PIEFS: Physics-Informed Eigenfunction Features with Learnable Scaling

Varvara Nazarenko [1] **Timur Lidzhiev** [1] **Alexander Tarakanov** [1 2]

Abstract

Spectral methods are widely used to construct representations from the geometry of data, but they often rely on a fixed kernel, graph Laplacian, or manually selected feature scaling. We propose Physics-Informed Eigenfunction Features with Learnable Scaling (PIEFS), a supervised neural representation-learning framework with a spectral inductive bias, based on a modified Dirichlet energy. In PIEFS, scalar coordinate maps are trained under empirical Gram orthogonality, a supervised linear readout, and a Dirichlet penalty in which the input gradient is transformed by a learnable metric $A(x) = \Lambda(x)U(x)$. The diagonal factor $\Lambda(x)$ controls anisotropic scaling, while the orthogonal factor $U(x)$ is parameterized by a structured product of Givens rotations. This construction yields task-adaptive Dirichlet-regularized coordinates rather than eigenfunctions of a fixed supervision-independent operator. Experiments on synthetic, tabular, and image-based benchmarks study the effect of identity, diagonal, and rotation-scaling metrics, and compare the resulting coordinates with classical baselines and NeuralEF. The results support PIEFS as a compact supervised spectral representation method and identify optimization stability, validation on explicit operator eigenproblems, and richer metric parameterizations as the main directions for future work.

1. Introduction

Contributions.

- We formulate PIEFS as a supervised spectral-style representation method with sequential coordinate maps, empirical Gram orthogonality, cross-entropy on a linear readout, and a modified Dirichlet penalty.
- We study three metric settings inside the Dirichlet term: identity (OFF), volume-preserving diagonal scaling (DIAG), and diagonal scaling after a structured Givens rotation (TROTTER; apply order fixed as in Sec. 2.4).
- We evaluate the same training pipeline across five benchmark settings (Table 2) against RF, LR, PCA+LR, and NeuralEF (Deng et al., 2022), with visualizations of learned maps and training dynamics.

Graph-based spectral methods build a Laplacian on finite samples and use its eigenmaps as geometry-aware features (Gomez-Chova et al., 2008; Kunegis et al.). Their theory connects graph eigenfunctions to Laplace–Beltrami modes when data concentrate on a manifold (Belkin & Niyogi, 2008), which explains strong performance in clustering and semi-supervised learning (Ng et al., 2001). Two bottlenecks motivate mesh-free alternatives: eigencomputation cost grows with dataset size (Ford, 2015), and test-time evaluation at new points typically requires rebuilding the graph and its spectrum (Belkin & Niyogi, 2008). Neural surrogates for operator eigenproblems are an active line of work (Jin et al., 2020; Deng et al., 2022; Choo et al., 2020). Automatic differentiation makes it feasible to minimize Rayleigh-type objectives with PDE-style regularity while remaining discretization-free in the input domain (Feld et al., 2019; Lagaris et al., 1997).

We study *physics-informed eigenfunction features with learnable scaling* (PIEFS), implemented as learnable-metric Dirichlet coordinates: a sequence of scalar maps (ϕ_j) trained with a modified Dirichlet penalty that applies a data-dependent linear map $\mathbf{A}(\mathbf{x})$ to gradients (Evans, 2022), together with batch Gram orthogonality and cross-entropy on linear logits. The outer schedule freezes earlier coordinates and updates one map at a time, yielding cheap inference once training finishes. Section 2 derives the losses and weighting; Section 3 reports benchmarks and visualizations; Section 4 discusses empirical findings, limitations, and future directions.

[1]Faculty of Computer Science, HSE University, Moscow, Russia [2]AI VK, Moscow, Russia. Correspondence to: Varvara Nazarenko <varunaza@gmail.com>, Timur Lidzhiev <trlidzhiev@gmail.com>, Alexander Tarakanov <atarakanov@hse.ru>.

Proceedings of the AI4Physics Workshop at the 43rd International Conference on Machine Learning (AI4Physics@ICML 2026), Seoul, South Korea. 2026. Copyright 2026 by the author(s).

2. Methodology

This section defines the PIEFS objective, the sequential training schedule, and the learnable metric used in the Dirichlet penalty. We first derive the loss as a weighted sum of three terms, then the dynamic weights, then the learnable gradient metric $\mathbf{A}(\mathbf{x})$. An optional graph-Laplacian warm-start could set auxiliary targets but is *not* used in the experiments below.

2.1. Loss Function

We motivate the coordinate construction with a Rayleigh-type variational picture: classical spectral problems minimize a Rayleigh quotient of Dirichlet form subject to orthogonality, whereas here we minimize a *composite* objective that couples modified Dirichlet energy, a batch Gram surrogate for orthogonality, and cross-entropy on linear logits—this is not equivalent to solving a fixed self-adjoint eigenproblem on $L^2(p)$. We consider a probability distribution with density $p(\mathbf{x})$. This distribution defines the following inner product:

$$\langle f_1, f_2\rangle_0 = \int f_1(\mathbf{x}) f_2(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \tag{1}$$

Here f_1 and f_2 are functions with finite second moments.

The corresponding Dirichlet bilinear form is:

$$\langle f_1, f_2\rangle_{\text{de}} = \int \sum_{a=1}^{d} \frac{\partial f_1(\mathbf{x})}{\partial x_a} \frac{\partial f_2(\mathbf{x})}{\partial x_a} \, p(\mathbf{x}) \, \mathrm{d}\mathbf{x} \tag{2}$$

Here d is the dimension of $\mathbf{x}$.

This product becomes a regular Dirichlet Energy for $f_1 = f_2$:

$$\begin{aligned} \langle f, f\rangle_{\text{de}} &= \int \left(\sum_{a=1}^{d} \frac{\partial f(\mathbf{x})}{\partial x_a} \frac{\partial f(\mathbf{x})}{\partial x_a} \right) p(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\ &= \int \|\nabla f(\mathbf{x})\|^2 \, p(\mathbf{x}) \, \mathrm{d}\mathbf{x} \end{aligned} \tag{3}$$

We introduce a learnable matrix $\mathbf{A}(\mathbf{x})$ to modify the Dirichlet energy and the corresponding bilinear form:

$$\langle f_1, f_2\rangle_{\text{mde}} = \int \big(\mathbf{A}(\mathbf{x})\nabla f_1(\mathbf{x})\big)^\top \big(\mathbf{A}(\mathbf{x})\nabla f_2(\mathbf{x})\big) \, p(\mathbf{x}) \, \mathrm{d}\mathbf{x} \tag{4}$$

For $f_1{=}f_2{=}f$, this gives the Modified Dirichlet Energy:

$$\langle f, f\rangle_{\text{mde}} = \int \big\|\mathbf{A}(\mathbf{x})\nabla f(\mathbf{x})\big\|^2 \, p(\mathbf{x}) \, \mathrm{d}\mathbf{x} \tag{5}$$

The goal is to obtain K coordinate maps $\phi_1(\mathbf{x}), \ldots, \phi_K(\mathbf{x})$ whose Dirichlet energy is small under (5) while remaining approximately orthonormal and predictive under cross-entropy; block-coordinate descent on the composite loss approximates this Rayleigh-type trade-off rather than enforcing a single Rayleigh quotient. Matrix $\mathbf{A}(\mathbf{x})$ is learned jointly with the basis so that a linear classifier atop $(\phi_1, \ldots, \phi_K)$ obtains high accuracy.

