GrAE ScaLE: Grassmannian AutoEncoder for Scalable Linearly-invariant Embeddings

Sarah Scullen SARAH.SCULLEN@PNNL.GOV
Pacific Northwest National Laboratory
Tim Marrinan TIMOTHY.MARRINAN@PNNL.GOV
Pacific Northwest National Laboratory
Audun Myers AUDUN.MYERS@PNNL.GOV
Pacific Northwest National Laboratory
Tegan Emerson TEGAN.EMERSON@PNNL.GOV
Pacific Northwest National Laboratory
University of Texas at El Paso

Abstract

Supervised learning and generation on subspace-valued data remain difficult in practice, since existing techniques for embeddings of points on the Grassmann manifold can be costly and difficult to scale. In this work, we introduce the Grassmannian Autoencoder (GrAE), an autoencoder architecture designed for subspace-valued data, together with a variational extension, the Grassmannian Variational Autoencoder (GrVAE). These architectures produce informative, robust, low-dimensional latent representations with improved computational scalability. We demonstrate our approaches across three novel subspace datasets designed to help disambiguate failure modes and support benchmarking for future subspace-learning architectures. Our GrAE architectures' compact latent representations offer an efficient and expressive alternative to supervised learning on the Grassmann manifold and a practical step toward scalable subspace learning.

Keywords: Representation Learning, Grassmannian, Subspace Learning, Variation Modeling, Autoencoder

1. Introduction

Subspaces provide a powerful framework for modeling variation in data. Unlike approaches that represent variation via group actions, subspace representations can flexibly capture multiple dimensions of linear variation. Most of the existing work leveraging subspace representations impose or derive a subspace model on their data by exploiting multiple "views" of the data (e.g., multiple images of a face under different illumination conditions (2), multiple genome sequences (22)). Despite promising results across several domains, subspace representations are limited in their operational adoption based on practicality. Historically, Grassmannian learning approaches are encumbered by scalability challenges (e.g., computational, time, or cost) and problems with efficiently capturing or modeling data variation for new samples during deployment. Scalability is the central driver of the presented work and we leave the modeling of variation for new samples to future work.

This work begins addressing scalability challenges through unsupervised deep learning. We develop and present a new architecture that affords a pretraining schema for Grassmannian learning, supports improved efficiency and expressivity of learned subspace representations, and mitigates limitations of prior techniques. Here we introduce an autoencoder

© 2026 S. Scullen, T. Marrinan, A. Myers & T. Emerson.

architecture designed specifically for subspace data. We assess the benefits of our approach by evaluating it against three new subspace datasets for benchmarking. Our central claim is that our Grassmannian autoencoder produces more informative, robust, low-dimensional latent representations of subspace data than existing approaches with lower computational requirements to achieve the desired expressivity for downstream tasks.

2. Related Work

Humans can perform robust identification and detection tasks under different sources of variation. For computer systems that rely on pixelwise comparisons, however, such changes can be difficult to overcome. Before equivariant deep learning made its debut, (30) proposed representations and metrics that are invariant to natural forms of variation. Building on this work, (2) showed that by estimating a low-dimensional subspace from a collection of images of a single subject under different lighting conditions, one could create illumination-invariant facial recognition systems. This construction could be considered an algorithmic ancestor to current invariant/equivariant learning approaches.

Equivariant Neural Networks: Equivariant neural networks are designed so that when a transformation is applied to the input, the output transforms in a predictable and consistent way. Invariant architectures, however, are characterized by encodings or outputs being unchanged under a transformation. For example, convolutional neural networks (CNNs) are equivariant to translations; if the input image is shifted, the feature maps shift by the same amount (20). When paired with pooling layers and decision heads, they can be made invariant (21). Additional architectures, initially used for computer vision tasks, exist for transformations including rotations (6; 36), reflections (10), and scale changes (35). If the types of variation represent group actions that can be defined explicitly or implicitly, group-equivariant networks can often be applied (6; 18; 11). A limitation of these architectures is that they require the group of transformations be known in advance and that the network architecture be designed to be equivariant to that group (7). Encoding the variance directly via the architecture can be difficult when the transformations are complex or not well-understood and have been shown to be computationally expensive, particularly for large groups or high-dimensional data (11).

Learning on Grassmann Manifolds: Subspace learning takes a different approach by mapping observations to points on a Grassmann manifold, where the geometry of the manifold naturally encodes the invariances of the data; the invariance is captured directly by the *input* to the model. The Grassmann manifold $\mathrm{Gr}(k, n)$ characterizes the set of all k-dimensional subspaces of an n-dimensional vector space. The approach of modeling variation as linear subspaces has been shown to be effective for a variety of tasks, including visual classification (15), action recognition (32; 23), hyperspectral images analysis (24), and genome analysis (22) where the data is known to lie on or near a Grassmann manifold. In this work, we restrict our attention to subspaces of real-valued Euclidean space. If $u \in \mathrm{Gr}(k, n)$ is an element of the Grassmann manifold and $U \in \mathbb{R}^{n \times k}$ is an orthonormal basis such that $U^\top U = I$ and $\mathrm{span}\{U\} = u$, then UQ is also an orthonormal basis for u where $Q \in O(k)$ is any $k \times k$ orthogonal matrix. Generalizing this idea, if an observation of an object or phenomenon is encoded by a $n \times k$ feature matrix, U, there might be other $n \times k$ feature matrices that are also valid observations of the thing, but are distinct from U.

Standard methods for clustering, classifying, or comparing subspaces often fail due to the non-Euclidean nature of Grassmannians (1), which has led to the development of specialized approaches that respect the Riemannian geometry. Many techniques exist that translate traditional machine learning tools to the Grassmann manifold, including clustering (12; 16; 23), linear discriminant analysis (13), and kernel-based classification methods (29; 14; 5). Recently, Delogne and Jacques proposed a collection of sketching-based approximations for Grassmannian kernel methods that significantly reduced the computational cost and storage needed to leverage such tools (9). Kernel support vector machines (kSVMs) offer one way to "nonlinearize" support vector machines that have been adapted to subspace data, which can improve classification accuracy when raw data is not separable in the input space (28).

Grassmannian Neural Networks: Beyond traditional machine learning, one of the first architectures to bring together Grassmannian learning and neural networks was the so-called GrNet (15), which has become the backbone of methods like GrasNet (33), GAE-LLE (4), and GDLNet (34). The subsequent works built on GrNet by leveraging the underlying geometry in optimization and by extending a self-attention mechanism for subspace learning. More recently, gyrovectro spaces have been utilized as a new learning paradigm for these same types of networks (25). GrNet is a supervised learning architecture characterized by a repeating sequence of layers. Two of the layer-types are of particular note: 1) `FrMap` layers which multiply the input by a collection of matrices with learned weights and 2) `ReOrth` layers in which an eigenvector decomposition is computed for each output of the `FrMap` layer to construct an orthonormal matrix representative for the latent representation.

