconstruction of functions by integrating both encoding and decoding functions [8], [9]. Although deep functional analysis models have grown in popularity, there is still a large gap in terms of systematic analysis of the quality of dimensionality reduction that is arguably more than reconstruction errors. The present research tends to concentrate on the Integrated Mean Squared Error (IMSE) and overlook the structural variance and geometric accuracy in the space of the embedded entities [10], [11]. This study will help fill this gap by performing in-depth simulative research that will examine the quality of dimensionality reduction under functional coding analysis. We introduce a multi-dimensional assessment system that does not only take the Integrated Mean Squared Error but also takes into account the structural variance (VPR) and geometric measures like Spearman rank conservation and reliability T(k) of the test, especially in terms of its stability at the new noise levels, accuracy of time-networks and its performance with respect to different functional complexities.

2 RESEARCH METHODOLOGY

We use a Functional Autoencoder (FAE) model to learn low-dimensional modes of multivariate functional measurements, where each measurement is made up of a few correlated curves observed on a shared domain and may be corrupted by measurement noise. The suggested methodology models every curve as a functional and not a finite vector and, therefore, allows the encoder-decoder structure to work with functions and reconstruct smooth paths. In order to represent realistic data-collection conditions, the design supports discretely observed curves by mapping the sampled values into the common functional representation before encoding. The second part of the paper describes the FAE architecture, formulating its functional mapping, and the training process employed in our simulation study.

3 MATERIALS AND METHODS

3.1 Functional Autoencoder (FAE) Algorithm

A Functional Autoencoder (FAE) extends the classical autoencoder to functional data, We have a set of multi-dimensional functional data $X = \{x_1, x_2, \dots, x_n\}$, where each $x_i = [x_{i1}(t), x_{i2}(t), \dots, x_{iP}(t)]$ and each $x_{ij}(t)$ is a function in the functional space $\mathcal{H}$ [10]. The goal is to learn a nonlinear function f that transforms the functional data into a compact representation in a finite-dimensional vector space of dimension d, i.e., $y_i = f(x_i)$.

Because functional observations are often discrete and noisy, modern FAE implementations commonly include a projection/feature step that maps sampled values onto a common functional representation, and a basis-driven decoding step that reconstructs smooth curves evaluable at any (t).

3.1.1 Structure of the (FAE)

The FAE consists of two main components [11]:

- *Encoder:* $\phi: \mathcal{H}^P \to \mathbb{R}^d$ maps the functional data to a finite-dimensional vector space.
- Decoder: $\psi: \mathbb{R}^d \to \mathcal{H}^P$ maps the compact representation back to the functional data space.

3.1.2 Functional Weights Equation

The functional weights $w_{k,j}^{(l)}(t)$ are functions that depend on the variable t (e.g., time or space). They *are typically represented using basis functions (e.g.,* B-spline or Fourier):

$$w_{k,j}^{(l)}(t) = \sum_{m=1}^{M} c_{k,j,m}^{(l)} \phi_m(t). \quad (1)$$

where:

- $c_{k,j,m}^{(l)}$ are the coefficients to be learned;
- $\phi_m(t)$ are the basis functions;
- M is the number of basis functions used.

3.1.3 Encoder Equations

Calculation of the activation of the node k of the first hidden layer is made as follows: The summation of inputs that are dependent on the functional weights on the domain τ_j is computed. These sums are then subjected to the activation function $\sigma(.)$ to get the value of the activities $O_k^{(1)}$:

$$O_k^{(1)} = \sigma\left(\sum_{j=1}^{P} \int_{\tau_j} x_j(t) w_{k,j}^{(1)}(t)\, dt\right). \quad (2)$$

where:

- $O_k^{(1)}$ is the activation of node k in the first hidden layer;
- $\sigma(\cdot)$ is the activation function (e.g., ReLU or tanh);
- $w_{k,j}^{(1)}(t)$ are the functional weights for the first layer;
- τ_j is the domain of the function $x_j(t)$.