In other words, we have a loss function that corresponds to the MDE:

$$\mathcal{L}_{\text{mde}} = \int \big\|\mathbf{A}(\mathbf{x})\nabla\phi_k(\mathbf{x})\big\|^2 \, p(\mathbf{x}) \, \mathrm{d}\mathbf{x} \tag{6}$$

where ϕ_k is the currently active coordinate map at outer index k during the sequential training schedule. There is a loss function that enforces the orthonormality constraint at the population level:

$$\begin{aligned} \mathcal{L}_{\text{orth}} &= \big\|\mathbb{E}_p[\phi\phi^\top] - I_K\big\|_F^2 \\ &= \sum_{\alpha,\beta=1}^{K} \Big(\mathbb{E}_p\big[\phi_\alpha(\mathbf{x})\,\phi_\beta(\mathbf{x})\big] - \delta_{\alpha\beta}\Big)^2, \end{aligned} \tag{7}$$

where $\delta_{\alpha\beta}$ is the Kronecker symbol.

Remark 2.1 (Monte-Carlo Gram estimator). The empirical Gram matrix $\hat{C}_k = \frac{1}{B}\Phi_{1:k}^\top\Phi_{1:k}$ is an unbiased estimator of $\mathbb{E}_p[\phi\phi^\top]$, so $\|\hat{C}_k - I_k\|_F^2$ minimised by SGD is a plug-in (bias-of-the-mean) estimator of (7); the two coincide as $B \to \infty$, and the residual gap is $O(1/B)$ in expectation (variance of $\phi_\alpha\phi_\beta$ across the batch).

Features $\phi_1(\mathbf{x}), \ldots, \phi_K(\mathbf{x})$ are used to train the classifier:

$$\log\big(p(l|\mathbf{x})\big) \propto B_l + \sum_{\gamma=1}^{K} W_{l\gamma}\phi_\gamma(\mathbf{x}) \tag{8}$$

Here B is the bias, W stores the weights, $\log(p(l|\mathbf{x}))$ denotes the logits. Weights and bias are learned from minimization of the cross-entropy loss function $\mathcal{L}_{\text{class}}$.

Finally, all three loss functions are combined together via weighting:

$$\begin{aligned} \mathcal{L} = \mathrm{sg}(w_{\text{orth}})\,\mathcal{L}_{\text{orth}} + \mathrm{sg}(w_{\text{class}})\,\mathcal{L}_{\text{class}} \\ + \mathrm{sg}(w_{\text{mde}})\,\mathcal{L}_{\text{mde}} \end{aligned} \tag{9}$$

Here $\mathrm{sg}(\cdot)$ denotes a stop-gradient operation: the adaptive weights are treated as constants during back-propagation.

Integrals over p enter the loss functions as in Eq. (6) and Eq. (7). In practice, the density of the distribution is not known. Therefore, Monte Carlo estimates replace these integrals in Eq. (6) and Eq. (7). Operationally, each integral is replaced by a batch average.

2.2. Sequential Neural Coordinate Maps

We represent each coordinate map ϕ_j by a scalar neural ansatz with parameters θ_j, chosen disjoint across j. At outer

Algorithm 1 Sequential block-coordinate descent for PIEFS.

for $k = 1$ to K **do**
  for each inner iteration until the budget is exhausted **do**
    Fix θ_j for all $j < k$.
    Evaluate $\Phi_{1:k}$ by applying $\phi_j(\cdot;\theta_j)$ with fixed θ_j for $j < k$ and the current iterate θ_k for $j = k$.
    Accumulate $\|\hat{C}_k - I_k\|_F^2$, the Dirichlet contribution $\|\mathbf{A}\nabla\phi_k\|^2$, and cross-entropy computed on logits whose coordinates with index $j > k$ are fixed to zero.
    Refresh $(w_{\text{orth}}, w_{\text{class}}, w_{\text{mde}})$ via (10) with $g_k = \|\hat{C}_k - I_k\|_F^2$, treating these weights as constants in the differentiation of $\mathcal{L}$.
  end for
end for

index k we optimize only θ_k; for every $j < k$ the maps enter the Gram and classification losses as *fixed* nonlinearities, so gradients with respect to θ_j vanish identically and the empirical Gram $\hat{C}_k$ is consistent with a block-coordinate descent on $(\theta_1, \ldots, \theta_K)$. The Dirichlet penalty involves $\nabla_{\mathbf{x}}\phi_k$ alone. The classifier logits append zeros in coordinates that have not yet been activated; that is, coordinates with index $j > k$ contribute zero to the classification loss throughout their development phase, which is equivalent to evaluating the affine readout on the leading k-dimensional coordinate subspace until the schedule completes.

We do not identify (ϕ_j) with eigenfunctions of a prescribed self-adjoint operator on $L^2(p)$; rather, they form orthogonal, Dirichlet-regularized coordinates biased by cross-entropy, analogously to learned spectral features (Deng et al., 2022).

The BasisSet learns (ϕ_j) sequentially (one active outer index at a time) while MetricNet supplies $\mathbf{A}(\mathbf{x}) = \mathbf{\Lambda}(\mathbf{x})\mathbf{U}(\mathbf{x})$; both couple through (5)–(9).

Algorithm 1 lists the sequential optimization schedule. At outer index k, the frozen stack $\mathbf{\Phi}_{1:k-1}$ and the trainable map ϕ_k jointly feed $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{class}}$, while $\mathcal{L}_{\text{mde}}$ couples $\mathbf{A}(\mathbf{x})$ to $\nabla_{\mathbf{x}}\phi_k$ (see (5)); only θ_k receives gradients (all θ_j with $j \neq k$ are detached via stop-gradient). The losses aggregate as in (9) with weights (10). After the schedule completes, inference uses the affine readout on the leading coordinates (8).