The parameters that govern these layers effect changes to the ambient-dimension (n) and subspace-dimension (k) of the intermediate latent representations, respectively. If one chooses dimensions less than those of the full input data, the algorithm is not well-defined with respect to the Grassmannian manifold. For this reason, the first two layers of the GrNet architecture form a bottleneck that directly impacts the invariance of the resulting models. That is, if $U \in \mathbb{R}^{n\times k}$ is an orthonormal basis for a point on $\mathrm{Gr}(k,n)$, $Q \in O(k)$, and UQ is another representative for the same point, GrNet applied to U is not necessarily equal to the same GrNet applied to UQ. Due to the construction of these layers, preserving the dimensions of the input data for all intermediate representations is necessary to maintain invariant representations, but the expressivity of the model is thus enabled by a lift to a higher dimension at the expense of scalability.

Grassmannian Autoencoders: Tools like Grassmannian kSVM (29; 5) represent the geometry induced by the chosen kernel. This means that the latent representation is not learned in a strictly data-driven manner and the choice of kernel influences the resulting classification. So while sketching-based approaches can help overcome some of the storage bottlenecks of kernel methods (9), they still approximate the chosen kernel geometry. On the other hand, Grassmannian neural networks like GrNet often learn a latent representation specifically to maximize classification accuracy (15). This approach can produce highly performant models, but latent representations may collapse to the class means to achieve a solution that is optimal with respect to accuracy, losing all within-class variability.

In order to capture the relative geometry of the inputs, while producing a latent representation that is discriminative enough for classification, a Grassmannian autoencoder (GAE-LLE) was recently proposed (4). GAE-LLE uses a paired architecture to learn a lower-dimensional Grassmannian representation, conditioned on the geometry of the input.

For data in $\mathrm{Gr}(k_0, n_0)$, one branch of GAE-LLE employs GrNet layers to embed the subspaces in $\mathrm{Gr}(k_1, n_1)$, where $k_1 \leq k_0$ and $n_1 \leq n_0$, while a second branch uses the pairwise Grassmannian distances of the input data to enforce consistent local neighborhoods in the spirit of the locally linear embeddings (LLE) (27). Consequently, GAE-LLE does not alleviate the computational bottlenecks of GrNet nor the storage requirements of kernel methods.

3. Novel Architectures and Datasets

The Grassmannian autoencoder (GrAE) and Grassmannian variational autoencoder (GrVAE) proposed in this work mitigate the issues of existing methods by computing basis-invariant features that are used as input to the subsequent network, and learning a lightweight map between these features and a Euclidean latent space, rather than mapping input subspaces to a lower-dimensional Grassmannian. Latent vectors are then passed through decoder that produces an unconstrained matrix, which is orthonormalized to enforce feasibility for the reconstructed Grassmannian samples. The network is trained without supervision, which implicitly avoids neural collapse, and the Grassmannian distance used in the reconstruction loss ensures that latent vectors remain expressive. Our work targets practical challenges seen in existing approaches by providing scalable methods to learn an adaptable, invariant, low-dimensional latent representation for subspace data that preserves underlying manifold structure, employs a pretraining loss function consistent with the manifold structure, and extracts meaningful representations for downstream tasks.

3.1. Grassmannian Autoencoder (GrAE)

We assume each dataset sample, $u \in \mathrm{Gr}(k, n)$, is a k-dimensional subspace of $\mathbb{R}^n$, represented by an orthonormal basis $U \in \mathbb{R}^{n\times k}$ such that $\mathrm{span}\{U\} = u$ and $U^\top U = I_k$. The orthonormal basis chosen as a representative for u is not unique, so the proposed GrAE is defined to ensure downstream subspace-learning tasks remain invariant under changes of basis. To remove the basis ambiguity, $U \sim UQ$ for any $Q \in O(k)$, the encoder is defined using the orthogonal projector, $P = UU^\top$, as input, which is invariant to the choice of basis for u.

The upper-triangular entries of P are vectorized, denoted by $\mathrm{vec}_\triangle(P)$, and passed through a multilayer perceptron (MLP) with two hidden layers and Gaussian Error Linear Unit (GELU) activations to obtain a latent code,

$$z = f_{\mathrm{enc}}(\mathrm{vec}_\triangle(UU^\top)) \in \mathbb{R}^d\,.$$

The decoder maps z to a matrix $V \in \mathbb{R}^{n\times k}$ using a second MLP, and then projects V back to a valid Grassmann point using a QR factorization,

$$V = f_{\mathrm{dec}}(z), \qquad QR = V, \qquad \hat{U} = \mathrm{span}\{Q\}\,.$$

This guarantees that $\hat{U}$ has orthonormal columns and therefore represents a valid element of $\mathrm{Gr}(k, n)$. The model is trained end-to-end using a reconstruction objective based on squared chordal distance,

$$\mathcal{L}_{\mathrm{rec}}(U, \hat{U}) = d^2_{\mathrm{ch}}(U, \hat{U}) = k - \|U^\top \hat{U}\|_F^2\,,$$

averaged over the training set. Parameters are optimized with Adam, and validation performance is monitored using the same reconstruction metric.

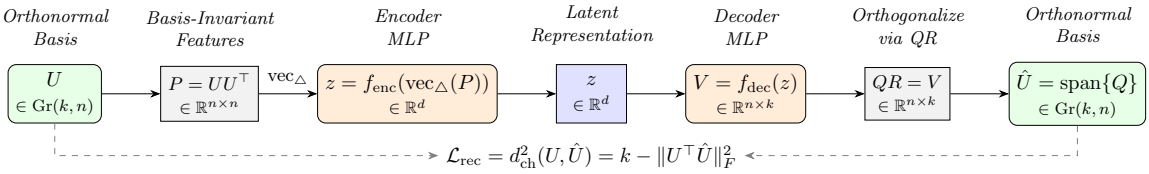


Figure 1: Diagram of the GrAE architecture.

We also present GrVAE: a Grassmannian Variational Autoencoder. GrVAE shares the same architecture as GrAE but modifies the loss function to balance reconstruction error with a Gaussian KL-divergence term,

$$\mathcal{L}(\phi,\theta) \;=\; \mathbb{E}_{q_\phi(z|U)}\big[d^2_{\mathrm{ch}}\big(U,\, f_{\mathrm{dec}}(z)\big)\big] \;+\; \beta\,\mathrm{KL}(q_\phi(z \mid U)\,\|\,\mathcal{N}(0, I_d))\,,$$

with $\beta > 0$ controlling the regularization strength. Further details about this architecture are provided in Appendix A.

3.2. Parameterized Subspace Dataset Construction

When subspace modeling fails to yield benefit for a particular application, the cause can be unclear; is it because there is no actual underlying linear variation that naturally affords a subspace representation or is the approach to learning from the representation limited in some way? Therefore, to more equitably compare architectures intended to leverage subspace structure in learning, we present three new benchmarking datasets. The first two datasets, unlike those typically seen in existing applications of Grassmannians, are native subspaces that can be produced by construction, eliminating the possibility that there is no underlying subspace. The third dataset we consider is more consistent with prior published applications where the subspace representation is inherited from multiple views by extracting an orthonormal basis that spans or captures some fixed percentage of variation for each collection of views. However, the parameterization of the third dataset provides opportunities to controllably stress test subspace learning techniques not afforded by domain-datasets.