3.1.4 Decoder Equations

To compute the reconstruction of the function $x_j(t)$, the sum of the values of $O_k^{(l-1)}$ of the preceding hidden layer multiplied by the final-layer functional weights $w_{j,k}^{(l-1)}(t)$ of the reconstructed function $\hat{x}_j(t)$:

$$\hat{x}_j(t) = \sum_k O_k^{(l-1)} w_{j,k}^{(l-1)}(t). \quad (3)$$

where:

- $\hat{x}_j(t)$ is the reconstructed function $x_j(t)$.
- $O_k^{(l-1)}$ is the activation of node k in the previous layer.
- $w_{j,k}^{(l-1)}(t)$ are the functional weights for the final layer.

3.1.5 Loss Function

We compute the difference between the original data $x_{i,j}(t)$ and the reconstructed data $\hat{x}_{i,j}(t)$ in all the samples and functional dimensions in order to compute the loss function. We square these differences then, and average them over all the samples and functional dimensions, which gives us the final loss function.

$$\mathcal{L}(\Omega) = \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{P}\int_{\tau_j}\left(x_{i,j}(t) - \hat{x}_{i,j}(t)\right)^2 dt, \quad (4)$$

where:

- Ω is the set of all functional weights.
- n is the number of samples.
- P is the number of functional dimensions.

3.1.6 Training Using Functional Gradient

To compute the functional gradient of the loss function with respect to the functional weights, we use functional calculus.

3.1.6.1 Gradient for the Output Layer

The gradient is calculated as a difference between original data and the reconstructed data and multiplied by the node activation factor in the first hidden layer.

$$\frac{\partial\mathcal{L}(\Omega)}{\partial w_{j,k}^{(2)}(t)} = \frac{1}{n}\sum_{i=1}^{n} O_i^{(1)}\,\delta_{i,j}^{(2)}(t). \quad (5)$$

where:

$$\delta_{i,j}^{(2)}(t) = -\left(x_{i,j}(t) - \hat{x}_{i,j}(t)\right). \quad (6)$$

3.1.6.2 Gradient for the Hidden Layer

We take the computed gradient of the output layer and multiply it with the input functional weight $x_j(t)$ followed by the application of the activation function.

$$\frac{\partial\mathcal{L}(\Omega)}{\partial w_{k,j}^{(1)}(t)} = \sum_{i=1}^{n}\delta_k^{(1)}\cdot\sigma'\cdot x_{i,j}(t). \quad (7)$$

where:

$$\delta_k^{(1)} = \sum_{j=1}^{P}\int_{\tau_j}\delta_j^{(2)}(t)w_{j,k}^{(2)}(t)\,dt. \quad (8)$$

and σ' is the derivative of the activation function.

3.1.7 Optimization Using Functional Adam Optimizer

The functional weights are updated using the Functional Adam Optimizer, a generalization of the Adam optimizer for functional data.

3.1.7.1 Calculating Exponential Moving Averages

The input representations and gradients moving averages with the decay parameters β_1 and β_2 are computed exponentially over time.

$$m_{s,j,k}^{(l)}(t) = \beta_1 m_{s-1,j,k}^{(l)}(t) + (1-\beta_1)\frac{\partial\mathcal{L}(\Omega_{s-1})}{\partial w_{s-1,j,k}^{(l)}(t)}. \quad (9)$$

$$v_{s,j,k}^{(l)}(t) = \beta_2 v_{s-1,j,k}^{(l)}(t) + (1-\beta_2)\left(\frac{\partial\mathcal{L}(\Omega_{s-1})}{\partial w_{s-1,j,k}^{(l)}(t)}\right)^2 \quad (10)$$

where:

- β_1 and β_2 are decay rates.
- $m_{s,j,k}^{(l)}(t)$ and $v_{s,j,k}^{(l)}(t)$ are the exponential moving averages of the gradients.