2.3. Loss Function Weighting

We use the following hierarchy of loss functions:

- orthonormality
- cross-entropy for classification
- MDE loss function

The objective is to learn representative features first and train the classifier with them. The final stage of MDE minimization provides regularization and improves the expressive power of constructed features.

Throughout all reported experiments, we fix two temperature scales: T_{orth}=0.1 and T_{class}=0.5. These values were selected via informal grid search on the MNIST validation set and held constant across all datasets; a formal hyperparameter ablation remains an open direction for future work.

The practical way to achieve the hierarchy of concern is dynamic weighting of loss functions that is updated each iteration of loss function minimization: Let $g_k = \|\hat{C}_k - I_k\|_F^2$.

$$\begin{cases} w_{\text{orth}} \propto 1, \\ w_{\text{class}} \propto \exp\big(-g_k/T_{\text{orth}}\big), \\ w_{\text{mde}} \propto \exp\Big(-\max\big(g_k/T_{\text{orth}},\ \mathcal{L}_{\text{class}}/T_{\text{class}}\big)\Big). \end{cases} \tag{10}$$

Here T_{orth} and T_{class} are temperature-style scales fixed *a priori* for the reported runs (T_{class}=0.5); an optional graph-based warm-start could in principle set them from held-out logistic loss (Gomez-Chova et al., 2008), but we do not use that stage here.

The weighting in (10) employs exponential dampening when loss values are large. The exponential form was chosen empirically; alternative functional forms such as $1/(1+g_k)$ or polynomial dampening were tested informally and found less robust in early experiments, though systematic ablation of this choice remains an open question for future work. Equation (10) implements the desired priority order: when g_k is large, the classification and MDE terms are suppressed; the MDE term becomes active only after both Gram and classification residuals are moderate.

2.4. Matrix Parametrization and TROTTER-Fixed Rotations

Equation (5) is evaluated using the gradient of the active map ϕ_k only (Sec. 2.2). We summarize the parametrizations for $\mathbf{A}(\mathbf{x})$ that appear in Table 2.

At the continuum level one may factor

$$\mathbf{A}(\mathbf{x}) = \mathbf{\Lambda}(\mathbf{x})\mathbf{U}(\mathbf{x}), \tag{11}$$

with diagonal $\mathbf{\Lambda}$ and orthogonal $\mathbf{U}$. Throughout training, $\mathbf{A}(\mathbf{x})$ is fit jointly with the coordinate nets and the classifier through the computation graph that evaluates (9); cross-entropy and the adaptive weights in (10) therefore couple $\mathbf{A}$ to label information, so it should not be read as an unsupervised surrogate for a fixed physics-only differential operator.

Diagonal scaling. The log-eigenvalues follow a zero-mean

MLP head so that

$$\sum_{i=1}^{d} \log \lambda_i(\mathbf{x}) = 0 \quad \Rightarrow \quad \det \mathbf{\Lambda}(\mathbf{x}) = 1. \tag{12}$$

A fully expressive $\mathrm{SO}(d)$ parametrization would require substantially more degrees of freedom, for example through dense skew-symmetric generators or matrix exponentials. Such variants are computationally expensive at the ambient dimensions considered here. We therefore use a structured Givens product as a stable low-parameter baseline.

Trained metric variants. **Metric disabled (OFF).** We set $\mathbf{A} = \mathbf{I}$ identically while still learning orthogonal features; Euclidean Dirichlet energies arise as the special case of (5).

Diagonal (DIAG). $\mathbf{A}(\mathbf{x}) = \mathbf{\Lambda}(\mathbf{x})$ with coordinates $\lambda_i(\mathbf{x}) = \exp(z_i(\mathbf{x}) - \bar{z}(\mathbf{x}))$ under (12).

Trotter rotation (TROTTER). We parameterize $\mathbf{U}(\mathbf{x})$ as a product of $d-1$ Givens rotations,

$$\mathbf{U}(\omega) = R_{d-2}(\omega_{d-2}) \cdots R_1(\omega_1)\, R_0(\omega_0), \tag{13}$$

with angles $\omega_i(\mathbf{x}) = \pi \tanh\big(\mathrm{MLP}(\mathbf{x})_i\big)$. Each plane rotation R_j lies in $\mathrm{SO}(d)$, so the product is orthogonal with determinant one; this fixed chain is a *structured* subset of orthogonal maps and does not parameterize all of $\mathrm{SO}(d)$ (which has dimension $d(d-1)/2$), but it remains an expressive, low-parameter family for high ambient d. Bounding angles via $\omega_i = \pi \tanh(\cdot)$ keeps them in $[-\pi, \pi]$, improving numerical stability and preventing unbounded extrapolation in the angle parameterization. The learned metric acts on tangent vectors $\mathbf{v}$—in practice $\mathbf{v} = \nabla\phi_k$—by rotating first and rescaling afterwards,

$$\mathbf{A}(\mathbf{x})\mathbf{v} = \mathbf{\Lambda}(\mathbf{x})\big(\mathbf{U}(\mathbf{x})\mathbf{v}\big), \tag{14}$$

which we adopt throughout this work. **Gradient flow.** Equation (14) exposes both factors to stochastic gradients via $\|\mathbf{A}(\mathbf{x})\mathbf{v}\|^2 = \sum_i \lambda_i^2 \big((\mathbf{U}\mathbf{v})_i\big)^2$. If $\mathbf{\Lambda}$ is applied before $\mathbf{U}$, the Euclidean energy $\|\mathbf{A}\mathbf{v}\|^2$ collapses to a diagonal-only functional and no longer differentiates through the rotational factor; all experiments below therefore adopt (14). **Network architecture.** All basis functions and metric networks use three fully-connected hidden layers of width 64 with ReLU activations. Basis functions ϕ_k output a single scalar; metric networks output K values (for the diagonal variant) or $(d-1)$ angle values (for the Trotter variant, one per adjacent pair rotation).

Numerical contraction. Adjacent plane rotations are fused into even–odd sweeps with depth bounded independently of the nominal Givens count, following standard practice for compact differentiable orthogonal maps.

Table 1. Training components used in the reported experiments.