3.2.1. Sampled subspace dataset

To produce a controlled training set we start from M ground-truth subspaces $\{U_i\}_{i=1}^M$ and generate J perturbed versions of each. For $j = 1, \ldots, J$ we randomly sample a full-rank mixing matrix, $A_j \in \mathbb{R}^{k\times k}$, and a noise matrix, $N_j \in \mathbb{R}^{n\times k}$, whose entries are drawn i.i.d. from a zero-mean Gaussian with a known, fixed variance. For each pair of indices, (i, j), we form a matrix $V_i^{(j)} \;=\; U_i\,A_j + N_j \;\in\; \mathbb{R}^{n\times k}$ according to a linear mixing model, and orthonormalize the result via the thin QR decomposition as $Q_i^{(j)} R_i^{(j)} = V_i^{(j)}$. The factors $R_i^{(j)}$ are discarded and jth perturbed sample of subspace i can then be written as

$$U_i^{(j)} = \mathrm{span}\{Q_i^{(j)}\} \in \mathbb{R}^{n\times k} \text{ such that } \left(U_i^{(j)}\right)^\top U_i^{(j)} = I_k,$$

creating the data set

$$\mathcal{D} \;=\; \left\{\, U_i^{(j)} \;:\; i = 1, \ldots, M,\; j = 1, \ldots, J \,\right\}.$$

Intuitively, A_j produces a (subspace-dependent) within-class change of basis and N_j injects shared off-subspace noise, so the perturbed samples cluster around their ground-truth subspace on $\mathrm{Gr}(k, n)$.

3.2.2. Cyclic Subspace Dataset

We also consider a synthetic dataset of linear subspaces with direct control over both the intrinsic subspace dimension and the ambient dimension. Specifically, each sample is a k-dimensional subspace of $\mathbb{R}^n$, where $k \leq n$. A sample is represented by an $n \times k$ matrix whose columns span the subspace. Given a nilpotent matrix $A \in \mathbb{R}^{n \times n}$ and a base vector $b \in \mathbb{R}^n$, we define the associated cycle $C(A, b) = \{A^{k-1}b, A^{k-2}b, \ldots, Ab, b\}$. The dataset sample is the k-dimensional subspace

$$\mathcal{S}(A, b) = \mathrm{span}\{A^{k-1}b, A^{k-2}b, \ldots, Ab, b\} \in \mathrm{Gr}(k, n).$$

This construction provides explicit control over the ambient dimension n and the subspace dimension k. We consider two possible classification tasks based on this construction: 1) Each class corresponds to a distinct nilpotent matrix and 2) Each class corresponds to a distinct base vector. More details on the construction can be found in Appendix B.

3.2.3. Trajectory subspace dataset

We use a synthetic video dataset designed to distinguish between linear and nonlinear motion trajectories. Each sample consists of a sequence of grayscale images containing a single moving two-dimensional Gaussian. The dataset contains two trajectory classes: linear trajectories and cubic trajectories and the classification task is to determine the trajectory type from the resulting image sequence.

Several dataset parameters are adjustable, including the image size, Gaussian variance, number of frames per video, and number of sampled start–end pairs. Although the intrinsic dimension of the resulting data manifold is not directly controlled by the parameters (e.g., the data is not a native subspace), taking the singular value decomposition (SVD) of all frames enables one to produce a $k-$dimensional subspace for each video by retaining the top$-k$ singular vectors. Details on the dataset construction can be found in Appendix C.

4. Experimental Design and Results

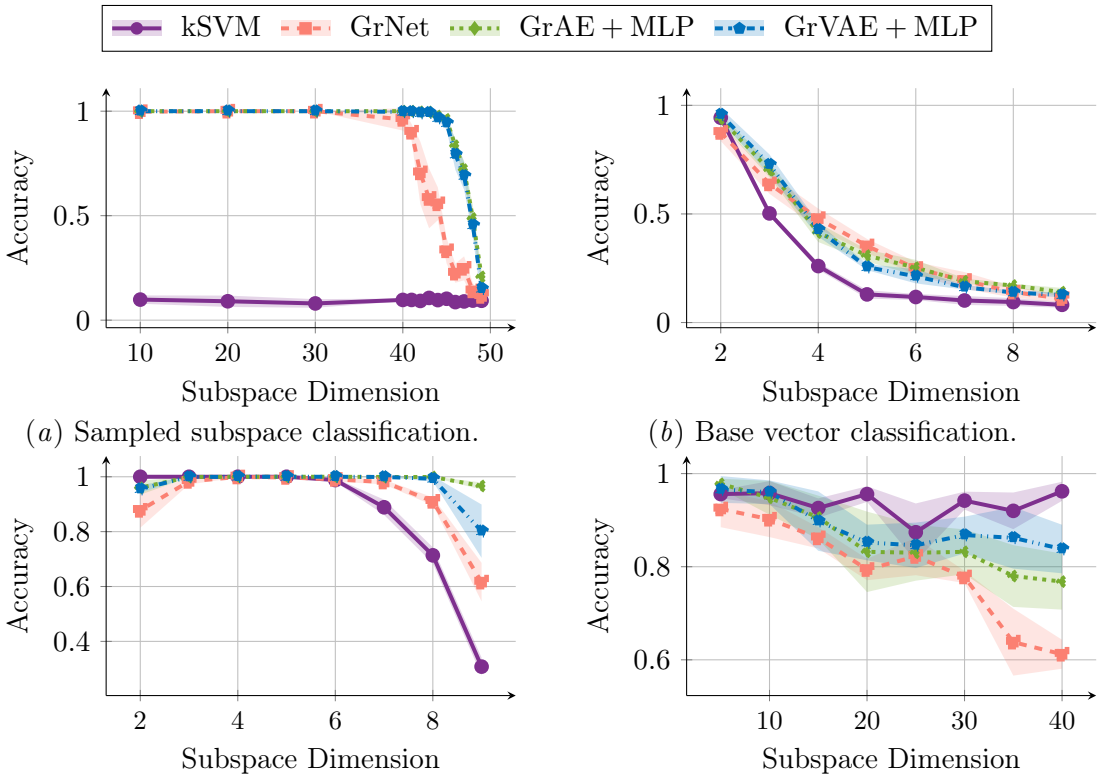
To assess the practicality of subspace learning, we present a collection of experiments that evaluate and compare subspace learning approaches across the three datasets introduced above. Specifically, the experiments are structured to probe the following attributes of subspace learning techniques: 1) Classification task performance across the presented datasets, 2) Computational costs for training and inference, 3) Qualitative variations across techniques, and 4) Performance stability under dimension changes.

In the sampled dataset setting, ground-truth subspaces are perturbed by random linear transformations and additive noise, then re-orthonormalized. For the cyclic and trajectory datasets, precomputed subspace bases are loaded and orthonormalized, ensuring all inputs share a standardized Grassmannian representation.

4.1. Experimental Setup

We conduct the following experiments to quantitatively and qualitatively compare GrAE, GrVAE, GrNet, and kernel-SVM (kSVM) (8) across the presented datasets. All models are trained using the same train/test splits. The GrAE and GrVAE architectures are pretrained in a fully unsupervised manner. After pretraining, an MLP classification layer is applied to the learned latent representation and finetuned for classification. Pairwise chordal-distances between subspaces are used to train the kSVM. Both kSVM and GrNet training are supervised, leveraging class labels. Number of samples, classes, generation parameters, and other choices needed for reproduction of the results in Section 4.2 are provided in Appendix D. We report results averaged over 5 independent runs using 50/50 train/test splits. Computational resources used for all experiments are described in Appendix E.