3.1.7.2 Bias Correction

Once the moving averages have been calculated we bias correct the exponential moving average of the representations and the gradients.

$$\hat{m}_{s,j,k}^{(l)}(t) = \frac{m_{s,j,k}^{(l)}(t)}{1-\beta_1^s}. \quad (11)$$

$$\hat{v}_{s,j,k}^{(l)}(t) = \frac{v_{s,j,k}^{(l)}(t)}{1-\beta_2^s}. \quad (12)$$

3.1.7.3 Weight Update

In order to compute the functional gradient, we compute the difference between the input representation in the previous step and the functional gradient in order to update the functional weights in the neural model. The resultant update is multiplied by the learning rate and then the square root of the functional gradient is divided with the learning rate to complete the update. To prevent the division by zero, a constant ϵ is added.

$$w_{s,j,k}^{(l)}(t) = w_{s-1,j,k}^{(l)}(t) - \gamma\cdot\frac{\hat{m}_{s,j,k}^{(l)}(t)}{\sqrt{\hat{v}_{s,j,k}^{(l)}(t)} + \epsilon}. \quad (13)$$

where:

- γ is the learning rate;
- ϵ is a small constant to avoid division by zero.

3.1.8 Training Algorithm

1) Initialization. Initialize the functional weights randomly using basis functions (Fourier) (1):

$$w_{k,j}^{(l)}(t) = \sum_{m=1}^{M} c_{k,j,m}^{(l)} \phi_m(t).$$

where $c_{k,j,m}^{(l)}$ are random coefficients.

2) Training. For each sample x_i in the training set:
   - Forward Pass. Compute the activations $O_k^{(1)}$ and $\hat{x}_j(t)$.
   - Backward Pass. Compute the functional gradients $\frac{\partial \mathcal{L}(\Omega)}{\partial w_{k,j}^{(1)}(t)}$ and $\frac{\partial \mathcal{L}(\Omega)}{\partial w_{j,k}^{(2)}(t)}$.
   - Update the weights using the Functional Adam Optimizer.

3) Iteration. Repeat Step 2 until convergence.

4 SIMULATION OVERVIEW

The controlled Monte Carlo simulation that we performed evaluated dimensionality-reduction behavior of Functional Autoencoders (FAE) on multivariate functional data [12]. In both cases each observation is a vector of observed correlated functional trajectories on a common time domain recorded on a discrete grid. This is achieved by the underlying signals being generated in a Fourier basis expansion with (φ) indicating the number of Fourier basis functions that are employed in order to control the structural richness of the functional process and additive Gaussian noise is added to describe measurement error. The simulation is a full-factorial design with respect to sample size (n ∈ {20,50,100}), time-grid resolution (Time ∈ {90,230}), fourier-basis complexity ($\varphi \in \{7,11,15\}$) and noise level (Noise ∈ {0.05,0.10,0.30}) with 54 experimental conditions. There are three sample sizes, and results are presented in Tables 1-6; in Table 1-3 the sample size is (20,50,100), in Table 4-6 the sample size is (20,50,100); performance is recorded at the possible combinations of (φ) and Noise, in Table 1-3 with (Time=90), as shown in Figure 1, and in Table 4-6 with (Time=230), as shown in Figure 2. This organization can compare the direct dependence of growing sample size and temporal resolution to the learnt FAE representations at the same Fourier complexity and noise regimes.

Table 1: Dimensionality reduction quality metrics for the FAE method at $n = 20$, $Time = 90$, across Fourier basis $\varphi = (7,11,15)$under noise levels $= (0.05,0.10,0.30)$.