Component	Role
Sequential coordinate training	Trains one map ϕ_k at a time
Gram orthogonality	Empirical penalty encouraging approximate coordinate independence
Cross-entropy readout	Linear classifier on coordinates; makes features task-adaptive
Modified Dirichlet penalty	Encourages smooth coordinates via weighted input gradients
Graph-Laplacian pretraining	Optional warm start (not used in reported runs)
Metric OFF	Dirichlet uses $\mathbf{A}(\mathbf{x}) = \mathbf{I}$
Metric DIAG	Dirichlet uses diagonal scaling $\mathbf{A}(\mathbf{x}) = \mathbf{\Lambda}(\mathbf{x})$
Metric TROTTER	Dirichlet uses $\mathbf{A}(\mathbf{x}) = \mathbf{\Lambda}(\mathbf{x})\mathbf{U}(\mathbf{x})$ (apply order as in Sec. 2.4)

PIEFS-TROTTER on Two Moons, Circles, and HTRU2 aggregates five independent optimization trajectories, each with 60,000 gradient steps, under (14). The same five-seed protocol applies to MNIST multiclass TROTTER at 60,000 steps, whereas CIFAR-10 ResNet-18 embeddings use 120,000 total steps per seed when TROTTER is enabled.

2.5. Pretraining

Optional Graph-Laplacian pretraining could fit eigenmaps on a subsample and calibrate T_{class} from a linear probe (Gomez-Chova et al., 2008). **All experiments below** disable this stage, fix $T_{\text{class}} = 0.5$, and train PIEFS from scratch without graph-based warm starts. A small five-seed ablation on HTRU2 with the warm start enabled is reported in App. B.

3. Experimental Results

Benchmark overview. We compare PIEFS to classical baselines and NeuralEF (Deng et al., 2022) on five benchmark settings summarized in Table 2; splits, coordinate counts K, validation vs. test reporting, and baseline choices are detailed in Sec. 3.1.

Default PIEFS protocol. Unless otherwise stated, all PIEFS runs use the same training procedure: sequential coordinate-map optimization, empirical Gram orthogonality, cross-entropy on a linear readout, dynamic loss weights, no graph-Laplacian pretraining, and no input augmentation. The columns **off**, **diag**, and **trotter** differ only in the metric $\mathbf{A}(\mathbf{x})$ used inside the Dirichlet penalty.

3.1. Experimental Setup

The five benchmark settings in Table 2 span synthetic binary tasks, tabular data, multiclass images, and CIFAR-10

Table 2. Accuracy (%, mean±std). PIEFS columns report test-split accuracy over five seeds. Classical baselines use the splits described in Sec. 3.1: validation split for Two Moons, Circles, and HTRU2, and official test split for MNIST and CIFAR-10 (R18 emb.). NeuralEF* reports five-seed test accuracy: for MNIST and CIFAR-10 (R18 emb.) from our rerun of the public codebase with $K=16$ linear probe, and for Two Moons, Circles, and HTRU2 (no public-codebase numbers available) from an RBF-kernel NeuralEF at the dataset-matched K. Bold marks the best PIEFS variant in each row. Note that NeuralEF* is *unsupervised* whereas the PIEFS columns here are *supervised*; a like-for-like unsupervised comparison is reported in App. C (Table 4).

Dataset	RF	LR	PCA+LR	NeuralEF*	PIEFS-off	PIEFS-diag	PIEFS-trotter
Two Moons	99.89±0.04	88.15±0.22	88.15±0.22	88.23±0.31	**99.87**±0.00	99.84±0.08	99.88±0.03
Circles	99.96±0.04	50.67±5.12	50.67±5.12	100.00±0.00	99.99±0.03	99.97±0.06	**100.00**±0.00
HTRU2	98.16±0.28	97.88±0.27	97.88±0.28	97.47±0.34	97.90±0.15	**97.97**±0.06	97.92±0.10
MNIST	97.07±0.03	92.20±0.00	85.94±0.00	82.52±0.29	**95.41**±0.32	94.91±0.28	94.94±0.20
CIFAR-10 (R18 emb.)	82.65±0.11	86.82±0.00	78.11±0.00	76.05±0.55	**85.50**±0.53	84.98±0.33	85.22±0.39

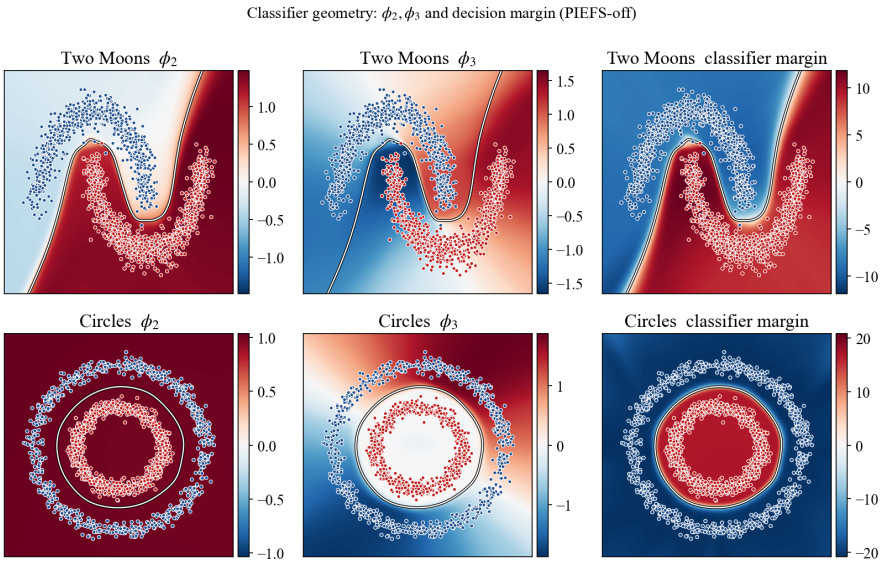


Figure 1. Classifier geometry induced by PIEFS coordinates. Columns 1–2 show ϕ_2 and ϕ_3 on Two Moons and Circles (PIEFS-off, one seed). Column 3 shows the signed classifier margin (linear-readout logit) pulled back to the input plane (x_1, x_2). Figure 2 shows the corresponding individual coordinate heatmaps for the same replicate.

ResNet-18 embeddings. The following paragraphs specify datasets, splits, coordinate counts K, and what is reported as validation vs. test.

Binary datasets. Synthetic benchmarks offer controlled evaluation and interpretability; accordingly, *Two Moons* and *Circles* are planar synthetic tasks with n=10,000 samples (Gaussian noise σ=0.1 for Two Moons and σ=0.05 for Circles) and a 70%/15%/15% split between training, validation, and test data (Pedregosa et al., 2011). *HTRU2* (Lyon et al., 2016; Lyon, 2016) provides eight tabular features for pulsar candidate classification on 17,898 samples with the same split and z-scored inputs.