4.2. Results



(*a*) Sampled subspace classification.

(*b*) Base vector classification.

(*c*) Nilpotent matrix classification.

(*d*) Trajectory video classification.

Figure 2: Mean classification accuracy as a function of subspace dimension for 2(*a*) the sampled subspace dataset, 2(*b*) cyclic subspace: base vector classification, 2(*c*) cyclic subspace: nilpotent matrix classifier, and 2(*d*) the trajectory dataset. The ambient dimensions used for each of the datasets are 50 (sampled), 10 (cyclic), and 400 (trajectory). Shaded regions indicate ±1 standard deviation about the mean from 5 independent trials.

We compare GrAE, GrVAE, GrNet, and a kSVM (see Appendix D.3) across four tasks: sampled subspace classification, cyclic subspace base-vector classification, cyclic subspace

nilpotent-matrix classification, and trajectory video classification. Figure 2 reports classification accuracy as a function of subspace dimension for each task and model. GrAE + MLP and GrVAE + MLP perform strongly across a broad range of subspace dimensions, exceeding the supervised baselines (GrNet, kSVM) in several regimes. In contrast, kSVM, performs worst on the lower-ambient-dimension tasks but very well on the highest-ambient-dimension task (trajectory). This is the only dataset where the number of points is less than the ambient dimension, suggesting that point density affects kSVM more strongly since its kernel is grounded in pairwise chordal distances. All of the architectures experience a performance degradation as the subspace dimension approaches the ambient dimension of the space, as expected. When $k = n$, each subspace spans the full ambient space, rendering classification of the subspaces infeasible. Notably, the autoencoder-based representations decline more slowly, suggesting that reconstruction-driven training yields discriminative latent structure that transfers effectively to downstream tasks.

Table 1: Computational scalability for classifying synthetic subspace data as a function of ambient dimension. The subspace dimension is fixed at 5 and the noise variance is $\sigma_n^2 = 2.5$.

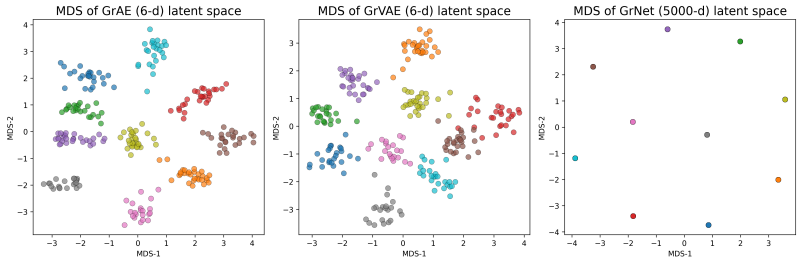
	Ambient Dim.	**25**	**50**	**75**	**100**	**250**	**500**
Training Time (seconds)	kSVM	0.033±0.001	0.038±0.002	0.042±0.002	0.045±0.002	0.071±0.001	0.113±0.002
	GrNet	8.896±0.026	60.172±0.149	124.332±0.169	172.408±0.328	480.81±0.512	1,395.01±1.674
	GrNet-reduced	8.264±0.004	8.863±0.015	52.411±0.078	60.013±0.211	185.408±0.315	480.766±0.893
	GrAE + MLP	33.324±0.023	33.282±0.08	33.429±0.016	33.372±0.051	34.187±0.095	35.249±0.115
	GrVAE + MLP	48.67±0.079	48.732±0.091	48.824±0.114	48.469±0.376	49.157±0.431	51.356±0.487

To assess tractability as ambient dimension increases—an important practical consideration for Grassmannian methods on real-world data —we measure training/inference time while fixing subspace dimension to 5 and varying ambient dimension from 25 to 500. Table 1 reports that computational cost depends strongly on the ambient dimension and on architecture choices within GrNet (e.g., the `FrMap` setting), motivating the comparison between the standard and reduced-parameter variants. Accuracy curves and inference times for the experiments reported in Table 1 can be found in Figure 5 of Appendix D.8.

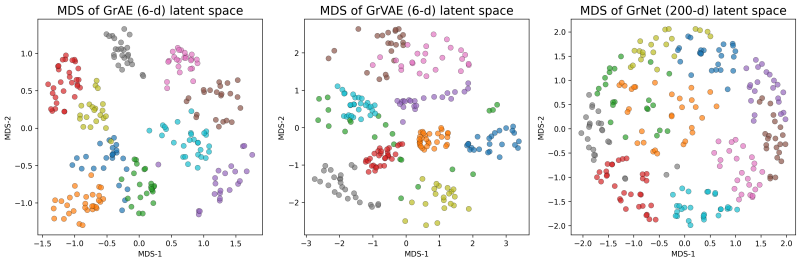
Recall that, by construction, the autoencoder architectures are dimensionally-reductive, mapping inputs from each ambient dimensions down to a fixed 6-dimensional space; this is reflected in the lack of statistically significant training-time growth as a function of ambient dimension. GrNet operates in the opposite direction, lifting inputs to feature spaces with $\geq 10\times$ dimensions for each respective dataset. Thus, the autoencoder architectures simultaneously outperform baselines, perform dimensionality reduction, and scale more efficiently. These results highlight that autoencoder-based approaches can offer favorable scaling behavior while maintaining competitive accuracy, supporting the practicality of GrAE and GrVAE for higher-dimensional settings through dimensional reduction over lifting.

In addition to these quantitative investigations, we also compare the learned latent representations from each method using multi-dimensional scaling (MDS) (19) visualizations to provide qualitative insight into how different training objectives and dimensional scaling shape the embedding geometry (e.g., cluster separation and preserved neighborhood relationships). The structure of the learned clusters in Figure 3 clearly reflects the differences in training objectives. For the sampled subspace dataset example, where each class consists of perturbations around a base subspace, GrNet collapses points to class means, while the

autoencoders (which are trained to reconstruct rather than classify) preserve within-class variability. The reverse pattern emerges for the nilpotent matrix task of the cyclic subspace data: the autoencoders, which learn structure from the data directly rather than class labels, produce tighter clusters while GrNet's supervised objective yields more diffuse groupings. Overall, the clustering that emerges in the latent spaces demonstrate that a low dimensional latent space is sufficient for the autoencoders to maintain both reconstruction fidelity and discriminative structure, achieving clear class separation without the high-dimensional lifting required by GrNet.



(a) Latent space comparison for models trained to classify sampled subspaces in $\mathrm{Gr}(10, 50)$.



(b) Latent space comparison for models trained to classify nilpotent matrices in $\mathrm{Gr}(8, 10)$.

Figure 3: 2D MDS visualizations from pairwise Euclidean distances between of latent representations from (left to right) GrAE, GrVAE, and GrNet, colored by class label, for 3(a) sampled subspace and 3(b) nilpotent matrix classification (cyclic subspace data).

5. Conclusions and Future Work

The Grassmannian Autoencoder architectures presented here show promise as a practical, scalable approach to subspace learning. GrAE and GrVAE learn representations through a reconstruction task, a significant departure from prior supervised learning architectures like GrNet. Across three datasets designed to improve benchmarking for subspace algorithms, this paper demonstrates three key benefits of GrAE variants: 1) superior classification performance, 2) improved computational scalability, and 3) less performance sensitivity to the difference between ambient and subspace dimensions. While our results show a promising path toward computational scalability, subspace generation and completion remain an enduring challenge; particularly when the sources of linear variation cannot be easily modeled or measured (e.g. capturing a new face under all illumination conditions). Building on our Grassmanian variational autoencoder and promising results, future work will focus on subspace generation including new training paradigms and generative techniques to address a remaining barrier to wider adoption.