φ	Noise	IMSE	dist ρ	T(k)	VPR
7	0.05	7.3E-32	0.996432	0.616667	0.959512
7	0.1	9.47E-32	0.994106	0.316667	0.802751
7	0.3	7.42E-32	0.988743	0.466667	0.292551
11	0.05	3.2E-32	0.993607	0.366667	0.916726
11	0.1	6.19E-32	0.994651	0.383333	0.759609
11	0.3	1.29E-30	0.991396	0.533333	0.297328
15	0.05	4.52E-32	0.995583	0.533333	0.941025
15	0.1	6.9E-32	0.995892	0.4	0.841659
15	0.3	1.13E-30	0.991892	0.383333	0.307214

Table 2: Dimensionality reduction quality metrics for the FAE method at $n = 50$, $Time = 90$, across Fourier basis $\varphi = (7,11,15)$under noise levels $= (0.05,0.10,0.30)$.

φ	Noise	IMSE	dist ρ	T(k)	VPR
7	0.05	0.004749	0.997909	0.224	0.765948
7	0.1	0.006998	0.99756	0.22	0.629704
7	0.3	0.029748	0.99559	0.296	0.193887
11	0.05	0.004333	0.997895	0.28	0.794337
11	0.1	0.007254	0.997014	0.256	0.619779
11	0.3	0.029886	0.996936	0.244	0.191291
15	0.05	0.004477	0.998167	0.28	0.817043
15	0.1	0.006931	0.997117	0.236	0.622546
15	0.3	0.030984	0.996026	0.232	0.174125

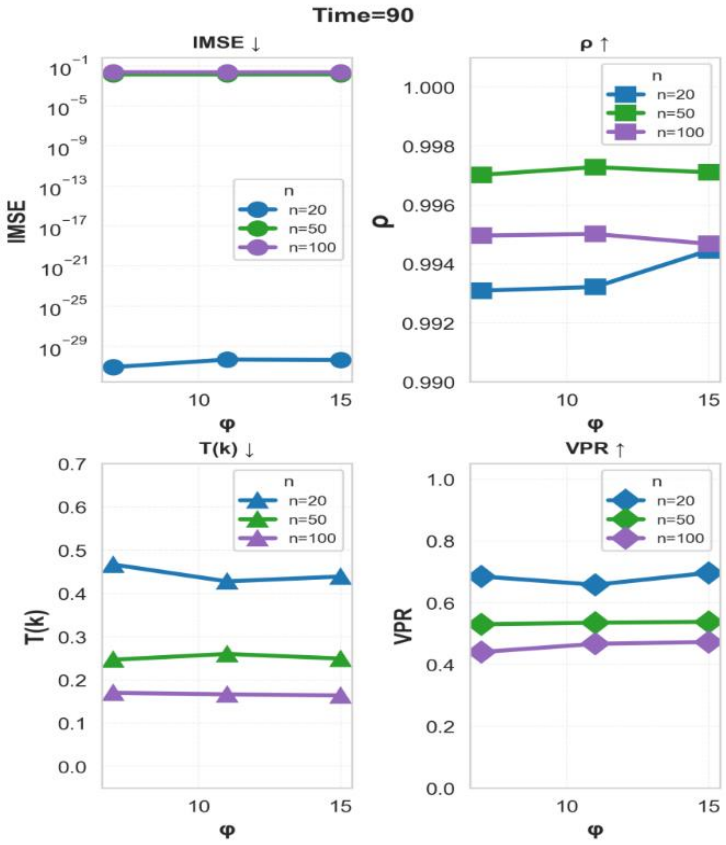

Figure 1: FAE comparison T =90 for all measures.