Multiclass and embedding benchmarks. *MNIST* (LeCun & Cortes, 2010) is used at native resolution with 784 input dimensions; we use the standard 60k training and 10k test split, further subdividing the test set into 5k validation and 5k held-out evaluation; we set K=16. *CIFAR-10* (Krizhevsky, 2009) is represented by 512-dimensional penultimate activations of a pretrained ResNet-18 using the standard 45k training, 5k validation, and 10k test partition; we set K=16. Classical baselines use standardisation for LR/PCA+LR and raw $[0, 1]$ inputs for RF.

Protocols. Unless noted otherwise, Table 2 reports PIEFS **test**-split accuracy with mean and standard deviation over five random seeds. The NeuralEF column for MNIST and CIFAR-10 ResNet-18 embeddings reports **test** accuracy (mean ± std, five seeds, K=16 linear probe) from our rerun of the public codebase (Deng et al., 2022); Deng et al. additionally report **84.98**% MNIST test for their strongest CNN-GP configuration (Table 1), which is not identical to our linear-probe setting. Basis networks (ϕ_k) use three hidden layers of width 64; metric networks ($\mathbf{\Lambda}$, $\mathbf{U}$) use two hidden layers of width 64.

Classical baselines. Random forests (200 trees) and multinomial logistic regression on raw or standardised features; **PCA+LR** applies PCA with 16 components followed by the same logistic regression (Pedregosa et al., 2011). Validation split for low-dimensional tasks, official test for MNIST and CIFAR-10 embeddings.

NeuralEF baseline. **NeuralEF** (Deng et al., 2022) is an unsupervised spectral baseline with a linear probe; we report five-seed test numbers for MNIST and CIFAR-10 embeddings from our rerun of the public codebase under the same $K{=}16$ linear-probe protocol. For the low-dimensional and tabular datasets (Two Moons, Circles, HTRU2), where no public-codebase configuration is provided, we use an RBF-kernel NeuralEF implementation at the dataset-matched K with the same linear-probe evaluation.

3.2. Main Results

The three PIEFS columns in Table 2 are **off** ($\mathbf{A}{=}\mathbf{I}$), **diag** ($\mathbf{A}{=}\mathbf{\Lambda}$), and **trotter** ($\mathbf{A}{=}\mathbf{\Lambda U}$ with Sec. 2.4 apply order). Five-seed TROTTER on Two Moons, Circles, HTRU2, and MNIST follows the same protocol as the other rows; CIFAR-10 (R18 emb.) TROTTER is likewise aggregated over five independent seeds with the step budget stated above.

Result highlights. On the *linearly separable* HTRU2 dataset, both LR and PIEFS-off match RF, confirming the method does not degrade when the raw features are already informative. On *Two Moons*, where LR achieves only 88.2% (the linear boundary cannot separate the two crescents), PIEFS-off reaches 99.87%, demonstrating nonlinear separability beyond a linear boundary on raw inputs. On *MNIST*, PIEFS-off reaches 95.41% **test** accuracy at $K{=}16$. NeuralEF (Deng et al., 2022) uses a linear probe without labels in eigenfeature training; Deng et al. report **84.98**% MNIST test for their CNN-GP setup vs. **82.52**$\pm$**0.29**% in our $K{=}16$ rerun (Table 2); neither matches PIEFS test at $K{=}16$. On CIFAR-10 embeddings our NeuralEF rerun reaches **76.05**$\pm$**0.55**% test vs. **85.50**$\pm$**0.53**% PIEFS-off test. On *Circles* (well-separated concentric rings), all three variants reach near-perfect test accuracy ($\geq 99.9\%$). On CIFAR-10 embeddings, PIEFS-diag trails PIEFS-off slightly, underscoring that extra scaling does not always help high-level features.

3.3. Qualitative Geometry

Figure 1 visualizes how the learned coordinates induce nonlinear classifier margins in input space (Two Moons and Circles; PIEFS-off, one seed). Figure 2 shows the leading three coordinate maps ϕ_1, ϕ_2, ϕ_3 for the same replicate without the decision-boundary overlay, highlighting the spectral progression from the dominant mode ϕ_1.

3.4. Training Dynamics and Orthogonality

Figure 3 summarizes training dynamics and orthogonality diagnostics. Panel a shows validation accuracy over training steps for PIEFS-off on all five datasets (mean $\pm$ std over seeds). All datasets converge within the first 15,000 steps and remain stable thereafter. The leading coordinates carry most of the discriminative signal: on MNIST the first four maps (step $15{,}000$, $4{\times}3750$) already reach 94.5% validation accuracy, and the remaining twelve add only ~ 0.5 percentage points, consistent with a spectral ordering in which early modes dominate. Panel b tracks the batch Gram residual $\|C_k - I_k\|_F$ for PIEFS-off at $K{=}16$; because the outer schedule reweights Gram, classification, and MDE terms, the curve need not be monotone.

Figures 4–5 show the empirical Gram matrix C_K and its residual $C_K - I$ computed on the MNIST validation set after training (PIEFS-off, $K{=}10$, seed 0). The matrix is close to the identity; the Frobenius deviation $\|C_K{-}I\|_F{=}0.23$ indicates approximate rather than exact orthogonality, which is expected under finite-batch estimation (Remark 2.1) and remains well-behaved in practice.

Gradients use automatic differentiation (Paszke et al., 2019); informal wall-clock remarks appear in App. A.1.

The adaptive weights in (10) prioritize Gram and cross-entropy early, then emphasize MDE once residuals shrink, reducing conflict between oscillatory gradients and classification.

4. Discussion and Limitations

We present PIEFS, a neural feature-learning method based on a modified Dirichlet-energy objective. The method learns a sequence of scalar coordinate maps under an empirical Gram-orthogonality constraint, while a supervised cross-entropy term biases these coordinates toward the downstream classification task. The resulting features should therefore be interpreted as task-adaptive, Dirichlet-regularized spectral coordinates rather than eigenfunctions of a fixed, supervision-independent differential operator.

The main empirical observation is that the learned coordinates provide useful nonlinear representations on several benchmarks. On Two Moons, PIEFS separates a geometry that is not captured by a linear classifier on the raw input. On MNIST and CIFAR-10 ResNet-18 embeddings, the learned coordinates also give competitive test accuracy with a compact feature dimension. At the same time, the results do not uniformly dominate classical baselines: random forests and logistic regression remain strong on HTRU2. These cases indicate that the method is effective across several geometries while not uniformly dominating strong classical baselines.