References

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[2] Peter N Belhumeur and David J Kriegman. What is the set of images of an object under all possible illumination conditions? *International Journal of Computer Vision*, 28:245–260, 1998.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.

[4] Zeming Chen, Benchao Li, Weixiao Dai, Zicheng Hu, and Ruisheng Ran. Grassmann neighborhood preserving autoencoder for image set classification. In *International Conference on Intelligent Computing*, pages 398–409. Springer, 2025.

[5] Sofya Chepushtanova and Michael Kirby. Sparse Grassmannian embeddings for hyperspectral data representation and classification. *IEEE Geoscience and Remote Sensing Letters*, 14(3):434–438, 2017.

[6] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 2990–2999. PMLR, 2016.

[7] Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant CNNs on homogeneous spaces. *Advances in Neural Information Processing Systems*, 32, 2019.

[8] Corinna Cortes and Vladimir Naumovich Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. URL https://api.semanticscholar.org/CorpusID:52874011.

[9] Rémi Delogne and Laurent Jacques. Random features for Grassmannian kernels. In *2025 IEEE Statistical Signal Processing Workshop (SSP)*, pages 221–225. IEEE, 2025.

[10] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *International Conference on Machine Learning*, pages 1889–1898. PMLR, 2016.

[11] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) equivariant representations with spherical CNNs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018.

[12] Peter Gruber and Fabian J Theis. Grassmann clustering. In *2006 14th European Signal Processing Conference*, pages 1–5. IEEE, 2006.

[13] Jihun Hamm and Daniel D Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 376–383, 2008.

[14] Mehrtash T Harandi, Mathieu Salzmann, Sadeep Jayasumana, Richard Hartley, and Hongdong Li. Expanding the family of Grassmannian kernels: An embedding perspective. In *European Conference on Computer Vision*, pages 408–423. Springer, 2014.

[15] Zhiwu Huang, Jiqing Wu, and Luc Van Gool. Building deep networks on Grassmann manifolds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[16] Suraj Jain and Venu Madhav Govindu. Efficient higher-order clustering on the Grassmann manifold. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3511–3518, 2013.

[17] Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian optimization in PyTorch, 2020.

[18] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pages 2747–2755. PMLR, 2018.

[19] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 991–999, 2015.

[22] Xiaoguang Li, Tao Zhou, Xingdong Feng, Shing-Tung Yau, and Stephen S-T Yau. Exploring geometry of genome space via Grassmann manifolds. *The Innovation*, 5(5), 2024.

[23] Tim Marrinan, J Ross Beveridge, Bruce Draper, Michael Kirby, and Chris Peterson. Finding the subspace mean or median to fit your need. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1082–1089, 2014.

[24] Timothy Marrinan, J Ross Beveridge, Bruce Draper, Michael Kirby, and Chris Peterson. Flag-based detection of weak gas signatures in long-wave infrared hyperspectral image sequences. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XXII*, volume 9840, pages 407–416. SPIE, 2016.

[25] Xuan Son Nguyen, Yang Yang, and Aymeric Histace. Matrix manifold neural networks++. In *International Conference on Learning Representations*, volume 2024, pages 40448–40478, 2024.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[27] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[28] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[29] Ryosuke Shigenaka, Bisser Raytchev, Toru Tamaki, and Kazufumi Kaneda. Face sequence recognition using Grassmann distances and Grassmann kernels. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2012.

[30] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519–524, 1987.

[31] Oleg Smirnov. TensorFlow Riemopt: a library for optimization on Riemannian manifolds. *arXiv preprint arXiv:2105.13921*, 2021.

[32] Pavan Turaga, Ashok Veeraraghavan, Anuj Srivastava, and Rama Chellappa. Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11): 2273–2286, 2011.

[33] Rui Wang and Xiao-Jun Wu. Grasnet: A simple Grassmannian network for image set classification. *Neural Processing Letters*, 52(1):693–711, 2020.

[34] Rui Wang, Chen Hu, Ziheng Chen, Xiao-Jun Wu, and Xiaoning Song. A Grassmannian manifold self-attention network for signal classification. In *IJCAI*, pages 5099–5107, 2024.

[35] Daniel Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 32, 2019.

[36] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

Appendix A. Grassmannian Variational Autoencoder (GrVAE) architecture

Let $\mu_\phi : \mathrm{Gr}(k,n) \to \mathbb{R}^d$ and $\log \sigma^2_\phi : \mathrm{Gr}(k,n) \to \mathbb{R}^d$ denote the encoder mean and log-variance networks, parameterized by ϕ. The approximate posterior is

$$q_\phi(z \mid U) = \mathcal{N}\big(z;\ \mu_\phi(U),\ \mathrm{diag}\big(\sigma^2_\phi(U)\big)\big)\,,$$

with reparameterization $z = \mu_\phi(U) + \sigma_\phi(U) \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I_d)$. The decoder mirrors the one defined for GrAE, where $f_{\mathrm{dec}} : \mathbb{R}^d \to \mathrm{Gr}(k,n)$ is an MLP with GELU activations, parameterized by θ, that outputs an orthonormal representative

$$\widehat{U} = f_{\mathrm{dec}}(z) \in \mathbb{R}^{n\times k}, \qquad \widehat{U}^\top \widehat{U} = I_k,$$

by producing an unconstrained matrix $\widetilde{U} \in \mathbb{R}^{n\times k}$ and orthonormalizing it via the thin QR decomposition. Consistent with the proposed GrAE, the GrVAE uses the chordal distance as the reconstruction loss on $\mathrm{Gr}(k,n)$, which is balanced with the KL-divergence term to yield the GrVAE loss as a function of the parameters of the encoder, ϕ, and decoder, θ,

$$\mathcal{L}(\phi,\theta) \;=\; \mathbb{E}_{q_\phi(z|U)}\big[d^2_{\mathrm{ch}}\big(U,\, f_{\mathrm{dec}}(z)\big)\big] \;+\; \beta\,\mathrm{KL}(q_\phi(z\mid U)\,\|\,\mathcal{N}(0,I_d))$$

for values of $\beta > 0$.

Appendix B. Cyclic Subspace Dataset Generation

Each cyclic subspace is a k-dimensional subspace of $\mathbb{R}^n$, where $k \le n$. A sample is represented by an $n \times k$ matrix whose columns span the subspace. Given a nilpotent matrix $A \in \mathbb{R}^{n\times n}$ and a base vector $b \in \mathbb{R}^n$, we define the associated cycle $C(A,b) = \{A^{k-1}b, A^{k-2}b, \ldots, Ab, b\}$. The dataset sample is the k-dimensional subspace

$$\mathcal{S}(A,b) = \mathrm{span}\{A^{k-1}b, A^{k-2}b, \ldots, Ab, b\} \in \mathrm{Gr}(k,n).$$

We consider two possible classification tasks based on this construction.