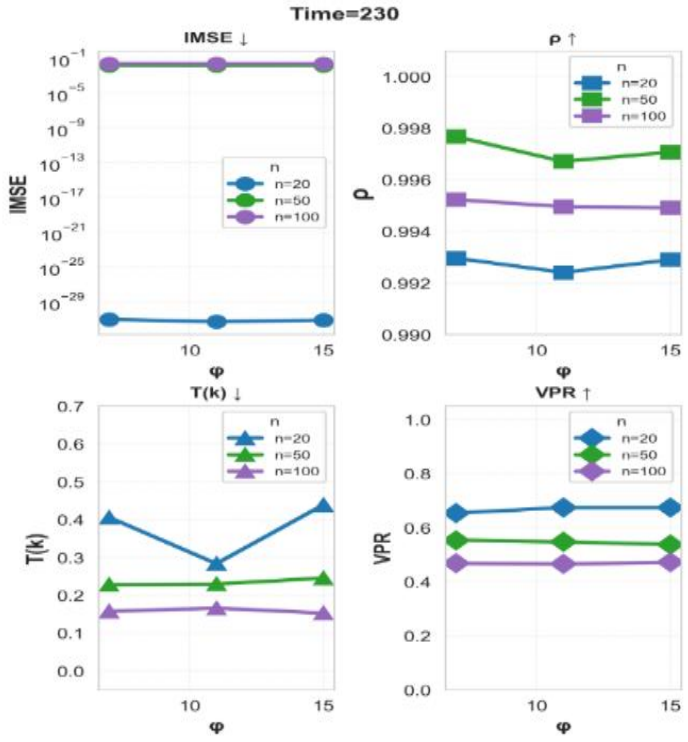

Figure 2: FAE comparison T =230 for all measures.

Table 3: Dimensionality reduction quality metrics for the FAE method at $n = 100$, $Time = 90$, across Fourier basis $\varphi = (7,11,15)$under noise levels $= (0.05,0.10,0.30)$.

φ	Noise	IMSE	dist ρ	T(k)	VPR
7	0.05	0.00718	0.996846	0.18	0.690044
7	0.1	0.011044	0.995621	0.172	0.500164
7	0.3	0.050522	0.992418	0.158	0.1305
11	0.05	0.006691	0.997165	0.176	0.724275
11	0.1	0.010595	0.996089	0.168	0.539548
11	0.3	0.049315	0.99179	0.156	0.136118
15	0.05	0.006627	0.997123	0.176	0.728408
15	0.1	0.01079	0.996055	0.146	0.552803
15	0.3	0.049521	0.990865	0.17	0.137064

Table 4: Dimensionality reduction quality metrics for the FAE method at $n = 20$, $Time = 230$, across Fourier basis $\varphi = (7,11,15)$under noise levels $= (0.05,0.10,0.30)$.

φ	Noise	IMSE	dist ρ	T(k)	VPR
7	0.05	4.09E-32	0.993788	0.533333	0.903156
7	0.1	3.9E-32	0.99437	0.316667	0.763277
7	0.3	1.97E-31	0.990721	0.366667	0.29598
11	0.05	2.68E-32	0.99354	0.3	0.935988
11	0.1	3.95E-32	0.993098	0.216667	0.785938
11	0.3	8.73E-32	0.990585	0.333333	0.30096
15	0.05	6.65E-32	0.99514	0.466667	0.950148
15	0.1	7.92E-32	0.994564	0.35	0.801037
15	0.3	8.61E-32	0.988981	0.5	0.27157

Table 5: Dimensionality reduction quality metrics for the FAE method at $n = 50$, $Time = 230$, across Fourier basis $\varphi = (7,11,15)$ under noise levels $= (0.05,0.10,0.30)$.

φ	Noise	IMSE	dist ρ	T(k)	VPR
7	0.05	0.003938	0.998087	0.236	0.82997
7	0.1	0.00644	0.99804	0.216	0.656935
7	0.3	0.032142	0.996881	0.232	0.173289
11	0.05	0.003882	0.996339	0.24	0.7997
11	0.1	0.00666	0.997507	0.228	0.667529
11	0.3	0.031415	0.996273	0.22	0.174534
15	0.05	0.004231	0.997718	0.272	0.808207
15	0.1	0.006797	0.997329	0.236	0.627974
15	0.3	0.032227	0.996169	0.228	0.177254

Table 6: Dimensionality reduction quality metrics for the FAE method at $n = 100$, $Time = 230$, across Fourier basis $\varphi = (7,11,15)$ under noise levels $= (0.05,0.10,0.30)$.