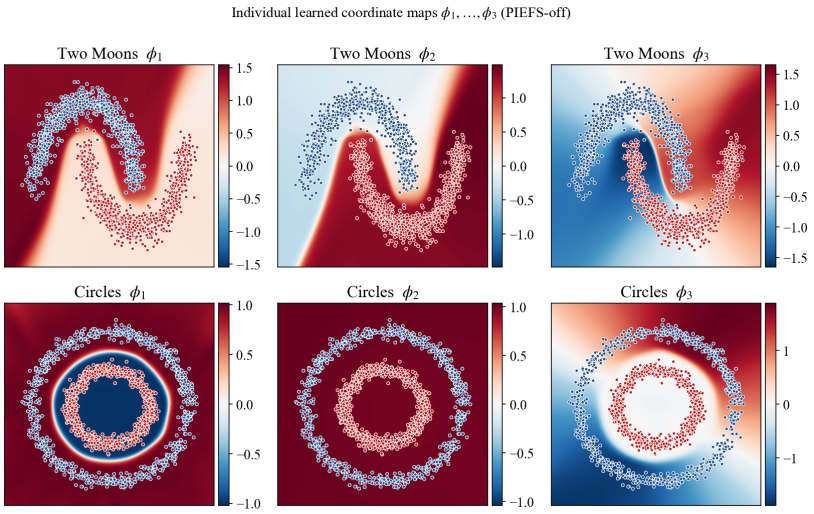
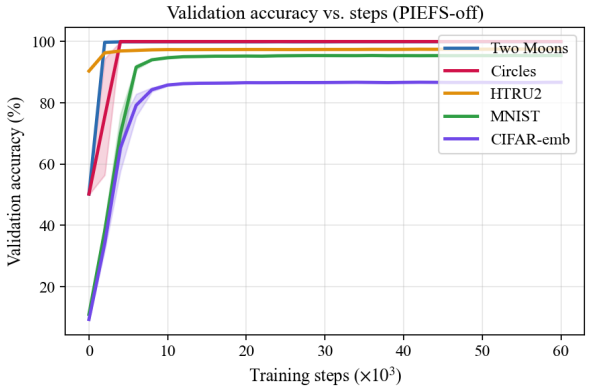
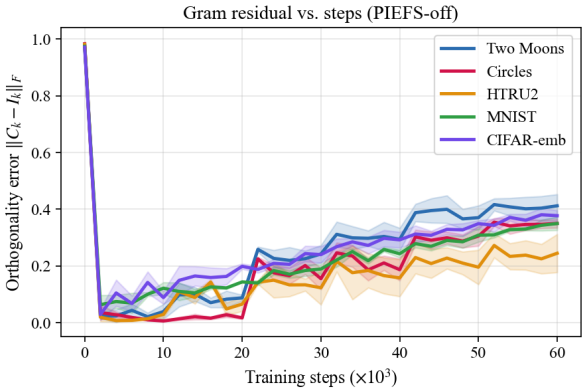


Figure 2. Leading learned coordinate maps ϕ_1, ϕ_2, ϕ_3 (spectral view). Each panel shows one coordinate map $\phi_j(x_1, x_2)$ as a heatmap; dots are validation points colored by class. Unlike Fig. 1, no decision-boundary contour is overlaid: these are the raw learned coordinates, showing the dominant mode ϕ_1 (absent from Fig. 1) and the progression to higher modes. PIEFS-off, same representative replicate as in Fig. 1.



(a) Validation accuracy vs. training steps for PIEFS-off across all five datasets (mean ± std over seeds).



(b) Orthogonality error (Frobenius norm $\|C_k - I_k\|_F$ of the batch Gram matrix vs. identity) for PIEFS-off across all five datasets (mean ± std over seeds). Non-monotone curves reflect dynamic loss weighting (10).

Figure 3. Training dynamics and orthogonality diagnostics (PIEFS-off, all five datasets). (a) Validation accuracy over training steps. (b) Batch Gram residual $\|C_k - I_k\|_F$. Curves show mean ± std over seeds.

The learnable metric $\mathbf{A}(\mathbf{x})$ provides a practical way to modify the Dirichlet penalty. The diagonal variant changes the relative scaling of gradient directions, while the TROTTER variant applies a product of Givens rotations followed by diagonal scaling. This parameterization keeps the rotational factor orthogonal by construction and avoids runtime QR or Gram–Schmidt orthogonalization. However, the current sparse Givens chain is not a full parameterization of $\mathrm{SO}(d)$, so its expressivity is limited. Richer orthogonal factors (e.g. dense skew-symmetric generators or matrix exponentials) would better approximate $\mathrm{SO}(d)$ but are costly at the ambient dimensions d used here; the structured Givens product is our stable baseline. Learning intermediate or hybrid rotation families remains an important open question.

A central limitation is the mathematical status of the learned coordinates. Since both the basis functions and, in some variants, the metric are trained with label supervision, the method does not recover the spectrum of a prescribed self-adjoint operator. Instead, it learns coordinates that balance three objectives: empirical orthogonality, classification accuracy, and Dirichlet-type smoothness. This distinction is important for interpreting the method: PIEFS is closer to supervised spectral representation learning than to a classical eigenvalue solver.

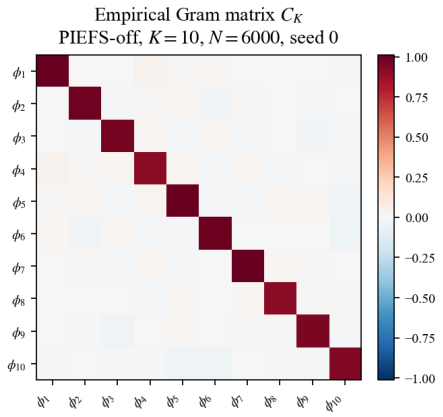


Figure 4. Empirical Gram matrix C_K on MNIST validation. Off-diagonal entries are near zero and diagonal entries near one, confirming approximate orthonormality. PIEFS-off, K=10, seed 0.

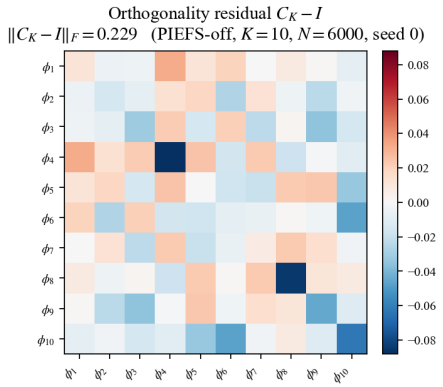


Figure 5. Orthogonality residual $C_K - I$ for the same replicate as Fig. 4. The largest deviations are $\approx$0.09; $\|C_K - I\|_F$=0.23, consistent with finite-batch estimation (Remark 2.1).