B.1. Nilpotent matrix class

In the first setting, each class corresponds to a distinct nilpotent matrix. We randomly generate M nilpotent matrices $A_1, A_2, \ldots, A_M \in \mathbb{R}^{n\times n}$, each with nilpotency index k. This is done using the protocol provided in Appendix B.3. For a sample from class j, we draw a random base vector $b \in \mathbb{R}^n$ ($b \sim \mathcal{N}(0, I_n)$, resampled until the resulting cyclic basis has rank k) and form the subspace

$$\mathcal{S}(A_j,b) = \mathrm{span}\{A_j^{k-1}b, A_j^{k-2}b, \ldots, A_jb, b\}$$

and the label is determined by the nilpotent matrix A_j.

B.2. Base vector class

In the second setting, each class corresponds to a distinct base vector. We randomly generate M base vectors $b_1, b_2, \ldots, b_M \in \mathbb{R}^n$. The base vectors are sampled $b \sim \mathcal{N}(0, I_n)$, then normalized to unit norm. Class base vectors are accepted only if their pairwise angular separation satisfies the minimum angle threshold, and are resampled until all class vectors meet the constraint. The experiments in this paper use angle$= 15°$. For a sample from class j, we draw a random nilpotent matrix $A \in \mathbb{R}^{n\times n}$ with nilpotency index k and form the subspace

$$\mathcal{S}(A,b_j) = \mathrm{span}\{A^{k-1}b_j, A^{k-2}b_j, \ldots, Ab_j, b_j\}.$$

and the label is determined by the base vector b_j.

B.3. Generating random nilpotent matrices for the cyclic subspace dataset

To generate random nilpotent matrices with specified ambient dimension n and nilpotency index k, we use a similarity transformation of an embedded shift matrix. Let $S \in \mathbb{R}^{k\times k}$ denote the upper shift matrix defined by

$$S_{ij} = \delta_{i+1,j},$$

where $\delta_{i+1,j}$ is the Kronecker delta. Equivalently, S has ones on the superdiagonal and zeros elsewhere:

$$S = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

This matrix satisfies $S^k = 0$, while $S^\ell \neq 0$ for all $\ell < k$, and therefore has nilpotency index k. For $k \leq n$, we embed S into the upper-left block of an $n \times n$ matrix of zeros:

$$C = \begin{pmatrix} S & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{n\times n}.$$

The matrix C is nilpotent with nilpotency index k. To randomize the coordinate representation while preserving nilpotency, we sample an invertible matrix $F \in \mathbb{R}^{n\times n}$ ($F_{ij} \sim \mathcal{N}(0,1)$, and resample until F is full rank) and define

$$A = FCF^{-1}.$$

Since similarity transformations preserve powers and Jordan structure, the resulting matrix A remains nilpotent with the same nilpotency index.

Lemma 1 *A is a random $n \times n$ nilpotent matrix with index of nilpotency k, for values of n and k such that $k \leq n$.*

Proof [Proof of Lemma 1] Fix $k \leq n$. We want to show that $A^k = (FCF^{-1})^k = \mathbf{0}$, where F is a full rank random matrix and C is the $n \times n$ upper shift matrix.

First, we show that all finite-dimensional shift matrices are nilpotent, i.e. that $S^k = \mathbf{0}$. By definition of the shift matrix, S^1 has ones on the superdiagonal and zeros elsewhere. Then, S^2 has all zeros, except the 2nd superdiagonal, i.e. applying S to S^1 shifts all of the column vectors one position down (to the right). Applying S for each power shifts all of the column vectors to the right one step each time, so S^{k-1} has all zeros, except the $(k-1)$th superdiagonal and $S^k = S(S^{k-1})$ therefore is the zero matrix, $S^k = \mathbf{0}$.

Next, we want to show that $C = \begin{pmatrix} S & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$ is nilpotent. By construction, $C^n = \begin{pmatrix} S^n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$, and since S is a shift matrix (and thus nilpotent), we know that $S^k = \mathbf{0}$. Since $n \geq k$, it is also true that $S^n = \mathbf{0}$ and thus, $C^n = \begin{pmatrix} S^n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \mathbf{0}$. This implies that C is nilpotent, and its index of nilpotency is k.

Finally, we take a random full rank matrix, $F \in \mathbb{R}^{n\times n}$, and its inverse, F^{-1}, and apply a similarity transform to C, $FCF^{-1} \in \mathbb{R}^{n\times n}$. Then, by construction C and FCF^{-1} are similar matrices and thus have the same eigenvalues. A matrix is nilpotent if and only if all of its eigenvalues are zero. Thus, C and FCF^{-1} have the same eigenvalues and since C is nilpotent with all zero eigenvalues, FCF^{-1} must also be nilpotent with all zero eigenvalues. So, we have that FCF^{-1} is a nilpotent matrix and its index of nilpotency is k. ■

Appendix C. Trajectory dataset example

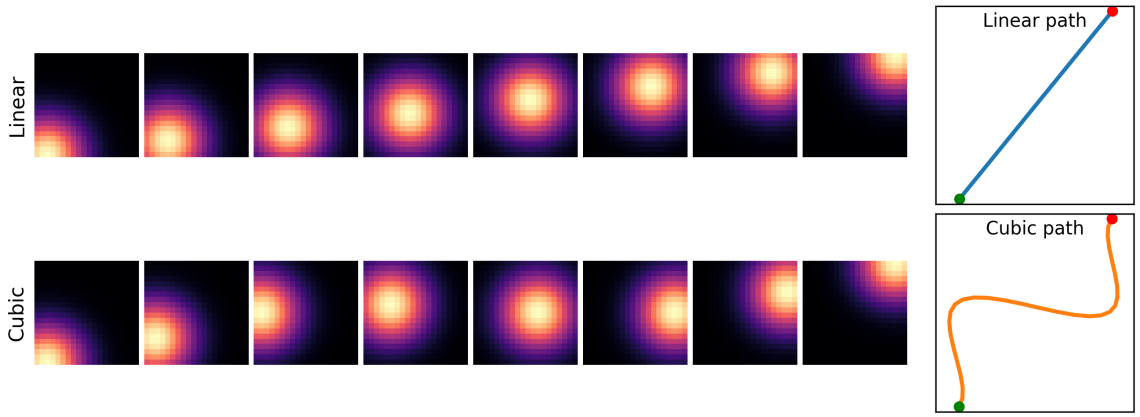


Figure 4: Two samples (one from each class) from the trajectory dataset: image size 20×20, Gaussian variance $\sigma^2 = 25.0$. Showing 8 frames (left-to-right) with each full trajectory path on the right. Start and end points marked by green and red, respectively.

Each video is generated by first sampling a starting point uniformly at random from the boundary of the image. An endpoint is then chosen on the opposite boundary so that the segment connecting the start and endpoint passes through the image center. From this same start-end pair, two trajectory types are constructed: a linear one, where the trajectory location μ_t moves along the straight center-crossing segment, and a cubic one, where μ_t follows a smooth curved path between the same endpoints with a center-alignment adjustment so that the trajectory also passes through the image midpoint. Thus, linear and cubic samples share common endpoints and differ only in how μ_t evolves between them.