φ	Noise	IMSE	dist ρ	T(k)	VPR
7	0.05	0.006285	0.996933	0.186	0.73421
7	0.1	0.010051	0.996542	0.142	0.547647
7	0.3	0.051923	0.99222	0.144	0.123238
11	0.05	0.006136	0.99732	0.17	0.713831
11	0.1	0.009893	0.996248	0.168	0.553869
11	0.3	0.051195	0.991295	0.16	0.129688
15	0.05	0.00603	0.997071	0.158	0.744129
15	0.1	0.00998	0.996028	0.154	0.54371
15	0.3	0.051235	0.991648	0.146	0.126977

Resting on the results of the simulation as shown in the six tables above, a brief statistical analysis may be carried out to represent the primary performance trends in the investigated scenarios.

First, the prevalent factor in each table is noise; e.g., in Table 1 the variance preservation ratio VPR decreases sharply as Noise increases from 0.05 to 0.30 (e.g., for(φ=7): from 0.9595 to 0.2926), with a similar decline across(φ=11) and (φ=15), whereas dist ρ remains very high with only a slight reduction (e.g., from 0.9964 to 0.9887). The reason is that an increase in noise decreases the signal-to-noise ratios and thus hampering the capability of the latent representation to represent the actual functional variability and thus collapsing VPR, whereas dist ρ is a rank measure that is robust to perturbations provided the global structure is not heavily compromised.

Second, when comparing the two groups' Tables 1-3 and 4-6 shows that settings producing near-zero IMSE (e.g., Table 1 and Table 4) are generally associated with higher VPR and relatively higher T(k) than settings where IMSE takes clearly positive values (e.g., Tables 2, 3 and Tables 5, 6). One possible statistical reason is that there are configurations where the reconstruction attains an easier regime (because of the properties of the sample/grid, or because of the calculation and normalization of IMSE) with results essentially perfect reconstruction, whereas in other configurations reconstruction becomes more difficult, as IMSE rises to non-scientific-notation values (from 3.21E-32 to 1.29E-30) in later tables, and the variance goes down and the neighborhood is in place, e.g. 0.004268-0.031072.

Third, one can see the impact of the time information through the corresponding table: VPR improves.modestly.in.some..low-noise.cases(e.g., (φ=7), Noise (=0.05): on 0.7659 in Table 2 to 0.8300 in Table 5 and on 0.6900 in Table 3 to 0.7342 in Table 6 and dist ρ is near one. This is not surprising since a higher/ denser temporal sampling gives more information to learn, the numeric approximation error tendency is lesser and the FAE could better represent functional dynamics. Lastly, the functional complexity φ effect exists but is second-order to noise: at the same noise level, VPR and T(k) tend to vary only a little between (7, 11, 15), as due to different φ levels changing the signal shape/complexity, but when noise dominates, signal quality and contamination drive the metrics more than does variance between 15 and 7.

5 CONCLUSIONS

The simulation study yielded three principal conclusions:

1) In multivariate functional data, the performance of functional autoencoders (FAEs) is first of all influenced by the level of noise: the more noise is used in the measurements, the less functional variance will be retained in the latent representation, the less accurate the potential reconstructions can be.
2) Regardless of this sensitivity to variance preservation and reconstruction, the extracted embeddings are very robust in preserving the global structure, and distance-order consistency is high across settings. It implies that FAEs are able to preserve general geometrical correlations despite the decline in local accuracy.
3) Local neighborhoods are not as stable to maintain, particularly when there are small samples, which underscores the fact that the quality of local embeddings is more prone to perturbation and sampling variability than global geometry. In any design that is compatible, a systematic improvement in embedding quality can be achieved by improving temporal information (intensive or higher-quality field sampling) and minor variations in functional complexity are subordinate to noise.