Another limitation is the use of finite-batch Gram penalties. The constraint $\widehat{C}_k \approx I_k$ is only a Monte Carlo approximation to global $L^2(p)$-orthogonality, and the residual may fluctuate during training as the classification and energy terms change the active coordinate. The dynamic weighting scheme mitigates this interaction but does not provide a monotone guarantee. This is visible in the Gram-residual traces (Fig. 3b), where the constraint can improve early and later deviate as other objectives become active.

The comparison with NeuralEF should also be interpreted with care. NeuralEF is an unsupervised spectral-feature method, whereas PIEFS uses label information through the cross-entropy term. The reported NeuralEF number is therefore a reference point for compact spectral representations, not a like-for-like comparison. An unsupervised ablation (App. C) repeats the pipeline with w_{class}=0 for an unsupervised reference.

The physics-informed component of PIEFS is the Modified Dirichlet Energy itself. Dirichlet energies arise in variational formulations of diffusion-type operators, elliptic PDEs, and spectral problems; here, the geometry of this energy is made data-adaptive through the learned metric $A(x)$. The experiments therefore evaluate a learned MDE-based representation method rather than a classical solver for a prescribed operator with known spectrum.

In summary, PIEFS provides a compact inductive feature map with a physics-inspired smoothness bias and inexpensive inference after training. The method is promising as a supervised spectral representation learner, but its current form should not be presented as a general-purpose neural eigensolver. The main directions for improvement are better optimization stability, richer metric parameterizations—especially flexible $\mathrm{SO}(d)$-valued factors beyond the implemented Givens chain—and validation on explicit PDE/operator eigenvalue problems.

Acknowledgements

The authors thank the HSE University supercomputer cluster (cHARISMa) (Kostenetskiy et al., 2021) for providing the computational resources used in this work.

Alexander Tarakanov gratefully acknowledges Nikolay Anokhin and Alexander D'yakonov of AI VK for fruitful discussions, valuable advice, and their generous support throughout the development of this work.

References

Belkin, M. and Niyogi, P. Towards a theoretical foundation for laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2007.08.006. URL https://www.sciencedirect.com/science/article/pii/S0022000007001274. Learning Theory 2005.

Choo, K., Mezzacapo, A., and Carleo, G. Fermionic neural-network states for ab-initio electronic structure. *Nature Communications*, 11, 05 2020. doi: 10.1038/s41467-020-15724-9.

Deng, Z., Shi, J., and Zhu, J. NeuralEF: Deconstructing kernels by deep neural networks. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4976–4992. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/deng22b.html.

Evans, L. C. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

Feld, T., Aujol, J.-F., Gilboa, G., and Papadakis, N. Rayleigh quotient minimization for absolutely one-homogeneous functionals. *Inverse Problems*, 35(6):064003, may 2019. doi: 10.1088/1361-6420/ab0cb2. URL https://dx.doi.org/10.1088/1361-6420/ab0cb2.

Ford, W. Chapter 5 - eigenvalues and eigenvectors. In Ford, W. (ed.), *Numerical Linear Algebra with Applications*, pp. 79–101. Academic Press, Boston, 2015. ISBN 978-0-12-394435-1. doi: https://doi.org/10.1016/B978-0-12-394435-1.00005-3. URL https://www.sciencedirect.com/science/article/pii/B9780123944351000053.

Gomez-Chova, L., Camps-Valls, G., Munoz-Mari, J., and Calpe, J. Semisupervised image classification with laplacian support vector machines. *IEEE Geoscience and Remote Sensing Letters*, 5(3):336–340, 2008. doi: 10.1109/LGRS.2008.916070.

Jin, H., Mattheakis, M., and Protopapas, P. Unsupervised neural networks for quantum eigenvalue problems, 2020. URL https://arxiv.org/abs/2010.05075.

Kostenetskiy, P. S., Chulkevich, R. A., and Kozyrev, V. I. HPC resources of the Higher School of Economics. *Journal of Physics: Conference Series*, 1740(1):012050, 2021. doi: 10.1088/1742-6596/1740/1/012050. URL https://iopscience.iop.org/article/10.1088/1742-6596/1740/1/012050.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Kunegis, J., Schmidt, S., Lommatzsch, A., Lerner, J., Luca, E. W. D., and Albayrak, S. *Spectral Analysis of Signed Graphs for Clustering, Prediction and Visualization*, pp. 559–570. doi: 10.1137/1.9781611972801.49. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611972801.49.

Lagaris, I., Likas, A., and Fotiadis, D. Artificial neural network methods in quantum mechanics. *Computer Physics Communications*, 104(1):1–14, 1997. ISSN 0010-4655. doi: https://doi.org/10.1016/S0010-4655(97)00054-4. URL https://www.sciencedirect.com/science/article/pii/S0010465597000544.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Lyon, R. HTRU2. 3 2016. doi: 10.6084/m9.figshare.3080389.v1. URL https://figshare.com/articles/dataset/HTRU2/3080389.

Lyon, R. J., Stappers, B. W., Cooper, S., Brooke, J. M., and Knowles, J. D. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123, 04 2016. ISSN 0035-8711. doi: 10.1093/mnras/stw656. URL https://doi.org/10.1093/mnras/stw656.

Ng, A., Jordan, M., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In Dietterich, T., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

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

A. Extended Protocols and Notes

A.1. Implementation and Runtime

Optimization uses mini-batch SGD with the budgets stated in the main text; hardware affects wall-clock but not reported accuracies. Approximating a single eigenfunction on a fine mesh can take on the order of two hours on commodity CPUs, whereas the sequential MNIST construction at d=784 is a few minutes per coordinate on a modern accelerator when using the same iteration budget. CPU cost remains a practical bottleneck; better initialization is left to future work.

B. Graph-Laplacian Pretraining Ablation

The optional Graph-Laplacian (GL) pretraining stage of Sec. 2.5 fits eigenmaps on a 1000-point subsample, distils the basis network against them for 2000 steps, and may calibrate T_{class} from a held-out logistic-regression cross-entropy (Gomez-Chova et al., 2008). Table 3 reports a five-seed HTRU2 ablation with PIEFS-off under three settings: GL off; GL on with T_{class} updated from the GL probe; and GL on with T_{class} frozen at 0.5. The three settings land within one standard deviation of one another on this low-dimensional benchmark.