Each frame contains a single Gaussian intensity profile centered at the current trajectory location. Specifically, for a trajectory location $\mu_t \in \mathbb{R}^2$ at frame t, the image intensity at pixel location $x \in \mathbb{R}^2$ is given by

$$I_t(x) = \exp\left(-\frac{\|x - \mu_t\|_2^2}{2\sigma^2}\right),$$

where σ^2 controls the spatial spread of the Gaussian. Frames are not cumulative, so each frame shows only the Gaussian at its current location. Figure 4 shows two samples from the dataset. The controllable parameters for this dataset are the variance of the Gausian, the number of frames, and the resolution (which determines the dimension of the ambient

space). While we only include two trajectory types (linear and cubic), the construction is readily extensible.

Appendix D. Model and data parameters

This section will report the specific parameter choices for each model and experiment.

D.1. GrAE and GrVAE

The encoder and decoder architecture are shared between the GrAE and GrVAE. In both cases, the input data consists of orthonormal matrices $\{U\}_{m=1}^{M} \in \mathbb{R}^{n\times k}$ such that $U^\top U = I_k$. Each matrix is mapped to a projection matrix representative, $U \to P = U^\top U \in \mathbb{R}^{n\times n}$ and then the upper triangular elements of P are vectorized $\text{vec}_\triangle(P) \in \mathbb{R}^{\frac{n\times(n+1)}{2}}$ to retain only the coordinates needed to uniquely represent the symmetric projection matrix. These transformations are done without using learnable parameters. This $\frac{n\times(n+1)}{2}$ vector is then mapped to the d-dimensional latent space using 3 fully-connected layers of an MLP. The first two are of size $\frac{n\times(n+1)}{2}\times\texttt{HD}$ (where HD denotes the number of nodes in each hidden layer) and $\texttt{HD}\times\texttt{HD}$, respectively, with GELU activations, and a final fully-connected layer of size $\texttt{HD}\times d$ maps inputs to the latent space. For the GrVAE model, another fully-connected layer of size $\texttt{HD}\times d$ is used to encode the log-variance term.

The decoder consists of a 3-layer MLP of with layers of size $d\times\texttt{HD}$, $\texttt{HD}\times\texttt{HD}$, and $\texttt{HD}\times(n\times k)$, with GELU activations after the first two layers. The resulting vector of dimension $n \times k$ is reshaped into an $n \times k$ matrix and orthonormalized using the thin QR-decomposition. Total per layer number of learnable parameters for GrAE and GrVAE can be seen in Table 2.

For the experiments in this work, the GrAE and GrVAE are both defined with $\texttt{HD} = 128$ and mapped to a latent dimension with $d = 6$. The learning rate is set to $\text{LR} = 7e-4$ and we train for 100 epochs using Adam optimizer. For the GrAE, we optimize using a reconstruction loss defined via the squared chordal distance,

$$\mathcal{L}_{\text{rec}}(U,\hat{U}) = d^2_{\text{ch}}(U,\hat{U}) = k - \|U^\top\hat{U}\|_F^2\,.$$

The loss function used for training the GrVAE is the same with the addition of a KL-divergence term, balanced using the regularization parameter β,

$$\mathcal{L}(\phi,\theta) \;=\; \mathbb{E}_{q_\phi(z|U)}\big[d^2_{ch}\big(U,\, f_{\text{dec}}(z)\big)\big] \;+\; \beta\,\text{KL}(q_\phi(z\mid U)\,\|\,\mathcal{N}(0,I_d))\,,$$

where we use $\beta = 0.5$ as the weight on the KL-divergence, and perform KL annealing for 10 epochs. Classification is performed for GrAE and GrVAE by a MLP classification layer on the learned latent space representations.

D.2. GrNet

We use the GrNet implementation from (15) ported for use with PyTorch. For all experiments, GrNet is trained with $\texttt{NumProj} = 8$ and $\texttt{PoolStride} = 2$ (used in the `ProjPooling` layer to average across projection matrices). The learning rate is set to $\texttt{LR} = 0.1$ and the algorithm is trained for 10 epochs (which has been checked for convergence). Two blocks of $\texttt{FrMap(DIM)} \to \texttt{ReOrth} \to \texttt{ProjMap} \to \texttt{ProjPooling(PoolStride)} \to \texttt{OrthMap(TopEigen)}$

Table 2: Learnable parameter counts in GrAE and GrVAE models for a point $U \in \mathrm{Gr}(k, n)$ with $\texttt{HD}$ the number of nodes in each hidden layer, and d the dimension of the latent space.

Method	Input	MLP	Encoding	Decoding	MLP	Output
GrAE	$\frac{n(n+1)}{2}\texttt{HD}$	$\texttt{HD}^2$	$d\texttt{HD}$	$d\texttt{HD}$	$\texttt{HD}^2$	$nk\texttt{HD}$
GrVAE	$\frac{n(n+1)}{2}\texttt{HD}$	$\texttt{HD}^2$	$2d\texttt{HD}$	$d\texttt{HD}$	$\texttt{HD}^2$	$nk\texttt{HD}$

are used, where the parameters for `DIM`, `PoolStride`, and `TopEigen` are specified in each of the experiments, followed by a `ProjMap` layer that is flattened into the latent space and a fully-connect layer is applied to achieve the final predictions. The model is trained with Riemannian stochastic gradient descent using cross entropy loss, with tools courtesy of RiemOpt (31) and GeoOpt (17).

D.3. kSVM

We use the kernel support vector machine (kSVM) implementation (3) from scikit-learn (26). The kernel is precomputed from the pairwise chordal distances between subspaces. To map these distances to similarity scores in $[0, 1]$, each entry of the distance matrix is transformed as

$$s_{ij} = 1 - \frac{d_{ij}}{\max_{i,j} d_{ij}},$$

where d_{ij} is the chordal distance between subspaces i and j. The resulting similarity matrix is passed as a precomputed kernel to the classifier with regularization parameter $C = 1$. For multi-class tasks, we use the standard one-vs-one scheme provided by the package.

D.4. Synthetic Subspace Dataset: Centroid Classification

For the results shown in Figure 2(*a*), 10 centroids are chosen uniformly at random from the Grassmann manifold, $\mathrm{Gr}(50, k)$ where $k = 40, 41, \ldots, 49$ varies across experimental scenarios. 50 random samples drawn for each centroid with a noise variance $\sigma_n^2 = 3.5$, for a total of 500 samples divided for a 50/50 train/validation split. For each subspace dimension, the average results of 5 independent trials is reported. For the GrNet, both FrMap layers have dimension 50 (e.g., no dimension reduction) and all eigenvectors are used.

D.5. Cyclic Subspace Dataset: Base Vector Classification

For the results shown in Figure 2(*b*), 10 base vectors in $\mathbb{R}^{10}$ chosen at random with a minimum pairwise angle of 15 degrees. For each class, 50 samples were generated by selecting a nilpotent matrix at random and left multiplying it by the base vector to create subspaces of dimension $2, 3, 4, \ldots, 9$ for each scenario, respectively. Subspaces of dimension 1 are not used for comparison because the samples consist solely of one of the 10 base vectors, in line with the construction described in Section 3.2.2, and the classification task becomes noise-free. We have verified that all methods have perfect classification accuracy for such 1-dimensional subspaces. The result of this subspace generation process is a total of 500 samples for each choice of subspace dimension, which are divided into a 50/50 train/validation split. For each subspace dimension, the average results of 5 independent trials is reported, with the

shaded region representing ± 1 standard deviation about the mean. The GrNet model is instantiated with `FrMap` dimensions $[10, 10]$ (the full ambient dimension), and `TopEigen` equal to the subspace dimension.