The findings indicate that the Functional Autoencoder (FAE) algorithm is effective in nonlinear relationship processing and dimension reduction preserving local structure. Nonetheless, the fact that the study uses simulated data restricts its use to realistic noise conditions. Future studies need to use FAE on real data and investigate its applicability in functional data classification and regression.

6 RECOMMENDATIONS

The researchers suggest using Functional Autoencoder (FAE) in order to cut the dimensionality of multivariate functional data with a focus on noise reduction and an increase on the density/regularity of the time-domain observation points to guarantee a more confident latent representation. They additionally suggest robustness testing of the findings by repeated use and variation of (k) in the case of proximity measures, careful recording of the error

calculation and normalization mechanism in case of near zero IMSE values to guarantee correct interpretation and reproducibility.

REFERENCES

[1] S. Singh, S. Coyle, and M. Zhang, "Shape-Informed Clustering of Multi-Dimensional Functional Data via Deep Functional Autoencoders," in 39th Conference on Neural Information Processing Systems (NeurIPS 2025), Vancouver, BC, Canada, Dec. 2025, [Online]. Available: https://openreview.net/forum?id=5mpQO0YpTv.

[2] S. Golovkine, E. Gunning, A. J. Simpkin, and N. Bargary, "On the estimation of the number of components in multivariate functional principal component analysis," Communications in Statistics - Simulation and Computation, 2025, [Online]. Available: https://doi.org/10.1080/03610918.2025.2459862.

[3] Y. Zhou et al., "Multimodal functional deep learning for multiomics data," Briefings in Bioinformatics, vol. 25, no. 5, Sep. 2024, [Online]. Available: https://doi.org/10.1093/bib/bbae448.

[4] G. Aneiros and P. Vieu, "Variable selection in infinite-dimensional problems," Statistics & Probability Letters, vol. 94, pp. 12-20, 2014, [Online]. Available: https://doi.org/10.1016/j.spl.2014.06.025.

[5] Y. Li, Y. Qiu, and Y. Xu, "From multivariate to functional data analysis: Fundamentals, recent developments, and emerging areas," Journal of Multivariate Analysis, vol. 188, pp. 1-21, 2022, [Online]. Available: https://doi.org/10.1016/j.jmva.2021.104806.

[6] T.-Y. Hsieh, Y. Sun, S. Wang, and V. Honavar, "Functional Autoencoders for Functional Data Representation Learning," 2021, [Online]. Available: https://epubs.siam.org/terms-privacy.

[7] S. Wu, C. Beaulac, and J. Cao, "Functional autoencoder for smoothing and representation learning," Statistical Computing, vol. 34, p. 203, 2024, [Online]. Available: https://doi.org/10.1007/s11222-024-10501-w.

[8] J. Yao and J. Mueller, "Deep Learning for Functional Data Analysis with Adaptive Basis Layers," 2021, [Online]. Available: https://doi.org/10.48550/arXiv.2106.10414.

[9] G. Aneiros, R. Cao, R. Fraiman, C. Genest, and P. Vieu, "Recent advances in functional data analysis and high-dimensional statistics," Journal of Multivariate Analysis, vol. 170, pp. 3-9, 2019, [Online]. Available: https://doi.org/10.1016/j.jmva.2018.11.007.

[10] F. Ieva and A. M. Paganoni, "Depth measures for multivariate functional data," in Communications in Statistics - Theory and Methods, 2013, pp. 1265-1276, [Online]. Available: https://doi.org/10.1080/03610926.2012.746368.

[11] P. Laforgue and S. Clémençon, "Autoencoding any Data through Kernel Autoencoders," 2019.

[12] P. Bickel, P. Diggle, S. Fienberg, U. Gather, I. Olkin, and S. Zeger, Springer Series in Statistics, doi: 10.1007/978-3-642-20192-9.