Table 3. HTRU2 test accuracy (%, mean±std, 5 seeds). PIEFS-off with three GL settings.

Setting	Test accuracy
GL off (headline)	97.90±0.15
GL on, T_{class} updated	97.94±0.11
GL on, T_{class} frozen	97.97±0.10

C. Unsupervised PIEFS Ablation

We run an unsupervised variant of PIEFS with w_{class}=0 in (9). For this ablation we disable dynamic weighting entirely and use static weights w_{orth}=w_{mde}=1: the dynamic schedule in (10) gates w_{mde} on the class loss, which is no longer optimised when w_{class}=0, so we hold the weights fixed instead. The Gram orthogonality and MDE loss *terms* are otherwise unchanged. The basis network and metric heads keep the same capacity as the supervised runs (three hidden layers of width 64). A linear probe (logistic regression, default scikit-learn settings) is fit on the train-split coordinates and evaluated on the test split. Table 4 reports mean±std over five seeds; the NeuralEF column reuses the K=16 linear-probe rerun from Table 2.

Table 4. Unsupervised PIEFS test accuracy (%, mean±std, 5 seeds) with w_{class}=0. NeuralEF column reports the K=16 linear-probe rerun (same as Table 2).

Dataset	off	diag	trotter	NeuralEF
Two Moons	**99.95±0.09**	80.77±2.34	93.07±5.28	88.23±0.31
Circles	**100.00±0.00**	78.35±1.51	97.79±1.28	100.00±0.00
HTRU2	97.42±0.25	97.18±0.39	**97.53±0.54**	97.47±0.34
MNIST	**89.85±0.56**	78.59±1.25	75.39±1.01	82.52±0.29
CIFAR-10 (R18 emb.)	**79.80±0.26**	70.87±0.42	70.33±1.32	76.05±0.55

Unsupervised PIEFS-off matches or exceeds NeuralEF on four of five datasets (e.g. MNIST 89.9 vs. 82.5, CIFAR-10 embeddings 79.8 vs. 76.1). The learnable-metric variants, however, consistently *underperform* OFF without labels: they reach lower Gram error yet lower accuracy, because with w_{class}=0 the metric can trivially shrink the Dirichlet penalty by driving the volume-preserving scalings toward extreme anisotropy (we observe per-coordinate $\log \lambda$ ranges up to $\sim$4), so optimization capacity shifts to orthogonality at the expense of discriminative content; a per-coordinate anisotropy penalty $\text{mean}\,(\log \lambda_i)^2$ partially restores them toward OFF. These observations indicate that the learnable metric requires a supervisory or geometric signal to be useful, which we leave to future work.

Table 5. Final batch Gram error $\|C-I\|_F^2$ (lower = more orthonormal; mean over seeds) for supervised and unsupervised PIEFS. Without labels the metric variants reach *lower* Gram error than OFF (yet lower accuracy, Table 4): the volume-preserving metric trivially minimises the Dirichlet term, diverting capacity to orthogonality.

	Supervised			Unsupervised		
Dataset	off	diag	trotter	off	diag	trotter
Two Moons	0.16	0.05	0.05	1.84	0.04	0.05
Circles	0.12	0.04	0.05	0.94	0.25	0.05
HTRU2	0.09	0.02	0.03	0.31	0.02	0.02
MNIST	0.12	0.13	0.10	0.22	0.12	0.11
CIFAR-10 (R18 emb.)	0.14	0.12	0.11	0.31	0.10	0.11

D. Gram Orthogonality vs. Accuracy

Figures 6–7 plot the final Gram error $\|C - I\|_F^2$ against test accuracy for all dataset–metric combinations. In the supervised setting (Fig. 6) all variants achieve high accuracy regardless of their Gram error, confirming that the method does not require near-perfect orthogonality to generalise well. In the unsupervised setting (Fig. 7) the learnable-metric variants (DIAG, TROTTER) reach *lower* Gram error yet *lower* accuracy than OFF: without label supervision the metric shrinks the Dirichlet penalty by driving anisotropic scalings, diverting optimisation capacity from discriminative content to orthogonality.

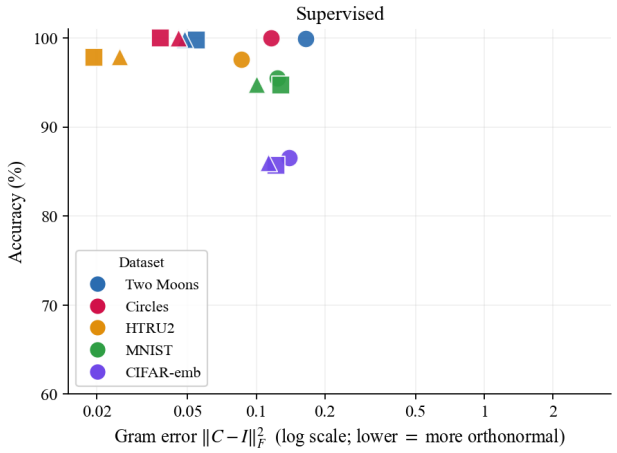


Figure 6. Supervised PIEFS: Gram error $\|C - I\|_F^2$ vs. test accuracy. All dataset–metric variants (circles = OFF, squares = DIAG, triangles = TROTTER) achieve high accuracy; orthogonality quality does not limit performance under supervision.

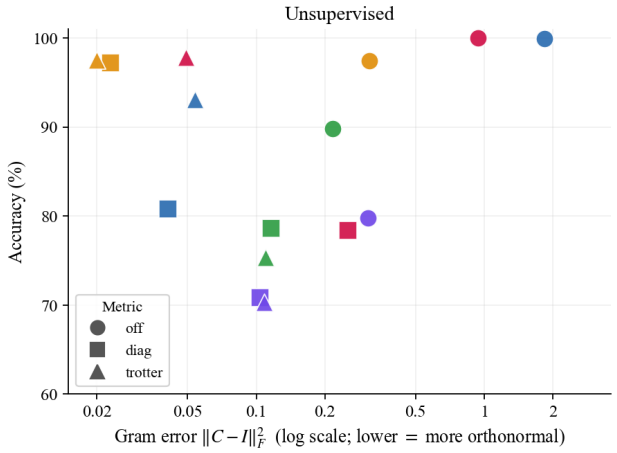


Figure 7. Unsupervised PIEFS: Gram error $\|C - I\|_F^2$ vs. test accuracy. Learnable-metric variants (squares, triangles) attain lower Gram error yet lower accuracy than OFF (circles), revealing the metric-degeneracy effect described in App. C.