D.6. Cyclic Subspace Dataset: Nilpotent Matrix Classification

For the results shown in Figure 2(*c*), 10 nilpotent matrices in $\mathbb{R}^{10\times 10}$ are chosen at random. For each class, 50 samples are generated by randomly drawing a base vector in $\mathbb{R}^{10}$ and right multiplying by the nilpotent matrix products to create subspaces of dimension $2, 3, 4, \ldots, 9$ for each scenario, respectively. Subspaces of dimension 1 are not used for comparison because the samples consist solely of one of the 10 base vectors, in line with the construction described in Section 3.2.2, and the classification task becomes pure noise. We have verified that all methods approximate a random baseline for such 1-dimensional subspaces. The result of this subspace generation process is a total of 500 samples for each choice of subspace dimension, which are divided into a 50/50 train/validation split. For each subspace dimension, the average results of 5 independent trials is reported, with the shaded region representing ± 1 standard deviation about the mean. The GrNet model is instantiated with `FrMap` dimensions $[10, 10]$ (the full ambient dimension), and `TopEigen` equal to the subspace dimension.

D.7. Trajectory Video Dataset: Trajectory Type Classification

For the results shown in Figure 2(*d*), a collection of 200 videos with 100 frames were created according to the model described in Section C, with 100 linear and 100 cubic trajectories. The dataset is class balanced with matched pairs so that each linear trajectory and its corresponding cubic trajectory share identical start and end points. The resulting frames are of size $= 20 \times 20$ and variance of Gaussian intensity profile is $\sigma_b^2 = 0.5$. The video frames are raster scanned and concatenated into matrices of size 100×400, and the thin singular value decomposition is used to generate subspaces of dimension $5, 10, 15, 20, 25, 30, 35$, and 40 that span $25\%, 47\%, 65\%, 79\%, 87\%, 93\%, 96\%$, and 98% of the variance in the frames, respectively. For each subspace dimension, the average results of 5 independent trials is reported, with the shaded region representing ± 1 standard deviation about the mean. The GrNet model is instantiated with `FrMap` dimensions $[400, 100]$ (the full ambient dimension and one fourth of the ambient dimension), and `TopEigen` equal to the subspace dimension.

D.8. Synthetic Subspace Dataset: Timing Comparison

To analyze the effect of the `FrMap` dimensions on accuracy and computation time, we chose 10 centroids uniformly at random from the Grassmann manifold, $\mathrm{Gr}(n, 5)$ and let the ambient dimension vary for $n = 25, 50, 75, 100, 250, 500$ across experimental scenarios. 50 random samples drawn for each centroid with a noise variance $\sigma_n^2 = 2.5$, for a total of 500 samples divided for a 50/50 train/validation split. For each subspace dimension, the average results of 5 independent trials is reported.

We train two GrNet models to demonstrate the effect of changing the FrMap dimensions on training time and classifier accuracy. The first GrNet (referred to as GrNet) has `FrMap` dimension = [ambient dimension, int(ambient dimension/2)] and the second GrNet (referred to as GrNet-reduced) has `FrMap` dimension = [int(ambient dimension/2), int(ambient dimension/4)] using the top 5 eigenvectors. The results can be seen in Figure 5.

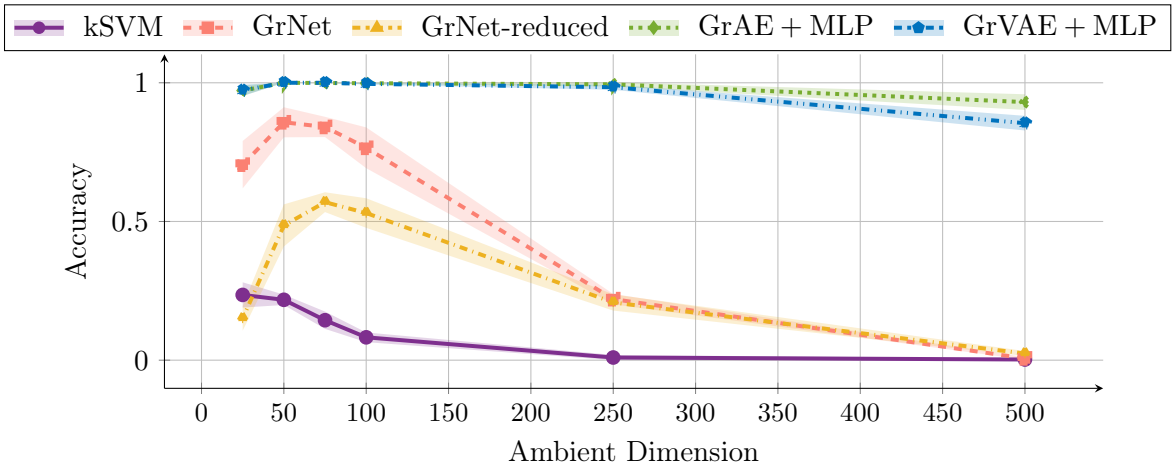


Figure 5: Computational scalability for classifying synthetic subspace data as a function of ambient dimension. The subspace dimension is fixed at 5 and the noise variance is $\sigma_n^2 = 2.5$. Each curve shows the mean of 5 independent trials and the shaded regions denote ± 1 standard deviation about the mean. GrNet has `FrMap` dimension = [ambient dimension, int(ambient dimension/2)] and GrNet-reduced has `FrMap` dimension = [int(ambient dimension/2), int(ambient dimension/4)] using the top 5 eigenvectors.

Table 3: Computational scalability for classifying synthetic subspace data as a function of ambient dimension. The subspace dimension is fixed at 5 and the noise variance is $\sigma_n^2 = 2.5$.

	Ambient Dim.	**25**	**50**	**75**	**100**	**250**	**500**
Encoding + Prediction Time (seconds)	kSVM	0.03±0.001	0.035±0.002	0.038±0.002	0.041±0.002	0.068±0.001	0.11±0.001
	GrNet	0.675±0.001	3.204±0.052	6.376±0.062	8.803±0.063	24.227±0.205	70.036±0.377
	GrNet-reduced	0.62±0.001	0.675±0.002	2.847±0.049	3.213±0.024	9.422±0.084	24.276±0.205
	GrAE + MLP	0.157±0.001	0.159±0.001	0.159±0	0.158±0.001	0.167±0.008	0.165±0.001
	GrVAE + MLP	0.268±0.001	0.271±0.001	0.27±0.001	0.268±0.003	0.272±0.001	0.277±0.002

Appendix E. Computing resources

Experiments were run using on-premesis high-performance computing resources with the following specs: CPU = AMD EPYC 7502, Cores per node = 64, Clock rate = 2.5GHz boost to 3.35GHz, CPU L3 cache = 128 MB, FLOPS per clock cycle = 16, Memory speed = 3200Mhz, Single core memory = 22 GB/s, Node memory = 256 GB, InfiniBand speed = 100 Gb/s, Peak gigaflops/node = 2560, 1 Node with 1 Core and 1 A100 GPUs.